CeMNet: Self-supervised learning for accurate continuous ego-motion estimation

Minhaeng Lee
University of California, Irvine
minhaenl@ics.uci.edu

Charless C. Fowlkes
University of California, Irvine
fowlkes@ics.uci.edu

Abstract

In this paper, we propose a self-supervised learning approach for estimating continuous ego-motion from video. Our model learns to estimate camera motion by watching RGBD or RGB video streams and determining translational and rotation velocities that correctly predict the appearance of future frames. Our approach differs from other recent work on self-supervised structure-from-motion in its use of a continuous motion formulation and representation of rigid motion fields rather than direct prediction of camera parameters. To make estimation robust in dynamic environments with multiple moving objects, we introduce a simple two-component segmentation process that isolates the rigid background environment from dynamic scene elements. We demonstrate state-of-the-art accuracy of the self-trained model on several benchmark ego-motion datasets and highlight the ability of the model to provide superior rotational accuracy and handling of non-rigid scene motions.

1. Introduction

Supervised machine learning techniques based on deep neural networks have shown remarkable recent progress for image recognition and segmentation tasks. However, application of these powerful learning methods to geometric tasks such as structure-from-motion has been somewhat slower due to a number of factors. One challenge is that standard layers defined in convolutional neural network (CNN) architectures do not offer a natural way for researchers to incorporate hard-won insights about the algebraic structure of geometric vision problems, instead relying on general approximation properties of the network to re-discover these facts from training examples. This has resulted in some development of new building blocks (layers) specialized for geometric computations that can function inside standard gradient-based optimization frameworks (see e.g., [10, 11]) but interfacing these to image data is still a challenge.

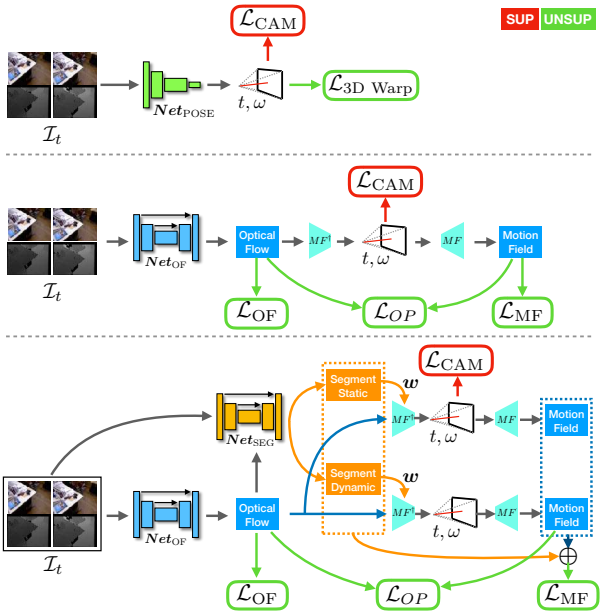


Figure 1. Overview of network architectures used in our experiments. The top panel shows conventional (*baseline*) approach that directly predicts 6DoF camera motion ($\mathcal{L}_{cam}$). The middle panel displays our proposed *single layer model* which predicts ego motion assuming a static (rigid) environment. We train the model with additional unsupervised losses based on optical flow ($\mathcal{L}_{OF}$), motion field ($\mathcal{L}_{MF}$), and orthogonal projection ($\mathcal{L}_{OF}$) described in Section 3.1. Our model supports both supervised (red) and unsupervised (green) losses during training. The bottom panel shows a *two layered model* variant that segments a scene into static and dynamic components and only uses static component for camera motion prediction. When input depth is not available, we utilize an additional monocular depth estimation network to predict it.

A second difficulty is that optimizing convolutional neural networks (CNNs) requires large amounts of training data with ground-truth labels. Such ground-truth data is often not readily available for geometric problems (e.g., requiring special-purpose hardware during acquisition rather than simple image annotations). This challenge has driven re-

cent effort to develop more realistic synthetic datasets such as Flying Chairs and MPI-Sintel [2] for flow and disparity estimation, Virtual KITTI [6] for object detection and tracking, semantic segmentation, flow and depth estimation, and SUNCG [30] for indoor room layout, depth and normal estimation.

