Geometric Consistency for Self-Supervised End-to-End Visual Odometry

Ganesh Iyer*[1], J. Krishna Murthy*[2]*, Gunshi Gupta[1], K. Madhava Krishna[1], Liam Paull[2]
[1] International Institute of Information Technology Hyderabad (India)
[2] Montreal Institute of Learning Algorithms (MILA), Universite de Montreal
{giyer2309, krrish94, gunshigupta9}@gmail.com

Abstract

With the success of deep learning based approaches in tackling challenging problems in computer vision, a wide range of deep architectures have recently been proposed for the task of visual odometry (VO) estimation. Most of these proposed solutions rely on supervision, which requires the acquisition of precise ground-truth camera pose information, collected using expensive motion capture systems or high-precision IMU/GPS sensor rigs. In this work, we propose an unsupervised paradigm for deep visual odometry learning. We show that using a noisy teacher, *which could be a standard VO pipeline, and by designing a loss term that enforces geometric consistency of the trajectory, we can train* accurate *deep models for VO that do not require ground-truth labels. We leverage geometry as a self-supervisory signal and propose "Composite Transformation Constraints (CTCs)", that automatically generate supervisory signals for training and enforce geometric consistency in the VO estimate. We also present a method of characterizing the uncertainty in VO estimates thus obtained. To evaluate our VO pipeline, we present exhaustive ablation studies that demonstrate the efficacy of end-to-end, self-supervised methodologies to train deep models for monocular VO. We show that leveraging concepts from geometry and incorporating them into the training of a recurrent neural network results in performance competitive to supervised deep VO methods.*

1. Introduction

Visual odometry (VO) is the process of estimating the ego-motion of a camera solely from a sequence of images it captures. This capability forms the backbone of any system that requires visual localization. Most solutions to the problems of visual odometry estimation and simultaneous localization and mapping (simultaneously estimating camera trajectory and building a representation of the world) rely on the use of feature matching/tracking or geometric methods in combination with keyframe-based optimization or bundle adjustment [4, 17]. One major challenge of such approaches is to design visual features that have good invariance properties and can be reliably associated. In contrast, deep learning methods *learn* feature representations instead of handcrafting them. Consequently, they have been applied to the problem of visual place recognition for SLAM (discovering that we are in a previously visited place) [16] as well as VO [1, 14, 15, 21, 24, 25, 26, 26].

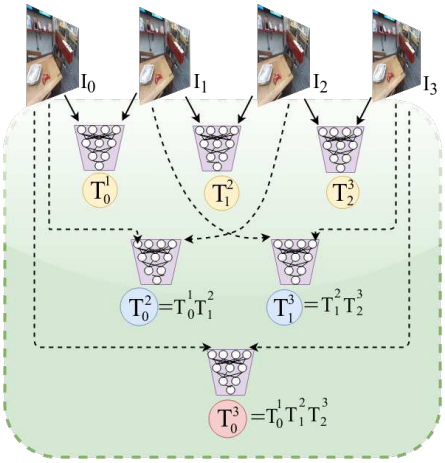

Figure 1. **System overview**: We leverage the observation that compounded sequences of transformations over short timescales should be equivalent to single transformations over longer timescales. This allows us to create additional constraints, that we refer to as "Composite Transformation Constraints", that can be used as consistency enforcers and aid in training deep architectures for VO without requiring ground-truth labels.

Most learning-based approaches to VO fall into one of the following categories:

- Supervised deep VO approaches assume the availability of ground-truth information in the form of per-frame camera pose in a global frame, usually gathered using a motion-capture system or expensive IMU/GPS sensor rigs [1, 12, 18, 21, 24, 25].
- Unsupervised deep VO approaches do not require

*The first two authors contributed equally to this work. We thank NVIDIA for donating a DGX-1 computer used in this work. This research was enabled in part by support provided by Compute Canada www.computecanada.ca

ground-truth pose information, but leverage some alternate visual information that can assist the learning process, such as depth [8, 15], stereo images [7, 26], or optical flow [14].

Most state-of-the-art deep approaches to VO employ sequence-models, such as long-short term memory (LSTM) units [9], to capture long term dependencies in camera motion [1, 15, 24, 25]. These models have been shown to correct drift in the estimated trajectory that may have been caused due to incorrect odometry estimates for a few frames in the sequence. However, existing approaches (that do not use depth information) lack tight consistency constraints across time steps. They rely solely on the statefulness of the LSTM model to bring about a weak *smoothing* effect.

We propose an unsupervised training scheme through our proposed model, CTCNet, for the task of learning VO estimation. We tackle the problem in a setting that does not assume the availability of ground-truth odometry data. To this end, we use *noisy* odometry estimates from a conventional VO pipeline (ORB-SLAM [17]) to train a recurrent architecture that outputs the relative camera pose transformation between frames. To compensate for noisy estimates used in training, we leverage geometry as a self-supervisory signal, and define a set of *Composite Transformation Constraints (CTCs)* across a series of image frames. These constraints arise naturally from the composition law for rigid-body transformations. Estimated transforms over short timescales, when compounded, must equal their counterparts that are computed (independently) over longer timescales. Fig. 1 shows an example of CTCs applied to an input image sequence comprising four frames. One such constraint here is that compositions of relative transforms between successive frames should equal the transform between the first and the fourth frames. For this to be meaningful however, we require that the longer timescale estimate (i.e., between the first and the fourth frames here) be computed independently.

