DiffPoseNet: Direct Differentiable Camera Pose Estimation

Chethan M. Parameshwara, Gokul Hari, Cornelia Fermüller, Nitin J. Sanket, Yiannis Aloimonos
Perception and Robotics Group, University of Maryland, College Park, MD, USA
{cmparam9,hgokul,fer,nitin,yiannis}@umiacs.umd.edu

Abstract

Current deep neural network approaches for camera pose estimation rely on scene structure for 3D motion estimation, but this decreases the robustness and thereby makes cross-dataset generalization difficult. In contrast, classical approaches to structure from motion estimate 3D motion utilizing optical flow and then compute depth. Their accuracy, however, depends strongly on the quality of the optical flow. To avoid this issue, direct methods have been proposed, which separate 3D motion from depth estimation, but compute 3D motion using only image gradients in the form of normal flow. In this paper, we introduce a network NFlowNet, for normal flow estimation which is used to enforce robust and direct constraints. In particular, normal flow is used to estimate relative camera pose based on the cheirality (depth positivity) constraint. We achieve this by formulating the optimization problem as a differentiable cheirality layer, which allows for end-to-end learning of camera pose. We perform extensive qualitative and quantitative evaluation of the proposed DiffPoseNet’s sensitivity to noise and its generalization across datasets. We compare our approach to existing state-of-the-art methods and show an improvement of up to $3.4 \times$ on KITTI, TartanAir, and TUM-RGBD datasets.

1. Introduction

The ability to localize is imperative for applications in mobile robotics, and solutions based on vision are often the preferred choice because of size, weight, power constraints and the availability of robust localization methods. Many mathematical frameworks and deep learning approaches have been developed for the problem of visual localization [12, 40] under the umbrella of Visual Odometry (VO) or Simultaneous Localization and Mapping (SLAM). However, their performance is subpar for commonly encountered challenging conditions in-the-wild that involve changing lighting, scenes with textureless regions, and dynamic objects.

Classical approaches [5, 8, 9, 25] for localization rely either on sparse feature correspondences between images or on the computation of dense motion fields (optical flow). One of the difficulties in optical flow estimation is bias due to noise [15, 16]. For example, if in a patch there are more gradients in one direction than another, their estimated optical flow will be biased towards the dominant direction. Even though over the past decade many learning-based approaches have proposed to improve optical flow estimation, this behavior still persists in optical flow approaches. This is demonstrated in Fig. 1, which shows the errors produced by the normal flow algorithm presented in this paper in comparison to three optical flow algorithms from the literature. As can be seen, all flow algorithms have large errors in regions of non-uniform gradient distributions.

To this end, the pioneers of the field remarked on the observation that the projection of optical flow on the image gradient direction is resilient to the bias and can be computed robustly. This projection is called the normal flow. Over the past few decades, a number of methods have been proposed that use the spatial gradient directly for 3D motion recovery. These methods are commonly called direct methods. In principle, such methods are robust and computationally cheaper than flow-based feature based approaches as they use the image brightness directly. However, despite the advantages of the normal flow formulation, the computational methods to estimate normal flow have not been robust enough to allow deployment in the wild. Thus, optical flow has been the go-to representation for ego-motion estimation, supported in recent years by the high accuracy and speed of deep learning algorithms [23, 27, 37]. To improve the robustness of camera pose estimation (ego-motion), we propose the first normal flow network NFlowNet.

Further, to estimate pose independent of scene structure from normal flow, direct approaches utilize minimal constraints. When optical flow or correspondence are available, pose is estimated using the depth-independent epipolar constraint. However different from these 2D measurements, normal flow is 1D and thus depth cannot be eliminated from the equations relating it to scene geometry and 3D motion.
Not making assumptions about the scene structure, the only constraint that can be imposed on the scene, is the depth positivity [35] or cheirality constraint [51]. Cheirality states that the scene has to be in front of the camera for it to be imaged, and thus the depth has to be positive. This constraint when enforced on normal flow can be utilized to estimate camera pose without making assumptions on scene depth or shape. Since the cheirality condition is an inequality constraint and hence is not differentiable, until recently it was not possible to employ it in a deep learning pipeline. To this end, we utilize the differentiable programming paradigm [2] implemented with the implicit differentiation [18] framework to reformulate the cheirality optimization into a differentiable layer and hence train our pose network in an end-to-end fashion.