In this paper, we overcome some of these difficulties by taking a "self-supervised" approach to learning to estimate camera motions directly from video. Self-supervision utilizes unlabeled image data by constructing an encoder that transforms the image into an alternate representation and a decoder that maps back to the original image. This approach has been widely for low-level synthesis problems such as super-resolution [3], image colorization [43] and inpainting [26] where the encoder is fixed (creating a downsampled, grayscale or occluded version of the image) and the decoder is trained to reproduce the original image. For estimation tasks such as human pose [34], depth [37, 44], and intrinsic image decomposition [15], the structure of the decoder is typically specified by hand (e.g., synthesizing the next video frame in a sequence based on estimated optical flow and previous video frame) and the encoder is learned. Self-supervision is appealing for geometric estimation problems since (a) it doesn't require human supervision to generate target labels and hence can be trained on large, diverse data, and (b) the predictive (decoder) component of the model can incorporate known constraints into the problem structure.

Our basic model for ego-motion estimation takes a pair of calibrated RGB or RGBD video frames as input, estimates optical flow and depth, determines camera and object velocities, and resynthesizes the corresponding motion fields. We show that the model can be trained end-to-end with a self-supervised loss that enforces consistency of the predicted motion fields with the input frames, yielding a system that provides highly accurate estimates of camera ego-motion. We measure the effectiveness of our method using TUM [31] and Virtual KITTI [6] dataset.

Relative to other recent papers [35, 37, 44, 21, 1] that have also investigated self-supervision for structure-from-motion, the novel contributions of our work are:

- We represent camera motion implicitly in terms of motion fields and depth which are a better match for CNNs architectures that naturally operate in the image domain (rather than camera parameter space). We demonstrate that this choice yields better predictive performance, even when trained in the fully supervised setting
- Unlike previous self-supervised techniques, our model uses a continuous (linearized) approximation to camera motion [25, 14] which is suitable for video odometry and allows efficient backpropagation while providing strong constraints for learning from unsupervised data.
- Our experimental results demonstrate state-of-the-art performance on benchmark datasets which include non-rigid scene motion due to dynamic objects. Our model improves substantially on estimates of camera rotation, suggesting this approach can serve well as a drop-in replacement for local estimation in existing RGB(D) SLAM pipelines.

2. Related Work

Visual odometry is a classic and well studied problem in computer vision. Here we mention a few recent works that are most closely related to our approach.

Optical Flow, Depth and Odometry: A number of recent papers have shown great success in estimation of optical flow from video using learning-based techniques [4, 12]. Ren *et al.* introduced unsupervised learning for optical flow prediction [27] using photometric consistency. Garg et. al. utilize consistency between stereo pairs to learn monocular depth estimation in a self-supervised manner [7]. [44] jointly trains estimators for monocular depth and relative pose using an unsupervised loss. SfM-Net [37] takes a similar approach but explicitly decomposes the input into multiple motion layers. [19] uses stereo video for joint training of depth and camera motion (sometimes referred to as scene flow) but tests on monocular sequences. Mahjourian *et al*. [21] use 3D ICP loss on top of 2D photometric loss to predict depth and ego-motion. Our approach differs from these recent papers in using a continuous formulation appropriate for video. Such a formulation was recently used by Jaegle *et al.*[14] for robust monocular ego-motion estimation but with classic (sparse) optical flow as input.

SLAM: While conventional simultaneous localization and mapping (SLAM) methods estimate geometric information by extracting feature points [16, 40] or use all information in the given images [5], recently several learning based methods have been introduced. Tateno *et al.* [33] propose a fusion SLAM technique by utilizing CNN based depth map prediction and monocular SLAM. Melekhov *et al.* propose CNN based relative pose estimation using end-to-end training with a spatial pyramid pooling (SPP) [22]. Other recent works [17, 20] model static background to predict accurate camera pose even in dynamic environment. Sun *et al.* try to solve dynamic scene problem by adding motion removal approach as a pre-processing to be integrated into RGBD SLAM [32]. The work of Wang *et al.* [38] train a recurrent CNN to capture longer-term processing of sequences typically handled by bundle adjustment and loop closure. [42] use virtual stereo camera based training to achieve photo-consistency and accurate depth reconstruction. Another recent work done by [1] shows scale-aware camera pose prediction by using spatial and temporal reconstruction losses

simultaneously.