In contrast to other works that estimate poses using deep learning [12, 18, 25], our network directly regresses to $\mathfrak{se}(3)$ exponential coordinates, and our loss function is formulated as an $L2$-norm over the coordinates. Furthermore, we also describe covariance recovery for VO estimates from our pipeline, using dropout [20] to perform approximate Bayesian inference [11].

Our experiments on the 7-Scenes [6] dataset demonstrate comparable, and in some cases, better performance compared with supervised methods. We also evaluate several variants of the proposed architecture and demonstrate the flexibility of this training process. To the best of our knowledge, this is the first approach to unsupervised VO estimation that does not require depth prediction as an auxiliary task, as is usually the case [15, 23, 26].

2. Related Work

Deep learning solutions for VO are a relatively recent but quickly evolving subset of methods for estimating camera ego-motion. While initial approaches relied on ground truth poses for training, recent approaches also explore the possibility of unsupervised training schemes.

2.1. Supervised Approaches

Numerous approaches [1, 14, 21, 24, 25] learn the task of VO estimation using ground-truth data available in the form of global-camera poses, recorded by high-precision GPU+IMS rigs.

Konda *et.al.* [14] first proposed an autoencoder to learn a latent representation of the optical flow between camera frames jointly with the ego-motion estimation task. Kendall *et.al.* [12] proposed a convolutional network based on the GoogLeNet architecture for monocular camera relocalization. Wang *et.al.* [25] further extend the idea to exploit long term dependencies between monocular frames through a recurrent convolutional network.

Clark *et.al.* [1], assimilate pose information over windows of sequential frames and their corresponding inertial information using an $SE(3)$ concatenation layer and separately fuse visual and inertial streams to provide robust trajectory estimates. Ummenhofer *et.al.* [21] propose 'DeMoN' for supervised joint estimation of depth, ego-motion, surface normals and optical flow given two successive views. They show that learning these multiple-tasks jointly leads to better performance on each of the tasks compared to scenarios where each task was learnt in a disjoint fashion.

Peretroukhin *et.al.* [18] recently propose a different approach to supervised VO. Rather than predicting relative transformations between pairs of frames, they train a CNN that *corrects* estimates from an existing VO framework. They use stereo pairs for training and rely on pose graph relaxation to correct existing pose values obtained from SVO [4].

However, the training of these networks is supervised against ground truth and is therefore limited by the availability of such recorded ground truth information.

2.2. Unsupervised Approaches

Recently, a lot of work has been conducted towards the estimation of depth in a scene, which can be used as a prior to find relative camera pose between associated successive image frames. *Handa et.al.* [8] in their library gvnn, introduced the 3D spatial transformer. Operating on a depth map along with the corresponding image, it finds the $\mathfrak{se}(3)$ warp $\boldsymbol{\xi}$ that transforms the camera coordinates of the current frame to those of the next frame, such that when projected back into the image space of the next frame, the photometric error between the resulting $SE(3)$ warped image and the actual next image is minimized. This work paved the