In this work, we design a novel normal flow network \textit{NFlowNet} and couple it with a differentiable cheirality layer for robust pose estimation. Our contributions (in the order for ease of understanding) can be summarized as follows:

- We introduce a network \textit{NFlowNet} to estimate normal flow. This estimated robust normal flow, beyond this paper, is useful for applications requiring computationally efficient solutions for navigation tasks in computer vision and robotics.

- We formulate the estimation of pose from normal flow using the cheirality (or depth positivity) constraint as a differentiable optimization layer.

- Extensive qualitative and quantitative experimental results highlighting the robustness and cross-dataset generalizability of our approach without any fine-tuning and/or re-training.

2. Related Work

2.1. Normal Flow and Camera Pose

Several works have developed direct methods that use normal flow for pose estimation. The idea is that normal flow can be interpreted as “the projection of optical flow on the gradient direction.” Thus, given a normal flow vector, the optical flow is constrained to a half-plane [1]. If the 3D motion is only due to translation, this constrains the focus of expansion, i.e., the intersection of the translation axis with the image plane [35] to a half-plane. Based on this concept, [20] proposed different algorithms to solve for the case of translation only, and [42] analyzed the method’s stability in the presence of small rotations. Modeling the scene as piece-wise planar, [7] solved for 3D motion and calibration [6], and [24] added constraints for combining multiple flow fields. Not making depth assumptions, [14] developed constraints on the sign of normal flow, which geometrically separate the rotational and translational flow components and can be implemented as pattern matching. Others proposed techniques for separating 3D motion components by searching for lines in the image, where certain 3D motion components cancel out [41, 53]. Recently [4] modeled the cheirality constraint by approximating it with a smooth function, which allowed the use of modern optimization techniques. The method first solves for 3D motion from normal flow, and then refines using a regularization defined on depth. Experimental results demonstrate that the proposed pipeline outperforms other flow based approaches. Inspired by these findings, we follow a similar pipeline, but we develop the constraints within a neural network approach.
2.2. Learning-based Camera Pose Estimation

Early studies of learning-based camera pose (VO) models [46] [29] [28] were mainly focused on supervised learning approaches modelled as either absolute pose/relative pose regression problems. However, these methods require real-world ground truth poses which is often difficult to obtain. In order to alleviate the need of ground truth data, self-supervised VO was proposed. SiMLea[54]rns depth and pose simultaneously by minimizing photometric loss between warped and input image. [52] and [55] extend this idea to joint estimation of pose, depth and optical flow. Learning-based models suffer from generalization issues when tested on images from a new environment. Most of the VO models are trained and tested on the same dataset. Most recently, TartanVO [47] addresses generalization issues by incorporating the camera intrinsics directly into the model and training with a large amount of data. Recent advances in differentiable optimization layers (or differentiable programming) [2, 18] has enabled a new generation of generalizable pose learning approaches. [26] embeds an Epipolar geometric constraint into a self-supervised learning framework via bi-level optimization of camera pose and optical flow. BlindPnP [11] embeds geometric model fitting algorithms (PnP algorithm, RANSAC) into implicit differentiation layers. All the above work focus on reconstructing the structure (either through network or optimization) and/or optical flow correspondences for estimating camera motion. In our work, we address robust pose estimation by depending on structure-less cheirality constraints and normal flow. Finally, we utilize the concepts of Best of both worlds to enable speedup using data prior and novel data generalization from mathematical optimization.

3. Overview of Proposed Approach

The network architecture is illustrated in Fig. 2. It consists of NFlowNet, a network for estimating normal flow (Section 4), which then is used for self-adaptive camera pose estimation (or odometry estimation) (Section 5). The camera pose estimation proceeds in three steps. First we initialize the PoseNet using supervised training with successive images as input (Sec. 5.1). Next, pose is estimated using differentiable optimization by embedding a cheirality constraint defined on normal flow (Sec. 5.2). This way, similar to the classic SfM approach, pose is estimated independent of depth. In a last step, the pose in PoseNet is refined through a self-supervised refinement loss by using the pose estimates from the cheirality constraint to minimize the error in normal flow.

