Simultaneously Learning Corrections and Error Models for Geometry-based Visual Odometry Methods

Andrea De Maio[1] and Simon Lacroix[1]

Abstract— This paper fosters the idea that deep learning methods can be used to complement classical visual odometry pipelines to improve their accuracy and to associate uncertainty models to their estimations. We show that the biases inherent to the visual odometry process can be faithfully learned and compensated for, and that a learning architecture associated with a probabilistic loss function can jointly estimate a full co-variance matrix of the residual errors, defining an error model capturing the heteroscedasticity of the process. Experiments on autonomous driving image sequences assess the possibility to concurrently improve visual odometry and estimate an error associated with its outputs.

I. INTRODUCTION

Visual odometry (VO) is a motion estimation process successfully applied in a wide range of contexts such as autonomous cars or planetary exploration rovers [1]. Seminal works largely resorted to stereovision. By tracking point features in images, 3D points correspondences are used to recover the motion between two stereoscopic acquisitions. The integration of elementary motions yields an estimate of the robot pose over its course. Continuous work on VO led to a well established processes pipeline, composed of feature extraction, matching, motion estimation, and finally optimization. This scheme has been extended to single camera setups, in which case motions are estimated up to a scale factor, retrieved *e.g.* by fusing inertial information. Direct methods for VO have also recently been proposed. They bypass the feature extraction process and optimize a photometric error [2]. These methods overcome the limits of sparse feature-based methods in poorly textured environments or in presence of low quality images (blurred), and they have proven to be on average more accurate.

The advent of convolutional neural networks (CNN) sprouted alternate solutions to VO. The full estimation process can be achieved by deep-learning architectures in an end-to-end fashion (see *e.g.* [3], [4], and especially [5] – note these work consider the monocular version of the problem, leaving the scale estimation untackled). In such approaches, the system has to learn the various information necessary to perform vision-based egomotion estimation, which can be a daunting task for a CNN. This paper builds upon existing work that exploits a CNN to predict *corrections* to classic stereo VO methods [6], aiming at improving their precision. We argue that complementing classical localization processes with learning-based methods can return better results than delegating the full pose estimation process to a CNN. On

[1]LAAS-CNRS, Université de Toulouse, 7, Avenue du Colonel Roche, 31031 Toulouse, France, E-mail: {andrea.de-maio,simon.lacroix}@laas.fr
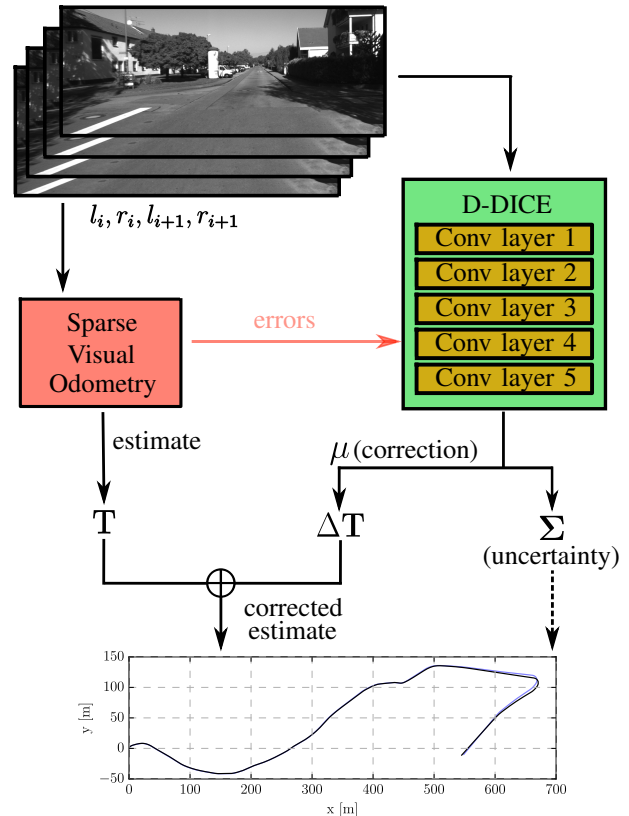
Fig. 1: D-DICE produces corrections to classic visual odometry methods in a probabilistic framework. The system generates full covariance matrices that can be used to minimize errors in pose-graph optimization.

the one hand, classical localization processes do not output totally erroneous poses, as a learning approach could when confronted with features unseen in the training set. On the other hand, classical processes can be monitored by some explicit indicators (e.g. number of tracked points in VO, inliers, etc.), so as to detect erroneous cases. Our developments consider that visual odometry estimation errors do not have zero mean, as assessed in *e.g.* [7], [8], and provide corrections that improve the precision of VO. Furthermore, at the same time, they produce a full error model for each computed motion estimation (in form of a Gaussian model), akin to [9]. This is a significant achievement, as it is generally complex to derive precise error models for geometrical VO methods.

II. PROBLEM STATEMENT AND RELATED WORK

Consider a robot moving in a three dimensional environment. Let $x_i \in \mathbb{R}^6$ be its pose (3 translations and 3

orientations) at time i in a given reference frame. The actual motion (ground truth) between time instants i and $i+1$ is represented by a homogeneous transformation matrix ${}^i\mathbf{T}_{i+1}$.

A vision-based motion estimator uses raw image data $\boldsymbol{\mathcal{I}}_i \in \mathbb{R}^n$ to obtain an estimate ${}^i\hat{\mathbf{T}}_{i+1}$. In the VO case, the raw data $\boldsymbol{\mathcal{I}}_i$ is a pair of monocular or stereoscopic images captured at two different time instants i, $i+1$ (*i.e.* 2 or 4 images). The error $\boldsymbol{e}_i$ of VO is:

$$\boldsymbol{e}_i = {}^i\mathbf{T}_{i+1} \cdot {}^i\hat{\mathbf{T}}_{i+1}^{-1} \tag{1}$$

We can create a dataset $\mathcal{D} = \{\boldsymbol{\mathcal{I}}_i, \boldsymbol{e}_i | \forall i \in [1, d]\}$, where d is the size of the dataset. The literature provides two different approaches to leveraging this type of dataset. The two approaches enhance a classic VO process with learning to either estimate *(i)* a *motion correction* to apply to ${}^i\hat{\mathbf{T}}_{i+1}^{-1}$, thus improving its accuracy [6], or *(ii)* an *error model* associated with ${}^i\hat{\mathbf{T}}_{i+1}^{-1}$ [9], thus allowing its fusion with any other motion or pose estimation process. Alternatively, with the same semantic, substituting errors with actual motion transforms, it is possible to directly learn the motion estimate and associated error [5].