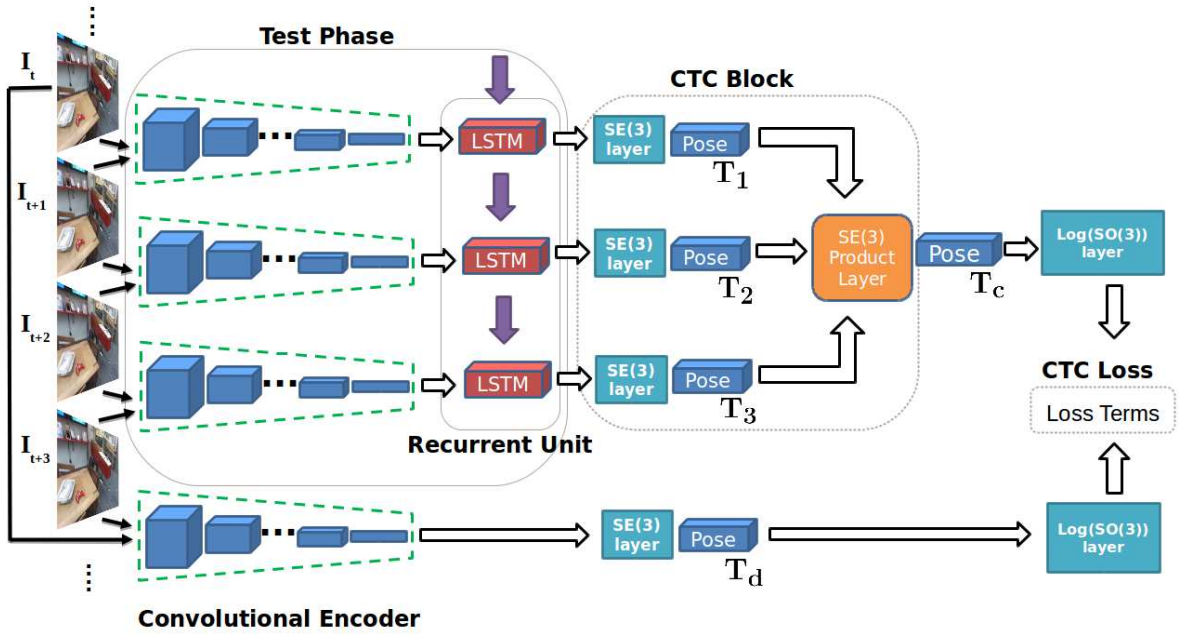

Figure 2. **End-to-end architecture:** An example of Composite Transformation Constraints (CTCs) being applied to 4 successive input images. During training, two estimates are generated from the inputs: one for a sequential pairwise constraint and one for a CTC constraint. At test time, each frame is only fed into the network once to receive the output pose from the $SE(3)$ layer. Therefore, the system can run in an online fashion and in real-time on a GPU. (In practice, when training, we use up to 18 frames in an input window and hence generate multiple CTCs that are applied to frames in the window. Here we show only one CTC block to avoid clutter.)

way for self-supervised methods that don't require ground truth pose information. Another work along similar lines by Zhou *et.al.* [26] learns both depth and pose from monocular frames, using a novel depth-based pipeline for reconstructing successive frames, although it is unable to recover depth in metric scale. Vijayanarasimhan *et.al.* [23] further propose 'SfM-Net' to jointly predict depth, segmentation, optical flow, camera and rigid object motion. They propose both unsupervised and supervised variants based on the availability of ground truth ego-motion or depth.

In a more recent work, Li *et. al.* [15] use stereo and monocular geometric constraints to create a composite loss function during training and use only monocular frames for testing. In contrast, we use an LSTM based architecture that exploits mutliple-views along with their associated pose consistency constraints, while still using frames from a single camera.

Furthermore, all these unsupervised approaches use depth prediction as a convenient auxiliary task to aid in learning. Our approach is orthogonal to these, in the sense that we rely purely on geometric consistency and do not need such auxiliary tasks for unsupervised learning of VO.

3. Learning VO without ground-truth labels

The central idea of this paper is to leverage geometric consistency and use it as a proxy for ground-truth labels. In this section, we describe composite transformation constraints in detail and present our network architecture, loss function, and training details. We also briefly describe how covariance recovery can be easily incorporated into the proposed approach, without additional training overhead.

3.1. Composite Transformation Constraints

Composite transformation constraints are based on the fundamental law of composition of rigid-body transformations. Simply put, if we have transformations between two sets of frames $A \mapsto B$ and $B \mapsto C$, then the transform from $A \mapsto C$ is simply the concatenation of the two former transforms. As a toy example (Fig. 2), given a sequential set of frames $\mathcal{F} = (I_t, I_{t+1}, I_{t+2}, I_{t+3})$ at time t, we train a neural network to predict the transforms: $[T_t^{t+1}, T_{t+1}^{t+2}, T_{t+2}^{t+3}]$. Since we do not have access to ground-truth labels, we cannot quantitatively evaluate the accuracy of the predicted transforms. However, for geometrical consistency to hold, we know that the following composite transformation constraints must be satisfied.

$$\begin{aligned} T_t^{t+1} \cdot T_{t+1}^{t+2} \cdot T_{t+2}^{t+3} &= T_t^{t+3} \\ T_t^{t+1} \cdot T_{t+1}^{t+2} &= T_t^{t+2} \\ T_{t+1}^{t+2} \cdot T_{t+2}^{t+3} &= T_{t+1}^{t+3} \end{aligned} \tag{1}$$

The extent to which the above constraints are satisfied

is a measure of trajectory consistency. We have a convolutional encoder that feeds into a recurrent neural network as our deep architecture for VO estimation (details in Sec 3.2, see Fig. 2). We first feed all frames in $\mathcal{F}$ into this network and estimate all successive transformations of the form T_i^{i+1}. This provides us with all the information required to evaluate the left-hand sides of the above constraints. To evaluate the right-hand sides, we estimate all T_j^is $(j \neq i)$ using only the convolutional encoder and feeding it frames I_i and I_j.

As an example, for an image pair (I_t, I_{t+2}), the predicted transform T_t^{t+2} must be equal to the product of transforms T_t^{t+1} and T_{t+1}^{t+2}, predicted sequentially for frames (I_t, I_{t+1}, I_{t+2}). For larger input sequences, we can naturally formulate many more such CTCs. All of them are jointly optimized during the training phase.

Note that, although traditional LSTM-based architectures (without CTCs) would suffice to provide smooth trajectories by mitigating noise between intermediate transforms (smooths them out so that they do not deviate much from the neighboring odometry estimates), it does not ensure geometric consistency of the obtained estimates. The composite transformation constraint is, therefore, essential in bringing about consistency in the predicted sequential transforms, such that the LSTM not only provides smooth trajectory estimates but also estimates that are consistent within the underlying geometry of the trajectory.