4. NFlowNet for Normal Flow Prediction

The first step in motion analysis from video is to compute an image motion representation. Most approaches either detect and track distinct features or compute gradient-based optical flow. The latter is estimated by assuming the intensity $I(x)$ (or a function of the intensity) at a point $x = [x, y]^T$ to remain constant over a short time interval $\delta t$. This is referred to the brightness constancy constraint [51]

$$I(x, y, t) = I(x + u\delta t, y + v\delta t, t + \delta t)$$ (1)

Here $u$ and $v$ refers to the image pixel motion. Approximating Eq.(1) with a first order Taylor expansion, we obtain Eq.(2)

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v = -\frac{\partial I}{\partial t}$$ (2)

We call the component of the flow along the gradient direction, the normal flow. Denoting the spatial gradients as $\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ and the flow as $u = [u, v]^T$, the normal flow vector $n$ (a 2d vector) is defined as:

$$n = \frac{u \cdot \nabla I}{\|\nabla I\|^2} \nabla I$$ (3)

Using the brightness constancy constraint (eq. 1), the normal can be computed directly from the spatial and temporal image derivatives, $I_t$, as $n = \frac{I_{tt}}{\|\nabla I\|^2} \nabla I$. Since this constraint alone is not sufficient to determine two-dimensional image motion, additional constraints need to be introduced. Traditionally, variational methods combining multiple global smoothness assumptions, have been the dominant approach for optical flow estimation, and more recently they have been replaced by deep learning algorithms. However, these methods tend to perform subpar on regions with little features or repeated texture due to the strong reliance on the training dataset, and in boundary regions due to oversmoothing, especially when the size of the flow varies significantly in the image. Computing normal flow solely on spatio-temporal derivatives, is unreliable and prone to errors. Hence, we propose a novel Normal flow network called NFlowNet.

We use an encoder-decoder convolutional neural network, which we train in a supervised way. Given an image pair, normal flow describes the pixel motion parallel to the image derivatives. To learn normal flow, we utilize the TartanAir dataset. Specifically, we utilize Eq. 3 to compute ground-truth normal flow. We train the NFlowNet supervised using the $l_2$ loss between our network predictions $\hat{n}$ and ground truth $\tilde{n}$, i.e.,

$$\argmin_{\tilde{n}} \|\tilde{n} - \hat{n}\|^2_2$$ (4)

In the experimental section (Sec. 7), we show that NFlowNet generalizes to the real-world and other datasets without any fine-tuning or re-training.
5. Self-Adaptive Pose Estimation from Normal Flow

We use a deep network to regress relative poses, that is, the 3D rigid motion of the camera between subsequent time steps and denoted as \( P_{t+1} \). The conversions between absolute poses and relative poses is explained next.

If the absolute pose at time \( t \) is given by \([T_t, R_t]^T\), where \( T_t \in \mathbb{R}^{3 \times 1} \) and \( R_t \in SO(3) \). The relative pose between \( t \) and \( t + 1 \) under a linear velocity assumption is denoted as \([V, \Omega]^T\) and is given by

\[
V = \frac{T_{t+1} - T_t}{dt}; \quad \Omega_x = \frac{\logm (R_t^R R_{t+1})}{dt} (5)
\]

Here, \( dt \) is the time increment between \( t \) and \( t + 1 \), \( \logm \) is the matrix logarithm operator and \( \Omega_x \) converts the vector \( \Omega \) into the corresponding skew symmetric matrix

\[
\Omega_x = \begin{bmatrix}
0 & -\Omega_z & \Omega_y \\
\Omega_z & 0 & -\Omega_x \\
-\Omega_y & \Omega_x & 0
\end{bmatrix} (6)
\]

5.1. PoseNet for Initializing Pose Estimation

In the first stage, we learn coarse relative poses using a CNN+LSTM. The feature representations learned by the CNN layers are passed to the LSTM for sequential modelling. We use a supervised \( l_2 \) loss between the ground truth \((\hat{P} = [\hat{V}, \hat{\Omega}]^T)\) and predicted poses \((\tilde{P} = [\tilde{V}, \tilde{\Omega}]^T)\). Here, \( \hat{V} \) and \( \hat{\Omega} \) represent the translational and rotational parts of the pose. The orientation is represented in \( X – Y – Z \) Euler Angles. Denoting as \( \lambda \) a weighting parameter, we solve the following optimization using backpropagation:

\[
\arg\min_{\tilde{P}} \left( ||\hat{V} - \tilde{V}||_2^2 + \lambda ||\hat{\Omega} - \tilde{\Omega}||_2^2 \right) (7)
\]

5.2. Differentiable Cheirality Layer for Fine Pose Estimation

To enable self-supervised learning of continuous pose (given a initialization value), we propose to utilize the cheirality constraint or depth positivity constraint, which states that all world points have to be in front of the camera, i.e., have positive depth. This condition has been classically used in structure from motion problems to disambiguate the physically possible camera poses from the set of computed solutions. The main reason for utilizing the cheirality rather than the acclaimed epipolar constraint or scene planarity constraint is due to the minimalism in the assumptions. Since in our formulation, depth positivity is enforced using the normal flow, which in-turn makes minimal assumptions about the scene structure, our formulation generalizes to novel scenes with remarkable accuracy (Sec. 6.2.2).