A. Directly learning VO and an error model

The work in [5] introduces an end-to-end, sequence-to-sequence probabilistic visual odometry (ESP-VO) based on a recurrent CNN. ESP-VO outputs both a motion estimate ${}^i\hat{\mathbf{T}}_{i+1}^{-1}$ and an associated error. The learned error model is a diagonal covariance matrix, hence not accounting for possible correlations between the different motion dimensions. It is unclear how the probabilistic loss is mixed to the mean squared error of the Euclidean distance between the ground truth and the estimated motions. Finally, the authors make use of a hand-tuned scaling factor to balance rotation and translation. The article presents significant results obtained on a large variety of datasets, with comparisons to state-of-the-art VO schemes. The results show that ESP-VO is a serious alternative to classic schemes, all the more since it also provides variances associated with the estimations. Yet, they are analysed over whole trajectories, which inherit from the random walk effect of motion integration, and as such do not provide thorough statistical insights – *e.g.* on the satisfaction of the gaussianity of the error model or on the evaluation of the mean log-likelihood.

B. Learning corrections to VO

The work presenting DPC-Net [6] learns an estimate of $\boldsymbol{e}_i$, which is further applied to the VO estimate ${}^i\hat{\mathbf{T}}_{i+1}^{-1}$ to improve its precision. The authors introduce an innovative pose regression loss based on the SE(3) geodesic distance modelled with a vector in Lie algebra coordinates. Instead of resorting to a scalar weighting parameter to generate a linear combination of the translation and rotation errors, the proposed distance function naturally balances these two types of errors. The loss takes the following form:

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

where $\boldsymbol{\xi} \in \mathbb{R}^6$ is a vector of Lie algebra coordinates estimated by the network, $g(\boldsymbol{\xi})$ computes the equivalent of (1) in the Lie vector space, and $\boldsymbol{\Sigma}$ is an empirical average covariance of the estimator pre-computed over the training dataset. Such covariance matrix cannot be used as an uncertainty measure but only as a balancing factor between rotation and translation terms. The paper provides statistically significant results showing that DPC-Net improves a classic feature-based approach, up to the precision of a dense VO approach. In particular, it alleviates biases (*e.g.* due to calibration errors) and environmental factors. The system interlace low rate corrections with estimates produced by the underlying VO, which processes all the images, using a pose-graph relaxation approach.

C. Learning an error model of VO

Inferring an error model for VO comes to learn the parameters of a predefined distribution to couple VO with uncertainty measures. The work in [9] introduces DICE (Deep Inference for Covariance Estimation), which learns the covariance matrix of a VO process as a maximum-likelihood for Gaussian distributions. Nevertheless, it considers the distribution over measurement errors as a zero-mean Gaussian $\mathcal{N}(0, \boldsymbol{\Sigma})$. Such model is acceptable for unbiased estimators, which unfortunately it is often not the case of VO. Yet, the authors show that their variance estimates are highly correlated with the VO errors, especially in case of difficult environmental conditions, such as large occlusions.

III. SIMULTANEOUSLY LEARNING CORRECTIONS AND UNCERTAINTIES

To jointly estimate a correction to the VO process and a full error model *after having applied the correction*, we initially enhance the network output of [9] adding a vector $\boldsymbol{\mu}_i \in \mathbb{R}^6$ to the output layer to account for biases in the estimator. Such vector is incorporated in the negative log-likelihood loss that is derived as follows. Given a dataset $\mathcal{D}$ of size d, where the observations $\{\boldsymbol{e_1}, \ldots, \boldsymbol{e_d}\}^\mathsf{T}$ of VO errors are assumed to be independently drawn from a multivariate Gaussian distribution, we can estimate the parameters of the Gaussian as

$$\arg\max_{\boldsymbol{\mu}_{1:d}, \boldsymbol{\Sigma}_{1:d}} \sum_{i=1}^{d} p(\boldsymbol{e}_i | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{3}$$

This is equivalent to minimize the negative log-likelihood

$$\arg\min_{\boldsymbol{\mu}_{1:d}, \boldsymbol{\Sigma}_{1:d}} \sum_{i=1}^{d} -\log\left(p(\boldsymbol{e}_i | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\right) \tag{4}$$

$$= \arg\min_{\boldsymbol{\mu}_{1:d}, \boldsymbol{\Sigma}_{1:d}} \sum_{i=1}^{d} \log|\boldsymbol{\Sigma}_i| + (\boldsymbol{e}_i - \boldsymbol{\mu}_i)^\mathsf{T} \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{e}_i - \boldsymbol{\mu}_i) \tag{5}$$

$$\approx \arg\min_{f_{\boldsymbol{\mu}_{1:d}}, f_{\boldsymbol{\Sigma}_{1:d}}} \sum_{i=1}^{d} \log|f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i)| + \\ (\boldsymbol{e}_i - f_{\boldsymbol{\mu}}(\boldsymbol{\mathcal{I}}_i))^\mathsf{T} f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i)^{-1} (\boldsymbol{e}_i - f_{\boldsymbol{\mu}}(\boldsymbol{\mathcal{I}}_i)) \tag{6}$$

We split the output of the network in two parts: the mean vector $f_{\boldsymbol{\mu}}(\boldsymbol{\mathcal{I}}_i)$ and the covariance matrix $f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i)$, where $f(\boldsymbol{\mathcal{I}}_i)$ represents the full output given a pair of stereo images.

A. Retrieving a valid covariance matrix

To enforce a positive definite covariance matrix we tested two different matrix decompositions.