3.2. Network Architecture

Our network consists of three major components - a convolutional encoder, a recurrent unit, and a CTC block. Fig. 2 illustrates the proposed end-to-end architecture for unsupervised VO.

3.2.1 Convolutional encoder

Our network follows a similar structure to FlowNetSimple and VGG-11 [2, 19]. The network takes as input a pair of RGB images, denoted I_t and I_{t+1}, stacked along their color channels. We initialize our convolutional layers with the pre-trained weights from VGG-11[1]. Unlike VGG-11, our input consists of two images stacked together as opposed to a single image, we replicate and concatenate the weight tensors from VGG-11 to initialize our first layer. The pre-trained weights provide a good initial point for learning feature extraction. After this initial series of convolutions and pooling, features are globally aggregated using a series of strided convolutional layers. During the training process, we continue fine-tuning our weights for the task of estimating $\mathfrak{se}(3)$ transformation parameters.

The output of our network is a $C-$dimensional vector, $\mathcal{V}$. This vector $\mathcal{V}$ is provided as input to a fully connected layer

[1]We use a slightly different variant from the one in the original paper [19]. Our variants use BatchNorm [10] before every nonlinearity.

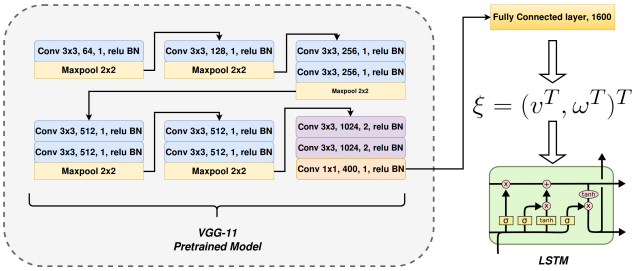

Figure 3. **Architectural specifics**: Our network builds on the popular VGG-11 network and takes images I_t and I_{t+1} that are resized to 320x240 and then stacked along the RGB channels. Each convolution layer is followed by a ReLU non-linearity. Then, batch normalization and max-pooling are successively applied. Finally 2 layers of strided convolution are applied followed by a 1x1 convolution layer to produce a latent vector of length 1080 that is used as an input to the LSTM unit. The LSTM has 1000 units in its hidden state. A final fully-connected layer maps the output of the LSTM to a 6-dimensional $\mathfrak{se}(3)$ coordinate vector.

that regresses a 6-vector comprising transformation parameters $\xi = (v^T, \omega^T)^T \in \mathfrak{se}(3)$ where v is the translational velocity, and ω is the rotational velocity respectively.

3.2.2 Recurrent unit

The vector $\mathcal{V}$ from the convolutional encoder is also input to a recurrent unit that maintains a hidden state. It regresses the $\mathfrak{se}(3)$ transformation parameters for each frame in the input sequence. We use LSTM units as recurrent units throughout this paper. We discuss the training methodology in further detail in Sec. 3.5.

3.2.3 CTC block

The CTC block gathers outputs from the encoder and the recurrent units and applies CTCs to them. It is built using various layers that perform lie-algebraic operations. We briefly describe each layer in this block.

SE(3) Layers: The $SE(3)$ layers are responsible for mapping the estimated $\mathfrak{se}(3)$ parameters into the corresponding $SE(3)$ transformation matrix and vice versa. A 3D rigid body transformation $\mathbf{T} \in SE(3)$ is a rotation R and translation t in 3D space, and is defined as follows.

$$T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \text{ where } R \in SO(3) \text{ and } t \in \mathbb{R}^3 \quad (2)$$

We denote a local transformation between the camera poses at times t and $t+1$ as T_t^{t+1}. We use $\xi = (v^T \ \ \omega^T) \in \mathfrak{se}(3)$ to define the lie-algebraic exponential coordinates of the local transformation.

An element in $\mathfrak{se}(3)$ can be mapped to one in $SE(3)$ by using the exponential map, which is simply the matrix exponential over a linear combination of the generators of

the tangent-space at the identity element of the Lie group. The exponential map can be inverted to obtain the logarithm map from $SE(3)$ to $\mathfrak{se}(3)$. Our implementations of the SE(3) layers use the exponential and logarithm maps and their corresponding small-angle approximations presented in [3].

***CTC computation*:** The network has multiple CTC blocks, where each such block is responsible for the computation of one particular composition constraint. In effect, a CTC block computes the constraint in the following manner. It first obtains $\mathfrak{se}(3)$ estimates for the left-hand side (LHS) of the constraint from the recurrent block, maps them to transformation matrices in $SE(3)$, and composes (concatenates) all of them. Then, it obtains an independent $\mathfrak{se}(3)$ estimate for the right-hand side (RHS) of the constratint. The LHS and RHS estimates are then passed to the $\mathfrak{se}(3)$ loss layer described below, which evaluates the constraint and computes gradients.

3.3. Loss Functions

Our complete loss function consists of a CTC error term and a regularization term. Our loss terms are generic, and can be applied to supervised as well as unsupervised settings.