Let us mathematically define the constraints. Denoting the magnitude of the normal flow (a scalar) at pixel \( x \) as \( n_x \) and the direction of the image gradient as \( g_x \) (a unit vector), we have

\[
n_x = ||n||_2 = \frac{1}{Z_x} (g_x \cdot A) V + (g_x \cdot B) \Omega, (8)
\]

where, \( V \) denotes the constant translational velocity and \( \Omega \) the rotational velocity \( \Omega \) in a small time instant, and \( Z_x \) is the depth of the point under consideration. Intuitively, \( V, \Omega \) warp the flow field and \( Z_x \) scales the flow field, and the matrices \( A \) and \( B \) determine how the motion flow field is projected onto the image plane due to translational and rotational velocities respectively and are given by
\[
A = \begin{bmatrix}
-1 & 0 & x \\
0 & -1 & y
\end{bmatrix}
\]

(9)

\[
B = \begin{bmatrix}
x y \\
(y^2 + 1) & -(x^2 + 1) & y \\
-xy & -x
\end{bmatrix}
\]

(10)

Let us consider in eq. (8) the two components of normal flow: the translational component, which depends on depth, and the rotational component, which is independent of depth. If we subtract the rotational component from both sides, we obtain

\[
n_x - (g_x \cdot B) \Omega = \frac{1}{Z_x} (g_x \cdot A) V.
\]

(11)

We can enforce that the left hand side (the derotated normal flow) and the right hand side (the translational component) must have same sign. Since the depth \((Z_x)\) is positive, the following product, which we denote as \(\rho_x(V, \Omega)\), is positive, i.e.,

\[
\rho_x(V, \Omega) = ((g_x \cdot A) V) \cdot (n_x - (g_x \cdot B) \Omega) > 0
\]

(12)

To arrive at an objective function to be used in an optimization, we can model the cheirality constraint by passing \(\rho_x\) through a smooth function, such as the ReLU function [4]. Since, the deep learning pipeline requires the function to be twice differentiable, we choose the GELU function [4]. Since, the deep learning pipeline requires the Cheirality layer and we denote this refined pose as \(\hat{P}_r = \begin{bmatrix} \hat{V}_r & \hat{\Omega}_r \end{bmatrix} \). Now we use \(\hat{P}_r\) to refine our PoseNet’s coarse pose to obtain a more accurate prediction of pose.

The final self-adaptive pose estimation is performed as a bi-level minimization in the network [18], in which an upper-level problem is solved subject to constraints imposed by a lower-level problem and is formally defined next.

\[
\arg\min_{\hat{P}_c} \mathbb{E} \left( \mathcal{R}(\rho_x(V, \Omega)) \right)
\]

subject to \(\arg\min_{\hat{P}_r} \mathbb{E} \left( \mathcal{R}(\rho_x(V, \Omega)) \right)\)

(14)

The lower-level problem (second row) of Eq. (14) enforces the cheirality constraint to obtain pose \(\hat{P}_r\), which then is used to compute the normal flow error in the upper-level loss function (first row). The upper-level loss enforces the consistency between the normal flow from NFlowNet and that computed using the motion parameters \(\hat{P}_c\) with the implicit depth term expressed by the motion parameters \(\hat{P}_r\).

In practice, we back-propagate through the upper-level to refine poses \(\hat{P}_c\) using supervision from \(\hat{P}_r\) through the lower-level. Using implicit differentiation, all that is computed from the lower layer is the gradient, and this step is agnostic to the optimizer used. Specifically, we derive \(\frac{\partial \hat{P}_c}{\partial \hat{P}_r}\), which is computed from the product of second order derivatives. (The interested reader is referred to [18], eq. (15) for details.) It is important to note that we rely on the generalizability of NFlowNet, hence it is not fine-tuned.