3. Continuous Ego-motion Network

Figure 1 provides an overview of three different types of architectures we consider in this paper. We take as input a successive pair of RGB images $\{I_t, I_{t+\delta}\}$ and corresponding depth images $\{d_t, d_{t+\delta}\}$. When depth is not available, we assume it is predicted by a monocular depth estimator (not shown). The first network, $\boldsymbol{Net}_{\text{POSE}}$, directly predicts 6 DoF camera motion by attaching several fully connected layers at the end of a standard CNN architecture. When camera motion is known, this baseline can be trained with a supervised loss $\mathcal{L}_{\text{CAM}}$ or trained with a self-supervised image warping loss $\mathcal{L}_{\text{3DWARP}}$ as done in several recent papers [44, 35, 37].

Instead of directly predicting camera motion parameters, we advocate utilizing a fully-convolutional encoder/decoder architecture with skip connections (e.g., [28, 29, 4, 12]) to first predict optical flow (denoted $\boldsymbol{Net}_{\text{OF}}$). We then estimate continuous ego-motion (t, ω) using weighted least-squares and resynthesize the corresponding motion field $MF(t, \omega)$. These intermediate representations can be learned using unsupervised losses ($\mathcal{L}_{\text{OF}}$,$\mathcal{L}_{\text{MF}}$, $\mathcal{L}_{\text{OP}}$) described below. When additional moving objects are present in the scene, we introduce an additional segmentation network, $\boldsymbol{Net}_{\text{SEG}}$, which decomposes the optical flow into layers that are fit to separate motion models.

In the following sections we develop the continuous motion formulation, interpret our model as projecting the predicted optical flow on to the subspace of ego-motion flows, and discuss implementation of segmentation into layers.

3.1. Estimating Continuous Ego-motion

Consider the 2D trajectory of a point in the image $\boldsymbol{x} = \{x, y\}$ as a function of its 3D position $\boldsymbol{X} = \{X, Y, Z\}$ and motion relative to the camera. We write

$$\boldsymbol{x}(t) \quad = \quad \{x(t), y(t)\} = \left\{\frac{fX(t)}{Z(t)}, \frac{fY(t)}{Z(t)}\right\},$$

where f is the camera focal length. To compute the projected velocity in the image $\boldsymbol{v}(\boldsymbol{x}) = (v_x, v_y)^\top \in \mathbb{R}^2$ as a function of the 3D velocity $\boldsymbol{V}(\boldsymbol{X})$ we take partial derivatives. For example, the x component of the velocity is:

$$\begin{aligned}\frac{\partial x(t)}{\partial t} &= \frac{f}{Z(t)}\frac{\partial X(t)}{\partial t} + fX(t)\cdot\frac{\partial}{\partial t}\frac{1}{Z(t)} \\ &= \frac{f}{Z(t)}V_x - fX(t)\cdot\frac{1}{Z^2(t)}\cdot\frac{\partial Z(t)}{\partial t} \\ &= \frac{1}{Z(t)}\begin{bmatrix} f & 0 & -x(t)\end{bmatrix}\begin{bmatrix} V_x \\ 0 \\ V_z\end{bmatrix}\end{aligned}$$

Dropping t for notational simplicity, we can thus write the image velocity as:

$$\boldsymbol{v}(\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})}A(\boldsymbol{x})\boldsymbol{V}(\boldsymbol{X}) \tag{1}$$

where the matrix $A(\boldsymbol{x})$ is given by:

$$A(\boldsymbol{x}) = \begin{bmatrix} f & 0 & -x \\ 0 & f & -y\end{bmatrix}.$$

In the continuous formulation, the velocity of the point relative to the camera $\boldsymbol{V}(\boldsymbol{X})$ arises from a combination of translational and rotational motions,

$$\boldsymbol{V}(\boldsymbol{X}) = \boldsymbol{\tau} + \boldsymbol{X}\times\boldsymbol{\omega}$$

where $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^\top \in \mathbb{R}^3$ is unit length axis representation of rotational velocity of the camera and $\boldsymbol{\tau} = (\tau_x, \tau_y, \tau_z)^\top \in \mathbb{R}^3$ is the translation. Denoting the inverse depth at image location $\boldsymbol{x}$ by $\rho(x) = \frac{1}{Z(x)}$, we can see that the projected motion vector $\boldsymbol{v}$ is a linear function of the camera motion parameters:

$$\begin{aligned}\boldsymbol{v}(\boldsymbol{x}) &= \rho(\boldsymbol{x})A(\boldsymbol{x})\boldsymbol{\tau} + B(\boldsymbol{x})\boldsymbol{\omega} \\ &= \begin{bmatrix}\rho(\boldsymbol{x})A(\boldsymbol{x}) & B(\boldsymbol{x})\end{bmatrix}\begin{bmatrix}\boldsymbol{\tau} \\ \boldsymbol{\omega}\end{bmatrix} \\ &= Q(\boldsymbol{x})\boldsymbol{T},\end{aligned}$$

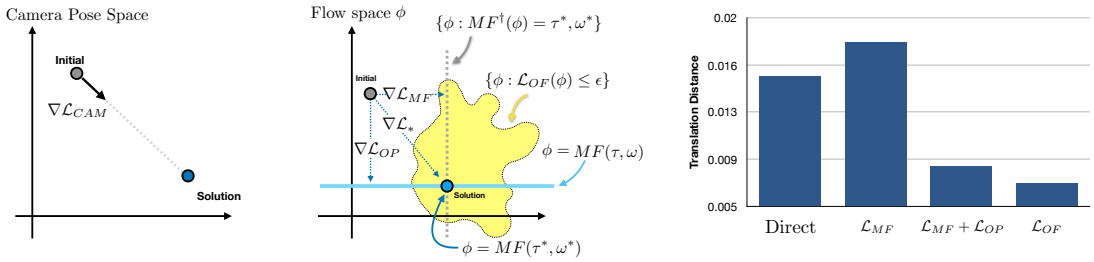
where the matrix B includes the cross product

$$B(\boldsymbol{x}) = \begin{bmatrix} -xy & f+x^2 & -y \\ -f-y^2 & xy & x\end{bmatrix}.$$

To describe motion field for the whole image, we concatenate equations for all N pixel locations and write $\mathcal{U} = \mathcal{Q}\boldsymbol{T}$ where

$$\begin{aligned}\mathcal{U} &= \begin{bmatrix}\boldsymbol{v}(\boldsymbol{x}_1) \\ \boldsymbol{v}(\boldsymbol{x}_2) \\ \vdots \\ \boldsymbol{v}(\boldsymbol{x}_N)\end{bmatrix} \in \mathbb{R}^{2N\times 1}, \\ \mathcal{Q} &= \begin{bmatrix}\rho_1 A(\boldsymbol{x}_1) & B(\boldsymbol{x}_1) \\ \rho_2 A(\boldsymbol{x}_2) & B(\boldsymbol{x}_2) \\ \vdots & \vdots \\ \rho_N A(\boldsymbol{x}_N) & B(\boldsymbol{x}_N)\end{bmatrix} \in \mathbb{R}^{2N\times 6}, \\ \boldsymbol{T} &\in \mathbb{R}^{6\times 1}.\end{aligned}$$

We assume the focal length is a fixed quantity and in the following write the motion field as a function $\mathcal{U} = MF(\boldsymbol{\rho}, \boldsymbol{T})$ which is linear in both the inverse depths $\boldsymbol{\rho}$ and camera motion parameters $\boldsymbol{T}$.



(a) Predict pose directly (b) Predict pose via flow space (c) losses comparison

Figure 2. Schematic interpretation of different loss functions. (a) Supervised training of direct models utilize a loss defined on camera pose space. (b) Our approach defines losses on the space of pixel flows and considers losses that measure the distance to the true motion field, the sub-space of possible ego-motion fields (blue), and its orthogonal complement (gray dashed). The model is also guided by photometric or scene-flow consistency between input frames (yellow) (c) shows prediction error for supervised models trained with different combinations of these losses and indicates that using losses defined in flow-space outperforms direct prediction of camera motion.