CTC Loss: This loss is dictated by the composite transformation constraints as described in Sec 3.1. It is computed between a direct transformation T_d predicted between non-consecutive frames and a composite transformation T_c, composed as a product of smaller sequential transformations resulting from the predictions for successive frames:

$$\mathcal{L}_{ctc} = \|\xi_d - \xi_c\|_2^2 \tag{3}$$

where ξ_d, ξ_c are the $\mathfrak{se}(3)$ exponential coordinates for the transforms T_c and T_d, respectively[2].

Regularization Term: While the unsupervised term $\mathcal{L}_{\text{ctc}}$ is useful for enforcing consistency, using the above term alone could result in a degenerate solution where the network can predict zeros for ξ_c, ξ_d. To prevent this collapse to a trivial solution, we introduce a regularization loss term. Specifically, we assume a prior on each of the transforms estimated by the network from a standard VO pipeline. Such a prior aids in avoiding trivial solutions and is inexpensive to obtain. For each transform ξ_* predicted by either the convolutional encoder or the recurrent unit, we have a prior $\hat{\xi}_*$ from a conventional visual odometry estimator, used as a regularization term.

$$\mathcal{L}_{reg} = \|\xi_* - \hat{\xi}_*\|_2^2 \tag{4}$$

[2]Since the 7scenes dataset has very little camera motion between frames, this choice of loss function (L2 norm between exponential coordinates) works here, but one may want to use the geodesic distance over $SE(3)$ for generality [3].

Again, it's essential to note that this estimator can be very noisy, and is used only in a *supporting role* to the CTC loss term.

The overall loss function is a weighted sum of the CTC loss and the regularization term. In the expression below, α, β are scalar weights associated with each of the loss terms.

$$\mathcal{L}_{final} = \alpha\mathcal{L}_{ctc} + \beta\mathcal{L}_{reg} \tag{5}$$

3.4. Covariance recovery

In VO, recovering the covariance of an estimate is very important, as it can be efficiently exploited when recovering global information using pose-graph optimization (e.g., in [17]). Kendall *et al.* [11] use dropout [20] as a means of bayesian approximation to recover covariance from relocalization estimates from a trained CNN.

Similarly, we use dropout at the penultimate fully connected layer of the convolutional encoder as well as the recurrent blocks. At test time, instead of removing dropout from the network, we retain dropout layers and generate K predictions for each input pair of frames. While generating each of these K estimates, we randomly drop a fraction γ of the units of the penultimate fully connected layer, which results in a different estimate in each pass. The hypothesis is that, if the network is very *confident* of its estimates, then the variance in the obtained samples must be low. We fit a Gaussian density function to the K samples and use this density for covariance recovery.

3.5. Training Details

For ease and flexibility during training, we divide our training process into two stages: a pre-training phase and a sequential training phase:

Pretraining phase: While we can train in an end-to-end fashion, we consider the option of pretraining the convolutional layers in order to provide structured and informative latent features as input to the LSTM during sequential training. The pre-training phase consists of training the output of the convolutional encoder against *noisy* VO estimates from a traditional odometry estimation framework. In the case of unsupervised training, when ground truth data may not be available, we rely on the frame-to-frame transformation estimates provided by RGB-D ORB-SLAM [17]. While these estimates are noisy, they provide a fair starting point for the network to learn from.

Sequential training phase: The sequential training phase consists of providing the network *windows* (sequences) of frame pairs as input.

For training the network we use the Adam optimizer [13], with an initial learning rate of 10^{-4}, and momentum equal to 0.9. We decrease the learning rate by a factor 0.5 every few epochs. We train for a total of 40 epochs. During the pre-training phase, we only use the regularization loss