6. Experiments

6.1. Implementation Details

6.1.1 Datasets

We use eight environments from the TartanAir [48] dataset (amusement, oldtown, neighborhood, soulcity, japanesealley, office, office2, seasedtown) for training and two environments (abandonedfactory and hospital) for testing our NFlowNet network. For odometry evaluation, we use the Tartan challenge test data [48]. We also conduct extensive experiments on the KITTI Odometry [17] and the TUM-RGBD [44] datasets to evaluate the robustness and generalization performance of our proposed system.

6.1.2 Networks and Optimization Layer

For the NFlowNet we use an encoder-decoder architecture based on EVPropNet [39] to directly regress sparse normal flow. The encoder contains residual blocks with convolutional layers and the decoder contains residual blocks with transpose convolutional layers. We choose the number of residual and transposed residual blocks as 2 and the expansion factor (factor by which the number of neurons are increased after every block) as 2. We backpropagate the gradients using a mean squared loss computed between...
groundtruth and predicted normal flow as given in Eq. 4. We used the Adam optimizer to train our network with a learning rate of $10^{-4}$ and batch size of 8 for 400 epochs.

Our PoseNet architecture is inspired from [46] and uses the VGG-16 encoder for the CNN stage [13] and two LSTM layers each with 250 hidden units for the recurrent layer-stage. We initially train this model with a subset of the TartanAir data for 30 epochs to obtain a coarse estimate to initialise the Cheirality Layer. We use the Adam optimizer and set a fixed learning rate of $10^{-5}$. We consider sequences of six consecutive image frames and a batch size of eight while training. During test time, we only use two consecutive image frames to estimate the relative camera pose between images $I_t$ and $I_{t+1}$.

For optimization layer, We use the L-BFGS [10] solver. The line search function was set to strong Wolfe [50], the number of iterations were set to 100 and the gradient norms were clipped to 100. We initialised the optimizer with the predictions from PoseNet. Our overall system is implemented in Python 3.7 and PyTorch 1.9.

6.1.3 Training and Testing Procedure

The whole training schedule consists of three stages. First, we only train NFlowNet and PoseNet in a supervised manner using the training strategies mentioned above. Then, we freeze NFlowNet and jointly train the PoseNet with the cheirality layer in an self-supervised fashion via the refinement loss. The self-supervised training is carried out for 120 epochs using four Nvidia P6000 GPUs. For cheirality layer, the stopping criteria is when the objective function is below $10^{-20}$ or the number of iterations exceeds 300. During testing, our final pose predictions are obtained by passing PoseNet priors through the cheirality layer along with the predictions from NFlowNet. Due to our self-supervised refinement training, the prior PoseNet estimates help in faster convergence of the cheirality optimization process (takes less than 5 iterations). We will release codes for inferencing NFlowNet and our differentiable chierality layer upon the acceptance of the paper.

6.1.4 Evaluation Metrics

To evaluate our NFlowNet and to compare against other optical flow networks, we project the optical flow obtained from the state-of-the-art approaches on the gradient direction, and we measure the pixel error. As normal flow is defined only along the gradient direction, we utilize the Projection Endpoint Error (PEE), an analog to the Average Endpoint Error (AEE) error metric proposed in [19]. The PEE between the projected optical flow ($\tilde{u}$) and the groundtruth ($\bar{u}$) is defined as:

$$\text{PEE} = \|\tilde{u} - \frac{\nabla I}{||\nabla I ||_2} \cdot \bar{u}\|$$

To evaluate the regressed relative poses from our model, we use two metrics:

**Absolute Trajectory Error:** [36] evaluates the global consistency by comparing the absolute poses between the estimated and the ground truth trajectory. As the predicted and groundtruth trajectories can be specified in arbitrary coordinate frames, they are aligned using [21], which finds the rigid-body transformation $S$. The error matrix between poses, $E_t$ is defined as:

$$E_t = Q_t^{-1}SP_t$$

Here, $P_t \in SE(3)$ is predicted pose at time $t$ from the sequence of the spatial poses, $Q_t \in SE(3)$ is pose at time $t$ from the ground truth trajectory, and $E_t$ is the error matrix between these poses. Then, ATE is defined as the root mean square error of the error matrices:

$$ATE = \left( \frac{1}{n} \sum_{t=1}^{n} \| \text{trans}(E_t) \|^2 \right)^{\frac{1}{2}}$$