1) LDL: The first one is the LDL matrix decomposition as in [9], to which the reader can refer to for a complete description. The predicted covariance matrix $f_{\boldsymbol{\Sigma}_i}(\boldsymbol{\mathcal{I}}_i)$ is generated through a vector $\boldsymbol{\alpha}_i = [\boldsymbol{l}_i, \boldsymbol{d}_i]^{\mathsf{T}}$, with $\boldsymbol{l}_i \in \mathbb{R}^{(\frac{n^2}{2} - \frac{n}{2})}$ and $\boldsymbol{d}_i \in \mathbb{R}^n$. We have then

$$\boldsymbol{\Sigma}_i \approx f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i) = L(\boldsymbol{l}_i)D(\boldsymbol{d}_i)L(\boldsymbol{l}_i)^{\mathsf{T}} \quad (7)$$

where $\boldsymbol{l}_i$ and $\boldsymbol{d}_i$ are the vectors containing the elements of the respective $\boldsymbol{L}$ and $\boldsymbol{D}$ matrices. The LDL decomposition is unique and exists as long as the diagonal of $\boldsymbol{D}$ is strictly positive. This can be enforced using the exponential function $exp(\boldsymbol{d_i})$ on the main diagonal. By doing so, the computation of its log-determinant, i.e. the first term of (6), can be reduced to $sum(\boldsymbol{d}_i)$, that is the sum of the elements in the vector $\boldsymbol{d}_i$. In the second term $f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i)^{-1}$ is replaced by the LDL product.

2) LL: Alternatively, it is possible to resort to the classical Cholesky decomposition. In this case $f_{\boldsymbol{\Sigma}}(\boldsymbol{\mathcal{I}}_i)$ is replaced by the $\boldsymbol{L}\boldsymbol{L}^*$ product, where $\boldsymbol{L}$ is a lower triangular matrix and $\boldsymbol{L}^*$ is its conjugate transpose. We consider $\boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}$ as no complex number is involved in our work. The $\boldsymbol{L}$ matrix can be generated through a vector $\boldsymbol{l}_i \in \mathbb{R}^{(\frac{n^2}{2} + \frac{n}{2})}$. This decomposition also has nice properties around its log-determinant, as it is easy to prove $\log|\boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}| = 2\sum_i \log(\boldsymbol{L}_{ii})$.

We tested both decompositions and did not experience relevant changes in the network accuracy. Although the presence of the exponential term in the LDL can alter the numerical stability of the loss function, clamping its value mitigates this problem. Therefore, we decided to pursue training using the first method in order to introduce fewer variables that could affect the comparisons in the results (Sec.IV-C).

B. Optimization problem

We incorporate the LDL formulation for the covariance matrix in the negative log-likelihood. Replacing $f_{\boldsymbol{\mu}}(\boldsymbol{\mathcal{I}}_i)$ with the estimated mean output vector $\hat{\boldsymbol{\mu}}_i$ we finally obtain

$$\mathcal{L}(\boldsymbol{\mathcal{I}_{1:d}}) = \underset{\hat{\boldsymbol{\mu}}_{1:d}, \boldsymbol{\alpha}_{1:d}}{\arg\min} \sum_{i=1}^{d} sum(\boldsymbol{d}_i) + \quad (8)$$
$$(\boldsymbol{e}_i - \hat{\boldsymbol{\mu}}_i)^{\mathsf{T}}(L(\boldsymbol{l}_i)D(exp(\boldsymbol{d}_i))L(\boldsymbol{l}_i)^{\mathsf{T}})^{-1}(\boldsymbol{e}_i - \hat{\boldsymbol{\mu}}_i)$$

Formulating the problem as in (8), the second term of the loss function loosely recalls the formulation of the Lie algebra loss in (2). The covariance matrix in this case is learned in relation to the input, capturing the heteroscedastic uncertainty of each sample. The learned covariance matrix acts as in [10], weighing position and orientation errors. The main difference resides in the nature of the learned

uncertainty, homoscedastic vs heteroscedastic: through back-propagation with respect to the input data, [11], we aim to learn a heteroscedastic error. Assuming that errors can be drawn from a distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, $\boldsymbol{\mu}_i$ matches the expected value for the predicted distribution. This corresponds to the desired correction in our case. At the same time, we estimate a covariance matrix $\boldsymbol{\Sigma}_i$, returning an uncertainty measure relative to each particular input and pose after the predicted correction. Inverting Eq.1, corrections are applied as

$$^{i}\mathbf{T}_{i+1} \approx \, ^{i}\mathbf{T}_{i\hat{+}1}^{VO} \, \cdot \, ^{i\hat{+}1}\mathbf{T}_{i+1}^{corr} \quad (9)$$

that is the composition of the pose estimate produced by VO and the estimated correction produced by the neural network.

IV. Experiments

A. Setup

1) Dataset: We carry out experiments using the KITTI dataset, which provides various sequences of rectified images acquired while driving in urban areas [12]. Depth images are generated using a semi global block matching algorithm to generate and filtered using a weighted least squares filter. We train the network splitting train and validation trajectories in different configurations: for all results shown here we trained using sequences `04` to `10` (which share the same calibration parameters), excluding one for testing and one for validation purposes. Most of our experiments are validated using four sequences (`05`, `06`, `09`, `10`).

For the initial motion estimates, we use the open-source VO implementation *libviso2* [13]. It is a feature-based approach, that uses a Kalman Filter in combination with a RANSAC scheme to produce SE(3) estimates using rectified stereoscopic pairs. The estimated corrections, and relative uncertainties, are expressed in camera frame (z axis pointing forward), and the Tait-Bryan angles are defined w.r.t. this reference frame (*i.e.* yaw encodes rotations around the optical axis z).