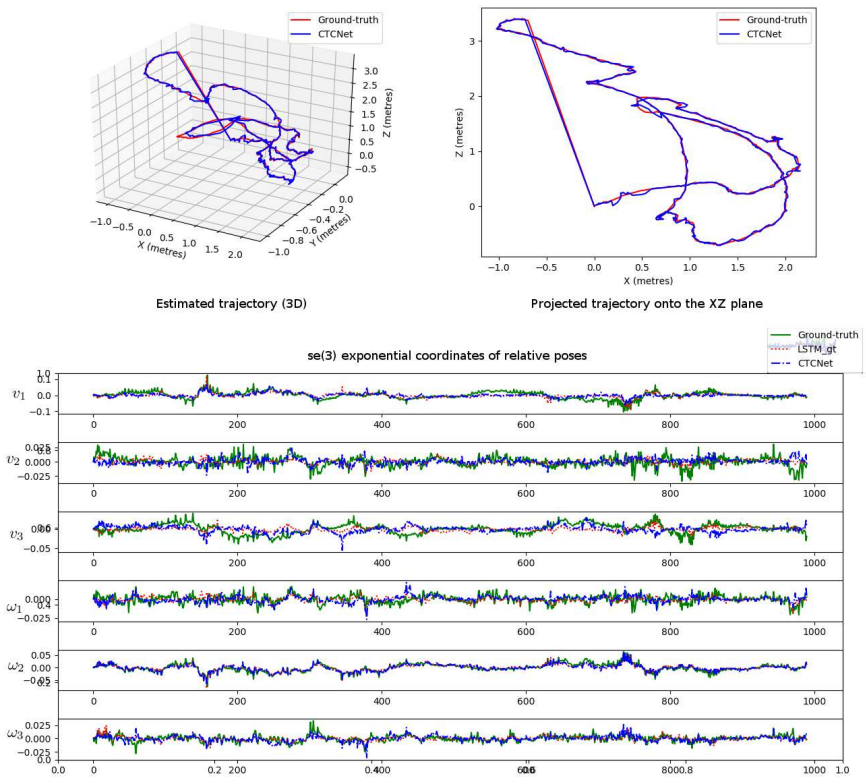

Figure 4. **Trajectory estimates** on a sequence from the 7-Scenes test split. Top (left to right): Output trajectories are shown in red, against ground-truth trajectories in blue. Bottom: $\mathfrak{se}(3)$ estimates of relative poses. Each of the 6 $\mathfrak{se}(3)$ coordinates is plotted independently. On this sequence, CTCNet performs better than LSTM$_{gt}$.

term. During sequential training, we start with sequences of length 3 frame pairs, i.e. $[I_t..I_{t+3}]$, and gradually increase to 18 frame pairs. We use several composite transformation constraints for each window, as well as regularization terms, with initial coefficient values $\alpha = 1$ and $\beta = 1$. We also experiment with end-to-end training of the full network, using the same loss terms. To prevent over-fitting, we apply a dropout of 0.7 at the penultimate fully connected layer of the convolutional encoder and the recurrent unit.

4. Experiments and Results

In this section, we describe the experiments carried out to analyze the efficacy of the proposed approach, and the findings we made in the process. We begin by describing the basic setup for various experiments, and then describe several variants of deep architectures that were evaluated. We then present qualitative, as well as quantitative comparisons and proceed to a discussion of further scope for work.

4.1. Dataset and Metrics

Dataset

Although most approaches [1, 14, 15, 18, 23, 24, 25, 26] evaluate their approach on the KITTI [5] benchmark, the camera in KITTI moves on the road plane and does not exhibit unconstrained 6 DoF motion. Wang *et. al.* [24] present results on a wide range of datasets, but their approach notably performs poorly when camera motion is unconstrained. To provide baseline results for several deep architecture on a challenging dataset, we conduct our experiments on the Microsoft 7-Scenes [6] dataset, which is increasingly being used to evaluate VO and/or relocalization performance [12, 22]. The dataset consists of tracked 640×480 resolution RGB camera frames collected using a handheld Microsoft Kinect camera. Although depth information is available, only the RGB images are used as input to all network variants we consider during training as well as testing.

The dataset consists of 7 scenes, with a total of 46 sequences comprising about 1000 frames each. We use the dataset-provided train/test splits for all our experiments. During the initial training phase, we often use frames that are randomly separated between 1-5 time steps apart in the same sequence. This allows for a wider range of transformations and allows for a higher number of training pairs. Overall, we composed a total of 49152 image pairs for training, 25686 image pairs for validation, and 15983 image

Network Architecture	Absolute Trajectory Error (ATE) (meters)			$\mathfrak{se}(3)$ error (L2-distance)		
	***redkitchen*-03**	***office*-02**	***fire*-04**	***redkitchen*-03**	***office*-02**	***fire*-04**
CNN_{gt}	0.0274 ± 0.0080	0.1216 ± 0.0820	0.1421 ± 0.0436	0.0293	0.0364	0.0354
LSTM_{gt}	0.0242 ± 0.0091	0.1119 ± 0.0327	0.1101 ± 0.0412	0.0257	0.0253	0.0291
CNN_{unsup}	0.0402 ± 0.0121	0.1394 ± 0.1027	0.1707 ± 0.0647	0.0382	0.0401	0.0368
LSTM_{unsup}	0.0392 ± 0.0121	0.1290 ± 0.0670	0.1700 ± 0.0513	0.0343	0.0400	0.0359
CNN_{aug}	0.0787 ± 0.0562	0.1662 ± 0.0908	0.1675 ± 0.0833	0.0605	0.0772	0.0547
LSTM_{aug}	0.0780 ± 0.0531	0.1318 ± 0.0613	0.1486 ± 0.0809	0.0352	0.0395	0.0345
ORB-SLAM	0.0326 ± 0.0140	0.1005 ± 0.0620	0.1057 ± 0.0515	0.0352	0.0426	0.0305
CTCNet	0.036 ± 0.0012	0.1226 ± 0.0183	0.12918 ± 0.0246	0.0286	0.0384	0.0338

Table 1. **Ablation analysis** of the proposed network architecture and variants. We evaluate the absolute trajectory error (ATE) (in meters). We also evaluate the relative pose estimation error in $\mathfrak{se}(3)$ exponential coordinates ($L2$-distance).

pairs (17 sequences) for testing.

Metrics

To compare the output trajectories of our approach with ground-truth, we use the absolute translation error (ATE) metric. Further, to evaluate the accuracy of relative pose estimation, we also analyze the $L2$-distance between the estimated $\mathfrak{se}(3)$ exponential coordinates and the corresponding ground-truth $\mathfrak{se}(3)$ vector.