**Relative Pose Error:** measures the local trajectory accuracy over a fixed time interval $\Delta t$. From a sequence of $n$ poses, we obtain $m = n - \Delta t$ relative pose error matrices $F_t^{\Delta t} = (Q_t^{-1}Q_{t+\Delta t})^{-1}(P_t^{-1}P_{t+\Delta t})$

$$t_{rel} = \text{RPE}_{trans}^{\Delta t} = \left( \frac{1}{m} \sum_{t=1}^{m} \| \text{trans}(F_t^{\Delta t}) \|^2 \right)^{\frac{1}{2}}$$

$$r_{rel} = \text{RPE}_{rot}^{\Delta t} = \frac{1}{m} \sum_{t=1}^{m} \angle \text{rot}(F_t^{\Delta t}),$$

where trans($) and rot($) denote the translation and rotation components of the pose matrix. $\angle R = \arccos \left( \frac{w(R) - 1}{2} \right)$ with $R$ denoting the rotation matrix.

6.2. Analysis of Experimental Results

6.2.1 Accuracy of Normal Flow

In the first case study, we quantitatively evaluate NFlowNet. We compare our network with optical flow methods of various flavours: (a) supervised (PWC-Net [45], LiteFlowNet [22]), (b) self-supervised (SelfFlow [31]).

In Table 1, we present a quantitative evaluation of normal flow. We trained our NFlowNet and fine-tuned optical flow networks with TartanAir (8 environments). We demonstrate the PEE error on the first four sequences of the environments abandonedfactory and hospital. NFlowNet performs better than the optical flow networks.
by up to 6×. By learning normal flow, we constrain the network to focus on prominent features (edges, textures) rather than dense correspondences in textureless regions. Through this “attention-like” mechanism, NFlowNet performs better than its optical flow counterparts with up to 46× smaller model size.

### 6.2.2 Comparison of Pose Estimation

In this section we compare our DiffPoseNet framework with various state-of-the-art camera pose estimation approaches. These approaches can be broadly classified into: (a) pure deep learning models, (b) pure geometric constraint based models and (c) hybrid deep learning models that incorporate some form of geometric constraints in the learning pipeline.

We present the Absolute Trajectory Error (ATE) of our model on the TartanAir challenge data in the MH000-007 sequences of TartanAir [48] dataset.

Table 2. ATE (m) ↓ on the MH sequences of TartanAir [48] dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>000</th>
<th>001</th>
<th>002</th>
<th>003</th>
<th>004</th>
<th>005</th>
<th>006</th>
<th>007</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORB-SLAM [33]</td>
<td>1.30</td>
<td>0.04</td>
<td>2.37</td>
<td>2.45</td>
<td>5.04</td>
<td>3.76</td>
<td>4.04</td>
<td>1.73</td>
</tr>
<tr>
<td>TartanVO [47]</td>
<td>4.88</td>
<td>0.26</td>
<td>2.00</td>
<td>0.94</td>
<td>1.07</td>
<td>3.19</td>
<td>1.00</td>
<td>2.04</td>
</tr>
<tr>
<td>Ours</td>
<td>2.56</td>
<td>0.31</td>
<td>1.57</td>
<td>0.72</td>
<td>0.82</td>
<td>1.85</td>
<td>1.32</td>
<td>1.24</td>
</tr>
</tbody>
</table>

Table 3. ATE (m) ↓ on the TUM-RGBD [44] benchmark.

<table>
<thead>
<tr>
<th>Method</th>
<th>000</th>
<th>001</th>
<th>002</th>
<th>003</th>
<th>004</th>
<th>005</th>
<th>006</th>
<th>007</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORB-SLAM2 [34]</td>
<td>-</td>
<td>0.016</td>
<td>0.078</td>
<td>0.055</td>
<td>0.052</td>
<td>0.054</td>
<td>0.004</td>
<td></td>
</tr>
<tr>
<td>DeepTAM [56]</td>
<td>0.116</td>
<td>0.078</td>
<td>0.055</td>
<td>0.052</td>
<td>0.054</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TartanVO [47]</td>
<td>0.178</td>
<td>0.125</td>
<td>0.122</td>
<td>0.049</td>
<td>0.062</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td>0.121</td>
<td>0.101</td>
<td>0.083</td>
<td>0.056</td>
<td>0.048</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Relative Pose Error ($t_{rel}$ and $r_{rel}$) ↓ results of various Pose estimation methods on KITTI [17] dataset. Note that, bold represents the best result and underline represents the second best.