2) Evaluation metrics: To evaluate the precision of the motion estimates, we make use of the absolute trajectory error (ATE) metric [14], [12]. It is defined as

$$ATE_{rot} = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{e}_i^R||_2$$
$$\quad (10)$$
$$ATE_{trans} = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{e}_i^t||_2$$

where $\boldsymbol{e}_i^t$ is the difference between the estimated position and the ground truth, and $\boldsymbol{e}_i^R$ is the rotation angle, in angle-axis representation, of the product $\boldsymbol{R}_i\hat{\boldsymbol{R}}_i^{\mathsf{T}}$. ATE comes with the advantage of returning a single scalar to evaluate rotation and translation errors, making it easy to compare them among multiple estimators. At the same time, its main disadvantage lies in the lack of robustness to isolated poor estimations and their relative position in the trajectory [14], [12], [15]. We use ATE for early architectural choices (section IV-B), but in order to provide a more informative analysis, we use relative error statistics. The idea of relative error is to select segments

of predefined lengths of the trajectory and compute the error on all the aligned sub-trajectories. This way, it is possible to obtain statistics on the tracking error (mean and standard deviation) and evaluate it for short or long-term accuracy [6], [14]. In our evaluations, we select sub-trajectories of 10, 20, 30, 40 and 50 % of the full trajectory length.

3) Network structures: We initially compared the results produced using the architectures presented in DPC-Net [6] and DICE [9]. The first trial has been to adapt the loss defined in Eq. 8 to DPC-Net. We noticed that the mean output vector was still rather constant throughout entire trajectories, regardless of the dataset, and the same behavior was experienced using the loss defined in Eq. 2. Similar tests were conducted with DICE. We experienced problems in reducing the average mean error along all the six dimensions at the same time, and an increase in its standard deviation. Alleging these issues as being caused by the shallow architecture of DICE, we modified its network structure, first removing the max pool layers to preserve spatial information [16], and achieving dimension reduction by setting the stride to 2 in early layers. We also increased the number of convolutional filters to tackle the estimation of both corrections and error models, adding dropout after each layer to prevent overfitting. For the rest of the paper, we refer to this network as Deeper-DICE (D-DICE, Table I).

The convolutional layers are followed by two fully connected layer, respectively composed of 256 and 27 output units. In the six-dimensional case, we need 21 values for the LDL decomposition and 6 for the mean vector. We trained the network using both monocular images and stereo images. Additionally, we explored if pairing monocular images to their corresponding disparity maps could be beneficial to the training process, even if the disparities were produced separately from VO.

We used the Adam optimizer with a learning rate of 1e-04 and halted the learning when test and train loss started diverging. All the experiments have been carried out using an Nvidia GeForce RTX 2080 Ti with a batch size of 32.

Layer	Kernel size	Stride	Number of channels
conv1	5x5	2	64
conv2	5x5	2	128
conv3	3x3	2	256
conv4	3x3	2	512
conv5	3x3	1	1024

TABLE I: D-DICE convolutional architecture.

4) Influence of input data: We tested the proposed architecture and loss using three different types of input data: monocular images, monocular images with associated disparity images, and stereo image pairs. Table II shows the ATEs on two sequences for the three input data: the stereo setup outperforms the two others in both sequences. Ideally, one would expect the best results with monocular images associated with disparity images: indeed, with such data the network does not have to infer depth information.

However, with respect to monocular images, this input data does not improve the ATE metric as much as stereo pairs. It is likely that efficiently exploiting disparity images would require a specific convolutional architecture, as their nature differs significantly from intensity images.

Throughout the remainder of the paper, we will provide results with D-DICE obtained using stereo images, as for DPC-Net.

ATE	Seq.	D-DICE (mono)	D-DICE (mono+disp)	D-DICE (stereo)
Trans. (m)	05	21.54	19.73	**10.23**
	10	12.15	8.56	**7.20**
Rot. (deg)	05	5.81	5.30	**2.62**
	10	3.01	2.47	**2.28**

TABLE II: Mean Absolute Trajectory Error (ATE) on two validation sets for three different input data types.

B. Qualitative Evaluation

To validate the choice of the proposed architecture and loss, we discuss preliminary results obtained exploring different possibilities in this regard.

1) Loss comparison: To show the impact of dynamically estimating a covariance matrix for each transform, we adapt the loss in Eq. 2 to our proposed architecture. Table III compares the ATEs obtained with this loss and the one we proposed in Eq. 8. As we are going to showcase in further results, the negative log-likelihood loss generally outperforms the loss associated with DPC-Net. At the same time, we experience a more stable improvement in translation than in rotation. This is especially true in sequences 09 and 10, where VO presents smaller errors (see figure 3). We associate this behavior to the need for tailored large uncertainties to poor estimations, as opposed to sequences where the necessity of corrections is lower due to a better tracking by VO. Besides, our loss yields the prediction of an error model, which can be used to further reduce trajectory errors, as shown in section IV-C.2.

ATE	Estimator	05	06	09	10
Translation (m)	D-DICE + Lie	20.22	5.51	19.51	11.60
	D-DICE + NLL	**10.23**	**4.65**	**16.50**	**7.20**
Rotation (deg)	D-DICE + Lie	4.19	1.85	**2.30**	**1.77**
	D-DICE + NLL	**2.62**	**0.84**	2.79	2.28

TABLE III: Comparison of ATEs obtained with the Lie loss and the negative log-likelihood.

2) Architecture comparison: Selecting an appropriate architecture was driven by a few factors. A first issue was represented by the significantly higher incidence of numerical instability when pairing it with a NLL loss, particularly in the matrix inversion and in the exponential. We also noticed that the DPC-Net system (architecture + loss) tends to output rather constant corrections throughout a whole sequence, certainly compensating biases. This behavior was less prominent when training D-DICE with the same loss.

(a) DPC-Net. Translation corrections.

(b) DPC-Net. Rotation corrections.

(c) D-DICE. Translation corrections.