4.2. Network Architectures Evaluated

We carry out extensive experiments on several variants of deep network architectures (supervised, unsupervised, stateless, stateful) for VO prediction, to analyze the benefits and pitfalls offered by each. Here, we enumerate each of the variants tested. The supervised variants are provided ground-truth pose estimates for supervision, whereas the unsupervised variants are trained without ground-truth pose information.

- CNN_{gt}: The convolutional encoder supervised using ground-truth pose information.
- LSTM_{gt}: The convolutional encoder with its output fed to the recurrent unit.
- CNN_{unsup}, LSTM_{unsup}: Unsupervised variants of CNN_{gt} and LSTM_{gt}, respectively.
- CNN_{aug}, LSTM_{aug}: Similar to CNN_{gt}, and LSTM_{gt} respectively. However, instead of simply taking in odometry estimates from ORB-SLAM [17], every time an image pair and its corresponding odometry esimate are drawn from a Gaussian distribution centered about the ORB-SLAM estimate, to account for noisy estimates.
- **CTCNet**: The proposed architecture that enforces composite transformation constraints (CTCs).

4.3. Results

We evaluate all network variants on the test split of the 7-Scenes dataset and the analysis is presented in Table 4. As one would expect, the networks trained against ground-truth turned out to be the best-performing models. Understandably, the LSTM variants achieved better performance compared to their convolutional counterparts, due to the additional context they store in their hidden state.

It can, however, be seen that CTCNet performs on par with supervised approaches, although it has been trained only using *noisy* estimates from a VO pipeline. Moreover, LSTM variants that were trained purely against the noisy estimates (i.e., without using CTC) perform poorly. Data augmentation / label noise shows a slight improvement in ATE, as evident from the CNN_{aug} and LSTM_{aug} results. In certain sequences, CTCNet achieves significantly lower error compared to CNN_{gt} and LSTM_{gt} (Fig. 4). This makes a strong case for using geometric consistency for unsupervised learning, especially in tasks such as visual odometry and SLAM.

4.3.1 Comparision with ORB-SLAM

From Table 4, we observe that CTCNet does better in terms of relative pose estimation when compared to ORB-SLAM and hence has a lower $\mathfrak{se}(3)$ error ($L2$-distance metric). However, ORB-SLAM does marginally better on ATE. This suggests that CTCNet performs better *locally*, whereas ORB-SLAM is better at a *global* level. Since ORB-SLAM is using keyframes it is able to optimize over an entire sequence of images with a similar viewpoint no matter how long it is. We believe this can be mitigated by training CTCNet on longer window lengths (currently it takes in a window of only 18 image pairs as input), but being able to flexibly control this window size the way that model-based

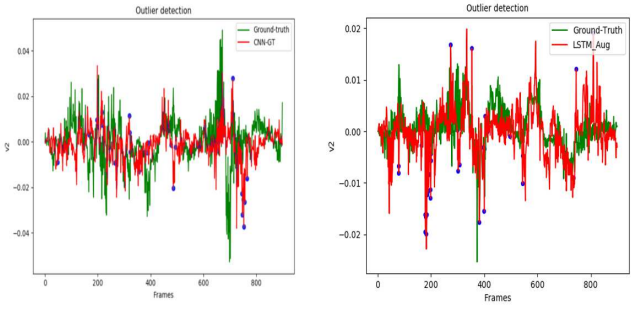

Figure 5. **Outlier detection**: Upon covariance recovery, estimates with covariance above a threshold are marked outliers (here shown in blue).

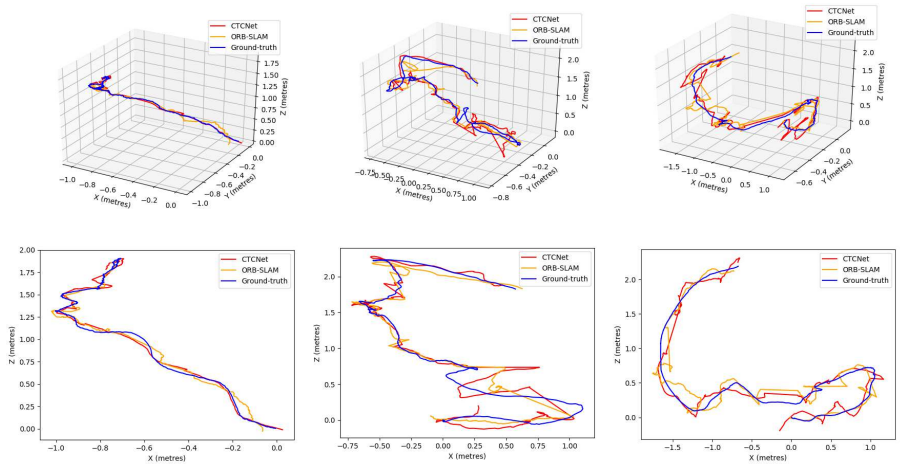

Figure 6. **Estimated trajectories**: (Top) Estimated 3D trajectories from CTCNet and ORB-SLAM plotted againscat ground-truth. (Bottom) 2D projections of these trajectories onto the XZ-plane.

approaches are able to[3] is not currently addressed in deep VO approaches and forms an interesting avenue for future work.