<table>
<thead>
<tr>
<th>Method</th>
<th>06</th>
<th>07</th>
<th>09</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepVO [46]</td>
<td>5.42</td>
<td>5.82</td>
<td>1.91</td>
<td>4.60</td>
</tr>
<tr>
<td>Wang et al. [49]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UnDeepVO [30]</td>
<td>6.20</td>
<td>1.98</td>
<td>3.15</td>
<td>2.48</td>
</tr>
<tr>
<td>GeoNet [52]</td>
<td>9.28</td>
<td>4.34</td>
<td>8.27</td>
<td>5.93</td>
</tr>
<tr>
<td>TartanVO [47]</td>
<td>4.72</td>
<td>2.95</td>
<td>4.32</td>
<td>3.41</td>
</tr>
<tr>
<td>BiLevelOpt [26]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4.36</td>
</tr>
<tr>
<td>ORB-SLAM [33]</td>
<td>18.68</td>
<td>10.69</td>
<td>0.37</td>
<td>15.3</td>
</tr>
<tr>
<td>VISO2-M [43]</td>
<td>7.34</td>
<td>6.14</td>
<td>23.61</td>
<td>19.11</td>
</tr>
<tr>
<td>Ours</td>
<td>2.93</td>
<td>1.70</td>
<td>4.06</td>
<td>2.35</td>
</tr>
</tbody>
</table>

6.2.3 Robustness of Pose Estimation

For most camera pose estimation approaches, the performance of the algorithm is also governed by external factors such as lighting, weather, and sensor noise. These external factors often lead to errors in motion fields and cause pose estimation to fail or diverge. In this case study we present a robustness analysis by injecting noise into the motion fields.

**Note:** Table 1 presents the Projection Endpoint Error (PEE) of different state-of-the-art methods as compared to our NFlowNet.
Figure 3. Qualitative comparison of trajectories between our DiffPoseNet and other state-of-the-art approaches on the KITTI dataset.

Figure 4. Comparison between our model and pure learning based VO on relative rotation error (in °/frame) from KITTI - 07.

Figure 5. Robustness evaluation of normal flow representation for pose estimation on KITTI-10.

We study the robustness of the normal flow based cheirality layer against the epipolar layer [26]. We artificially inject errors in the normal flow and optical flow and evaluate our framework’s performance under these conditions. The error is modelled as additive uniform noise $U(\epsilon)$, where $[-\epsilon, \epsilon]$ is the bound on noise. We induce this noise to both normal flow and optical flow only on the gradient direction where normal flow is well-defined to make the comparison fair.

Fig. 5 shows the relative pose errors, $t_{rel}$ and $r_{rel}$, over $\epsilon$ values $\{0, 5, 10, 15, 20\}$%. Here, BiLevelOpt refers to the pose estimation using the epipolar constraint layer [26] with optical flow as input. “Cheirality-OF” refers to the pose estimation via the cheirality layer using normal flow obtained by projecting SelFlow optical flow predictions on the gradient directions (we choose SelFlow as it has the best performance among all optical flow methods used in this paper).

Observe that, our DiffPoseNet is more resilient to noise and fails “more gracefully” compared to other methods. We owe this robustness to the lack of strong constraints used in our approach as compared to other methods that either rely on strong features or strong photometric consistency. We believe that carefully crafted optimization problems can lead to robust pose estimation neural networks that generalize well to novel datasets while being robust to noise, a capability rarely seen in most state-of-the-art methods [3, 38].

7. Conclusion

In this work, we combined the best of both worlds from classical direct camera pose estimation approaches and deep learning taking advantage of differentiable programming concepts. Specifically, we addressed the problem of estimating relative camera pose using a sequence of images. To achieve this, we introduced the DiffPoseNet framework. As part of this framework, we introduced a normal flow network called NFlowNet, that predicts accurate motion fields under challenging scenarios and is more resilient to noise and bias. Furthermore, we proposed a differentiable cheirality layer that when coupled with NFlowNet can estimate robust and accurate relative camera poses.

A comprehensive qualitative and quantitative evaluation was provided on the challenging datasets: TartanAir, TUM-RGBD and KITTI. We demonstrated the efficacy, accuracy and robustness of our method under noisy scenarios and cross-dataset generalization without any fine-tuning and/or re-training. Our approach outperforms the previous state-of-the-art approach. Particularly, NFlowNet can output accurate up to 6× motion fields with up to 46× smaller model size. We believe this will open a new direction for camera pose estimation problems.
References