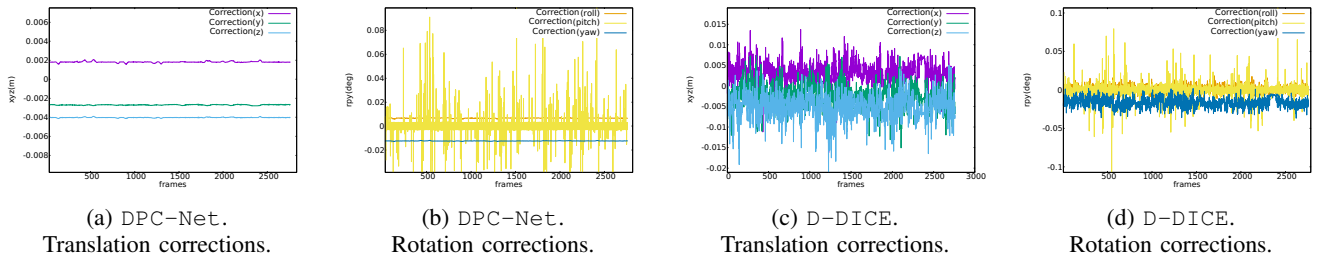(d) D-DICE. Rotation corrections.

Fig. 2: Estimated translation and rotation corrections using DPC-Net and D-DICE over time.

The two systems behave quite differently, as can be seen in Fig. 2, with D-DICE exhibiting more data-dependent corrections. The architectures of DICE and D-DICE are thoroughly compared in Section IV-C.2.

C. Quantitative Evaluation

We evaluate D-DICE performances in two schemes. First, we make use of the estimated mean vector as corrections for each image pair (Sec. IV-C.1). This approach does not use the uncertainty model and ensures a fair comparison with DPC-Net systems producing corrections outside a probabilistic context. With respect to uncertainty models (Sec. IV-C.2), we test the Gaussian assumption and compare log-likelihood values. Additionally, we use the covariance information to further reduce trajectory errors. In a first step, we correct the trajectory as in Eq. 9. Subsequently, we solve a pose-graph optimization problem, weighting errors on the inverse of the covariance matrix and manually adding a ground-truth loop closure.

1) Trajectory correction: We evaluate the accuracy of the correction produced by D-DICE with respect to the chosen baseline VO solution (Fig. 3). We also compare to DPC-Net, trained on the same data split as D-DICE. In these comparisons, both systems estimate transforms with consecutive stereo pairs, and the uncertainty estimates provided by D-DICE are not used. D-DICE outperforms DPC-Net in all the selected sequences (Table IV). While both systems constantly improve *libviso2*, we noticed a larger improvement in the translation errors especially on the y-axis (upwards) – the vertical drift is one of the most prominent weaknesses of this VO implementation using the KITTI dataset. For more in-depth results, we show mean relative segment errors with associated standard deviations in Table V.

ATE	Estimator	05	06	09	10
Translation (m)	Sparse VO	25.90	7.75	52.78	11.79
	DPC-Net	11.04	5.83	23.26	11.67
	D-DICE	**10.23**	**4.65**	**16.49**	**7.20**
Rotation (deg)	Sparse VO	7.73	3.77	6.92	5.19
	DPC-Net	**2.10**	2.25	4.18	4.19
	D-DICE	2.62	**0.84**	**2.79**	**2.28**

TABLE IV: Mean Absolute Trajectory Error (ATE) before and after applying corrections to *libviso2* VO

2) Uncertainty estimation: Since we assume a Gaussian error model $\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, we can measure its relevance by checking the fraction of samples that do not respect the following inequality:

$$\mu_i - n\sigma_i \leq e_i \leq \mu_i + n\sigma_i \tag{11}$$

where e_i is the error along the dimension i after correction, and μ_i, σ_i are respectively the mean and standard deviation predicted for the input associated with $\boldsymbol{e}$ on the i-th dimension. The parameter n is the number of considered standard deviations. We test against the three-sigma interval ($n = 3$) that, in case of samples drawn from a normal distribution, should cover approximately 99.7% of the samples. Statistics for a pair of KITTI sequences using D-DICE are displayed in Table VI. On average, more than 99% of samples falls within the three-sigma interval.

A common way of evaluating uncertainties is to inspect the mean log-likelihood value. Even though it is not possible to use it alone as a measure of fitness, it can be used to compare different distribution parameters. The log-likelihood directly describes how well the estimated distributions capture the errors in the dataset and is easily obtained as it is the function minimized by our neural network. Table VII shows benchmarks for the predicted distributions obtained by DICE and D-DICE with different losses. While it is true that a higher log-likelihood does not systematically translate into a reduced trajectory error, we found this trend generally confirmed in our experiments.

Finally, to study the utility of covariances in diminishing tracking errors, we set up an error minimization problem using a typical 3D pose-graph formulation. The optimizer seeks to minimize the function

$$\sum_{i \in \mathcal{D}} \boldsymbol{e}_{i,i+1}^{\mathsf{T}} \boldsymbol{\Sigma}_i^{-1} \boldsymbol{e}_{i,i+1} + \boldsymbol{e}_{d,1}^{\mathsf{T}} \boldsymbol{\Sigma}_{gt}^{-1} \boldsymbol{e}_{d,1} \tag{12}$$

where $\boldsymbol{e}_{i,i+1}$ is the error function between the nodes $\langle i, i+1 \rangle$ and the corresponding constraint $\boldsymbol{z}_{i,i+1}$. A loop closure is manually triggered adding the ground truth measurement $\boldsymbol{z}_{d,1}$ and a small covariance $\boldsymbol{\Sigma}_{gt}$ (practically, forcing the last point of the trajectories to fit the ground truth). We use the framework *g2o* to formulate and solve the problem using a Levenberg-Marquardt optimizer [17]. The results of the inclusion of the predicted covariances after pose-graph optimization are summarized in Tables VIII and IX. To provide a complete overview we show D-DICE and

Relative segment errors	Estimator	05	06	09	10
Translation (%)	Sparse VO	2.51 ± 1.88	1.30 ± 0.52	3.11 ± 1.66	1.16 ± 0.67
	DPC-Net	1.51 ± 0.71	1.76 ± 0.97	1.64 ± 0.84	1.27 ± 0.56
	D-DICE	**1.01 ± 0.44**	**0.91 ± 0.46**	**1.17 ± 0.53**	**0.95 ± 0.41**
Rotation (millideg/m)	Sparse VO	10.31 ± 10.40	9.62 ± 4.13	11.19 ± 2.73	12.35 ± 4.60
	DPC-Net	4.50 ± 2.47	7.50 ± 3.20	4.80 ± 1.92	7.75 ± 2.61
	D-DICE	**3.29 ± 1.47**	**3.25 ± 2.02**	**4.67 ± 2.36**	**5.82 ± 2.58**

TABLE V: Relative segment errors (mean error ± standard deviation, computed on all relative segments for each sequence) before and after applying corrections to *libviso2* VO.