4.3.2 Uncertainty estimation

Fig. 5 illustrates the performance of the proposed covariance recovery scheme. Using a dropout (with drop ratio 10%), we draw 10 estimates per input pair using the CNN_{gt} model. We plot the estimated relative $\mathfrak{se}(3)$ coordinates with respect to those from ground-truth transforms. Fig. 5 shows this plot along the dimension v_2, i.e., translational velocity along the Y-axis. If the covariance of a particular estimate is very high (i.e., if the 10 estimates drawn have their variance above a set threshold), that estimate is characterized as an *outlier* (shown in blue). We see that the network reasonably detects and characterizes several outliers. This piece of information is valuable, especially in weighing these estimates when constructing a global representation (using pose graph optimization, for instance). Moreover, this covariance recovery need not be learned. It suffices if dropout layers are present during training. Investigation of how the estimated uncertainty can be exploited to suppress the effect of outliers (see Fig. 6) is deferred to future work.

4.3.3 Generalization

We also evaluate CTCNet in scenarios that it has never encountered during training. To do so, we train CTCNet using only 4 scenes from the 7-Scenes dataset (*chess*, *office*, *redkitchen*, and *stairs*). We evaluate VO estimation performance on a sequence from the *fire* scene and report the obtained trajectory in Fig. 7. This sequence bears no resemblance to the training data presented to CTCNet, either during the pre-training phase or the sequential training phase. However, it has been trained on very little data (4 scenes); training on more should improve performance. Moreover, CTCNet alleviates the need for hyperparameter tuning, which is frequently required for traditional VO pipelines such as ORB-SLAM [17].

[3]ORB-SLAM has this flexibility built-in, using keyframes. At a frame rate of 30 fps, a window-size of 18 frames would mean that ORB-SLAM has a very stable keyframe that boosts performance.

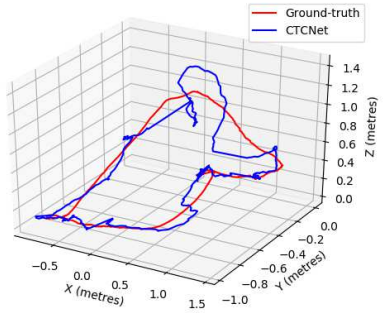

Figure 7. **Generalization to unseen data**: CTCNet was evaluated on a sequence that was in stark contrast to the kind of sequences it had been presented with during training. Estimated 3D trajectory plotted against ground-truth.

5. Conclusion

In this paper, we showcase a new end-to-end architecture for self-supervised training to regress pose transformations between monocular frame-pair sequences. We demonstrate the use of a differentiable flexible composite constraint and its application in both supervised and unsupervised settings. Our method works well in the supervised setting with reduced ATE, when tested on indoor sequences. In the future, we plan to extend the work to a full-fledged SLAM system. We would also look to generate and utilize depth information (RGB-D) of sequential scenes for dense reconstruction and trajectory estimation.

References

[1] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI, 2017. 1, 2, 6

[2] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 4

[3] E. Eade. Lie groups for 2d and 3d transformations. 5

[4] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014. 1, 2

[5] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 6

[6] B. Glocker, S. Izadi, J. Shotton, and A. Criminisi. Real-time rgb-d camera relocalization. *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2013. 2, 6

[7] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017. 2

[8] A. Handa, M. Bloesch, V. Pătrăucean, S. Stent, J. McCormac, and A. Davison. gvnn: Neural network library for geometric computer vision. In *ECCV Workshop on Geometry Meets Deep Learning*, 2016. 2

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory, 1995. 2

[10] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 4

[11] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2016. 2, 5

[12] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. *ICCV*, 2015. 1, 2, 6

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, 2014. 5

[14] K. Konda and R. Memisevic. Learning visual odometry with a convolutional network. *International Conference on Computer Vision Theory and Applications, VISAPP*, 2015. 1, 2, 6

[15] R. Li, S. Wang, Z. Long, and D. Gu. UnDeepVO: Monocular visual odometry through unsupervised deep learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018. 1, 2, 3, 6

[16] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, Feb 2016. 1

[17] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 2015. 1, 2, 5, 7, 8

[18] V. Peretroukhin and J. Kelly. Dpc-net: Deep pose correction for visual localization. *IEEE Robotics and Automation Letters (In Press)*, 2018. 1, 2, 6

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. 2, 5

[21] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *CVPR*, 2017. 1, 2

[22] A. Valada, N. Radwan, and W. Burgard. Deep auxiliary learning for visual localization and odometry. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2018. 6

[23] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *CoRR*, abs/1704.07804, 2017. 2, 3, 6

[24] 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*. 1, 2, 6

[25] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. *IEEE International Conference on Robotics and Automation (ICRA)*, 2017. 1, 2, 6

[26] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 1, 2, 3, 6