	σ	2σ	3σ
x	85.55%	97.45%	98.64%
y	83.82%	98.00%	99.36%
z	76.27%	95.82%	99.27%
$roll$	77.55%	96.00%	99.18%
$pitch$	85.64%	97.27%	98.91%
yaw	74.27%	96.64%	99.27%
mean	80.51%	96.86%	99.10%
	σ	2σ	3σ
x	87.24%	98.91%	99.63%
y	91.15%	99.42%	99.67%
z	79.49%	96.12%	98.94%
$roll$	75.72%	94.89%	98.69%
$pitch$	72.97%	95.14%	98.76%
yaw	70.57%	93.55%	98.69%
mean	79.52%	96.34%	99.07%

TABLE VI: Sequence 06 (top) and 05 (bottom). Percentages of samples that lie in the various sigma-intervals around the mean. Mean and standard deviations are produced by D-DICE.

Estimator	05	06	09	10	**mean**
DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	41.53	44.42	41.77	42.72	42.61
D-DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	43.15	44.24	41.77	42.14	42.82
DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	42.04	44.67	42.27	41.86	42.71
D-DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	44.08	44.67	41.66	41.60	**43.00**

TABLE VII: Mean log-likelihood for different network architectures and losses.

ATE	Estimator	05	06	09	10
Trans. (m)	DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	126.59†	**9.78**	**51.80†**	4.13
	D-DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	**7.97**	12.75	52.16†	**4.02**
	DPC-Net	5.73	8.14	4.00	4.22
	DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	123.21†	3.87	4.96	3.63
	D-DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	**5.64**	**2.54**	**3.61**	**3.02**
Rot. (deg)	DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	106.32†	3.42	15.00†	2.70
	D-DICE $\mathcal{N}(0, \boldsymbol{\Sigma})$	**3.06**	3.42	13.66†	**2.63**
	DPC-Net	2.12	2.40	**1.13**	1.74
	DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	106.46†	1.11	1.37	1.31
	D-DICE $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	**1.37**	**0.77**	1.52	**0.87**

TABLE VIII: ATE for 5 different estimators. Networks paired with $\mathcal{N}(0, \boldsymbol{\Sigma})$ do not use bias estimation to minimize the NLL. Values tagged with † point to cases where the optimizer noticeably got stuck in a local minimum.

DICE errors obtained considering biases or zero-mean Gaussians. Additionally, we use DPC-Net corrections with a fixed covariance. D-DICE consistently shows smaller errors compared to the other methods. It is worth to notice that without the corrections, particularly for long trajectories, the initial problem state given by VO may lock the optimizer in a local minimum. This mainly happens with DICE (Seq 05, 09) but also with D-DICE when a zero-mean normal distribution is considered (Seq 09).

Despite using only the latest left image available to VO, DICE is able to have good results on all datasets considered. Still, on average, D-DICE outperforms it. While it is hard to assess the exact reason for the more accurate results, we argue that such improvements are due to two major factors. First of all, our network has access to a larger spectrum of information, the stereo images, which are also available to the VO algorithm. Such information can be exploited to retrieve absolute scale for the translation part. This assumption is backed up by experiencing an average larger standard deviation for the translation part when using monocular images instead of stereo pairs. Additionally, even in the monocular case, having both left images gives the information necessary to extract the robot motion. In this case, the derived uncertainty will not be based only on contextual information (e.g. lack of texture, blurred images) but also on the type/magnitude of motion. Secondly, accounting for non-zero mean allows a further minimization of the negative log-likelihood. D-DICE can rely on estimating both the Gaussian parameters to minimize the same loss.

V. CONCLUSIONS

We presented an insight into the learning of errors in visual odometry. Relying on existing state-of-the-art techniques, we iterated on analysing what type of error and uncertainty can be learned by deep neural networks. We concentrated our efforts on approaches that complement classical visual odometry pipelines in order to ease the work done by the network and exploiting a robust and well established feature-based processes. We demonstrated that it is possible to assimilate the distribution over visual odometry errors to Gaussians, and proceeded to cast the error prediction to a full maximum likelihood for normal distributions case. Knowing that the errors are biased, we have modelled such Gaussians as non-zero mean distributions, showing

Relative segment errors	Estimator	05	06	09	10
Translation (%)	DICE $\mathcal{N}(0, \Sigma)$	$161.74 \pm 23.24^{\dagger}$	$\mathbf{2.16 \pm 1.28}$	$3.89 \pm 4.27^{\dagger}$	1.23 ± 0.59
	D-DICE $\mathcal{N}(0, \Sigma)$	$\mathbf{1.79 \pm 0.97}$	2.40 ± 1.37	$\mathbf{3.86 \pm 4.18}^{\dagger}$	$\mathbf{1.13 \pm 0.54}$
	DPC-Net	1.15 ± 0.71	1.98 ± 1.39	$\mathbf{1.23 \pm 0.53}$	$\mathbf{0.89 \pm 0.33}$
	DICE $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	21.60 ± 13.85	0.96 ± 0.71	1.37 ± 0.66	1.15 ± 0.46
	D-DICE $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	$\mathbf{0.98 \pm 0.57}$	$\mathbf{0.67 \pm 0.35}$	1.50 ± 0.70	0.92 ± 0.35
Rotation (millideg/m)	DICE $\mathcal{N}(0, \Sigma)$	$21.66 \pm 13.13^{\dagger}$	$\mathbf{12.82 \pm 5.71}$	$15.35 \pm 32.06^{\dagger}$	9.22 ± 4.60
	D-DICE $\mathcal{N}(0, \Sigma)$	$\mathbf{8.20 \pm 5.31}$	12.88 ± 5.29	$\mathbf{14.09 \pm 28.83}^{\dagger}$	$\mathbf{8.86 \pm 4.41}$
	DPC-Net	4.31 ± 2.50	9.12 ± 4.35	$\mathbf{3.24 \pm 1.56}$	5.50 ± 3.04
	DICE $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	$161.49 \pm 21.35^{\dagger}$	4.31 ± 2.00	3.74 ± 1.70	5.57 ± 2.59
	D-DICE $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$	$\mathbf{3.37 \pm 2.00}$	$\mathbf{2.95 \pm 1.60}$	4.43 ± 2.08	$\mathbf{4.17 \pm 2.33}$

TABLE IX: Relative segment errors for 5 different estimators (mean error $\pm$ standard deviation, computed on all relative segments for each sequence). Network paired with $\mathcal{N}(0, \Sigma)$ does not use bias estimation to minimize the NLL. Oppositely, $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ results are obtained employing the loss in Eq. 8. Values tagged with $\dagger$ denote cases where the optimizer noticeably got stuck in a local minimum.
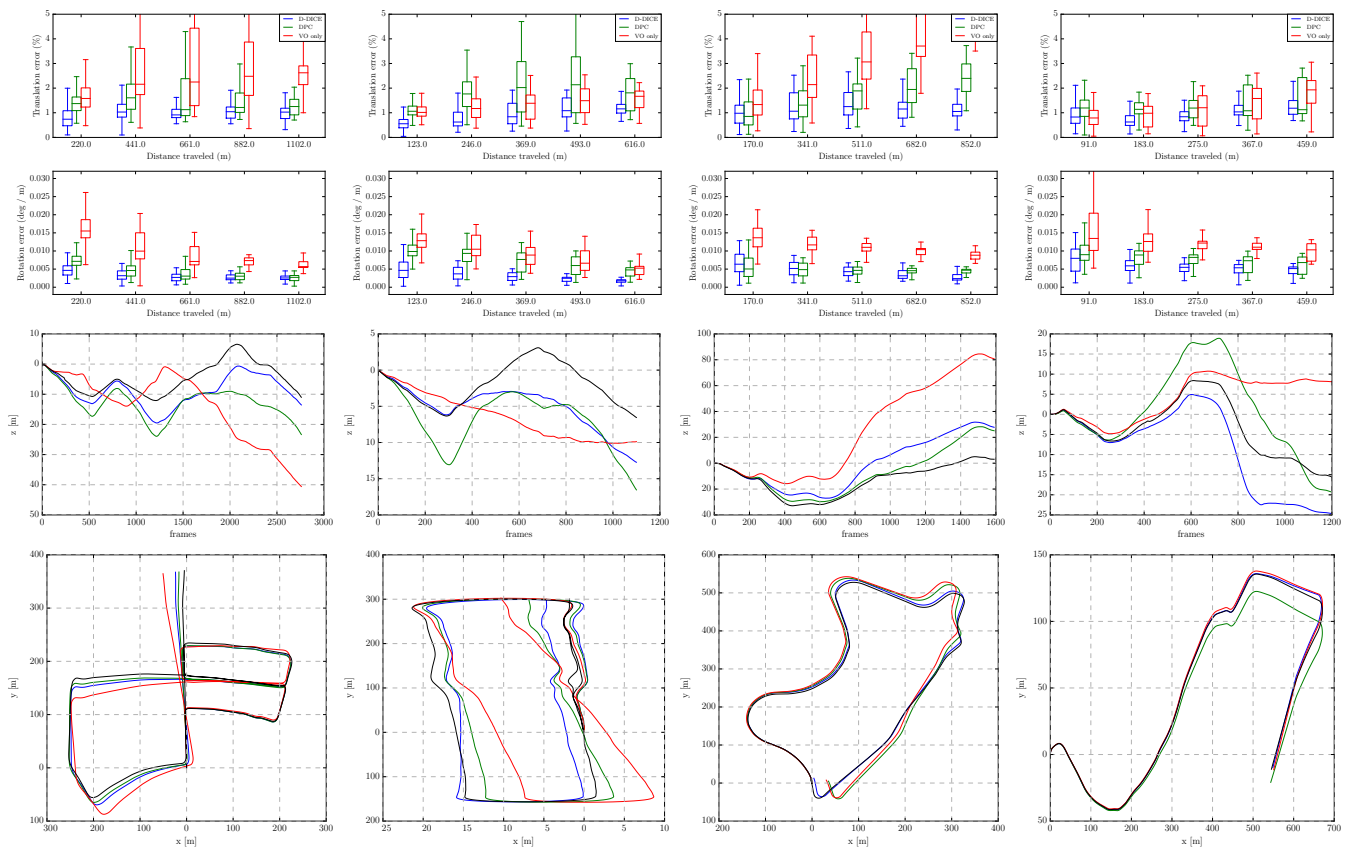

Fig. 3: Boxplots of the relative segment errors (top two rows), side view and top view of the trajectories (bottom two rows), for KITTI sequences 05, 06, 09, 10 (left to right). D-DICE (blue) and DPC-Net (green) corrections are used to reduce tracking errors using *libviso2* as the baseline estimator (red). The ground truth trajectory is in black.

the beneficial aspects of this approach compared to works that rely only on the estimation of the covariance matrix. Additionally, we have integrated visual odometry corrections with a more precise error model, inferred thanks to the assumption of biased distributions. To build on this matter, it can result interesting to adopt the same approach with dense estimators which have access to full disparity information. Finally, we plan to explore similar approaches with different perception processes that are yet to be associated with precise error models, *e.g.* iterative closest points algorithm based on LIDAR scans [18].
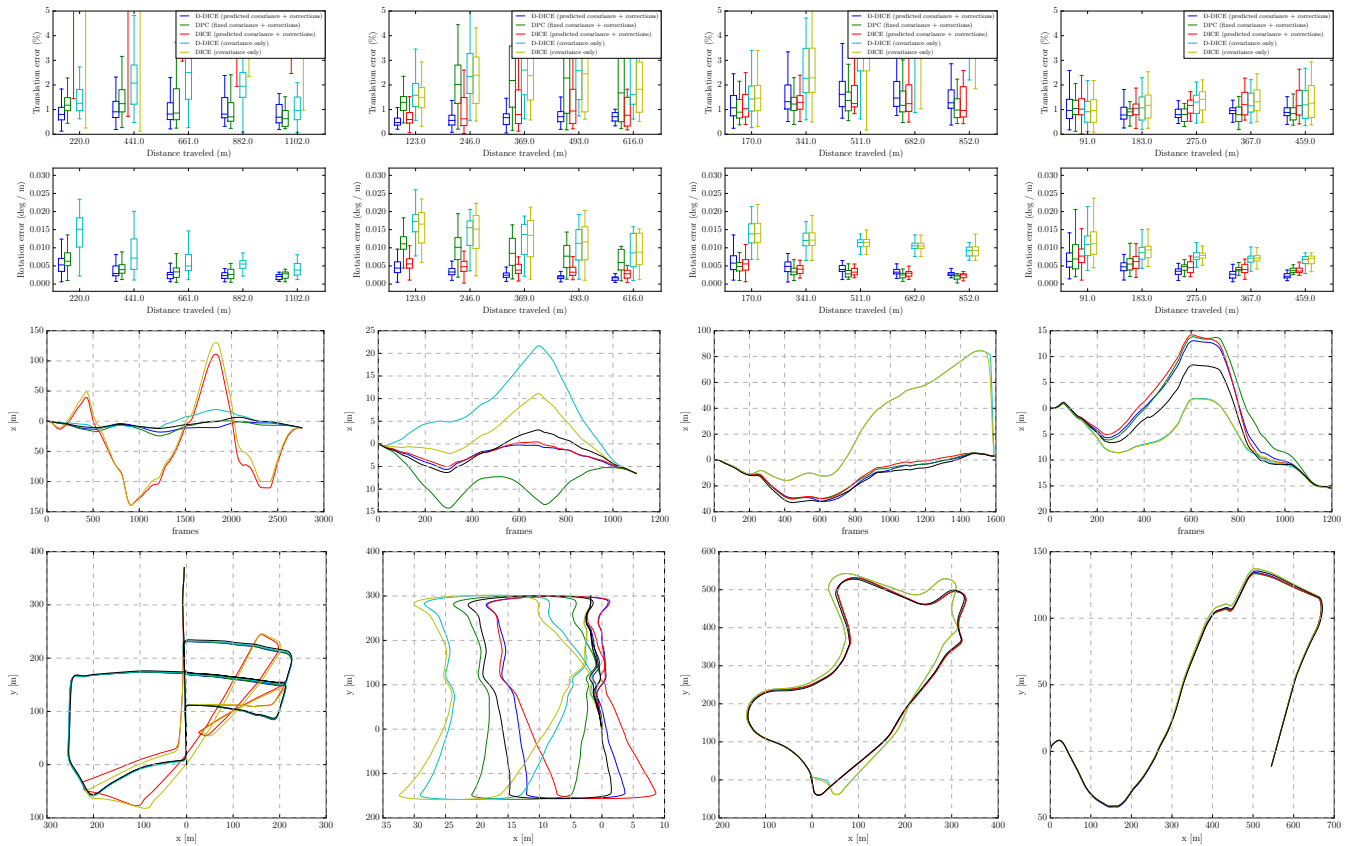
Fig. 4: Results after pose-graph minimization. Boxplots of the relative segment errors (top two rows), side view and top view of the trajectories (bottom two rows), for KITTI sequences `05`, `06`, `09`, `10` (left to right). D-DICE (blue), DPC-Net (green) and DICE (red) use corrections and uncertainty data (fixed variances for DPC-Net). Cyan and yellow are used for D-DICE and DICE versions that only learned error models. The ground truth trajectory is in black.

REFERENCES

[1] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[2] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[3] K. R. Konda and R. Memisevic, "Learning visual odometry with a convolutional network.," in *VISAPP (1)*, pp. 486–490, 2015.

[4] R. Li, S. Wang, Z. Long, and D. Gu, "Undeepvo: Monocular visual odometry through unsupervised deep learning," *arXiv preprint arXiv:1709.06841*, 2017.

[5] S. Wang, R. Clark, H. Wen, and N. Trigoni, "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 513–542, 2018.

[6] V. Peretroukhin and J. Kelly, "Dpc-net: Deep pose correction for visual localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2424–2431, 2017.

[7] G. Dubbelman, P. Hansen, and B. Browning, "Bias compensation in visual odometry," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2828–2835, IEEE, 2012.

[8] V. Peretroukhin, J. Kelly, and T. D. Barfoot, "Optimizing camera perspective for stereo visual odometry," in *2014 Canadian Conference on Computer and Robot Vision*, pp. 1–7, IEEE, 2014.

[9] K. Liu, K. Ok, W. Vega-Brown, and N. Roy, "Deep inference for covariance estimation: Learning gaussian noise models for state estimation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1436–1443, IEEE, 2018.

[10] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5974–5983, 2017.

[11] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *Advances in neural information processing systems*, pp. 5574–5584, 2017.

[12] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361, IEEE, 2012.

[13] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme," in *2010 ieee intelligent vehicles symposium*, pp. 486–492, IEEE, 2010.

[14] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[15] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of slam algorithms," *Autonomous Robots*, vol. 27, no. 4, p. 387, 2009.

[16] A. Handa, M. Bloesch, V. Pătrăucean, S. Stent, J. McCormac, and A. Davison, "gvnn: Neural network library for geometric computer vision," in *European Conference on Computer Vision*, pp. 67–82, Springer, 2016.

[17] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, IEEE, 2011.

[18] F. Pomerleau, F. Colas, R. Siegwart, *et al.*, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends® in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.