To infer the camera motion $\boldsymbol{T}$ given inverse depths $\boldsymbol{\rho}$ and image velocities $\mathcal{U}$, we use a least-squares estimate:

$$\boldsymbol{t}^*, \boldsymbol{\omega}^* = \arg\min_{\boldsymbol{t},\boldsymbol{\omega}} \sum_{i=1}^{N} w(\boldsymbol{x_i}) ||v(\boldsymbol{x}_i) - \frac{1}{Z(\boldsymbol{x}_i)} A(\boldsymbol{x}_i)t + B(\boldsymbol{x}_i)\omega||^2$$

where $w(\boldsymbol{x_i})$ is a weighting function that models the reliability of each pixel velocity in estimating the camera motion. The solution to this problem can be expressed in closed from using the pseudo inverse of matrix $\mathcal{Q}$. We denote the mapping from $\mathcal{U}$ to estimated camera motion as $\boldsymbol{T} = MF^{\dagger}(\boldsymbol{\rho}, \mathcal{U}, \boldsymbol{w})$.

In our model we utilize $MF^{\dagger}(\boldsymbol{\rho}, \mathcal{U}, \boldsymbol{w})$ to estimate camera model and $MF(\boldsymbol{\rho}, \boldsymbol{T})$ to resynthesize the resulting motion field. Both functions are differentiable with respect to their inputs (in fact linear in $\mathcal{U}$ and $\boldsymbol{T}$ respectively) making it straightforward and efficient to incorporate them into a network that is trained end-to-end using gradient-based methods.

3.2. Projecting optical flow onto ego-motion

Given the true motion field $\mathcal{U}$, it is straight forward to estimate the the true camera motion $\boldsymbol{T}^*$. In practice, the motion must be estimated from image data which is often ambiguous (e.g., due to lack of texture) and noisy. Typically there is a large set of image flows that are photometrically consistent from which we must select the true motion field. Our architecture utilizes a CNN to generate an initial flow estimate from image data, then uses $MF^{\dagger}(\boldsymbol{\rho}, \mathcal{U}, \boldsymbol{w})$ to fit a camera motion and finally reconstructs the image motion field corresponding to the camera motion. The composition of $MF^{\dagger}$ and MF can be seen as a linear projection of the initial flow estimate into the space of continuous motion fields.

A key tenant of our approach is that it is a better match to convolutional feature extractors to predict the ego motion field in the image domain (and subsequently estimate camera motion) rather than attempting to directly regress camera pose. In particular, this allows for richer loss functions that guide the training of the network. We illustrate this idea schematically for the case of supervised learning in Figure 2. Panel (a) depicts the direct approach in terms of a loss function whose gradient pulls the predicted pose towards the true pose.

We display the relationship between optical flow, motion field and camera pose in Figure 2(b). Among all possible image flows ϕ, we indicate in yellow the set which are photometrically valid (i.e., have a zero warping loss $\mathcal{L}_{OF} \leq \epsilon$). The blue line indicates the 6-dimensional subspace consisting of those motion fields that can be generated by all possible camera velocities (conditioned on scene depth). Introducing a loss on the camera pose (either directly on the prediction τ, ω, or on the resynthesized motion field $MF(\tau, \omega)$ serves to pull the flow prediction towards the orthogonal complement of this space (i.e., the set $\{\phi : MF^{\dagger}(\phi) = \tau^*, \omega^*\}$ denoted by the gray vertical line).

Our approach allows the consideration of two other loss functions that can provide additional guidance. When supervision is available, we can utilize a loss which directly measures the distance between the predicted flow and the true motion field ($\mathcal{L}_*$ in the figure). In the self-supervised setting, we can approximate this with the photometric warping loss $\mathcal{L}_{OF}$. In either supervised or unsupervised settings, we can include an orthogonal projection loss $\mathcal{L}_{OP}$, which encourages the model to predict flows that are close to the space of motion fields. In section , we describe how these losses are computed and adapted to the unsupervised setting.

While all of these losses are minimized in a perfect model, Figure 2(c) shows that this choice of loss during training has a substantial practical effect. In the supervised setting, optimizing the direct loss in the camera pose space (using generic fully connected layers), or in the flow space (using our least-squares fitting) results in similar prediction errors. However, adding the projection loss or directly minimizing the distance to the true motion field yields substan-

tially better predictions (i.e., halving average camera translation error).

3.3. Static and Dynamic Motion Layers

So far, our description has assumed a camera moving through a single rigid scene. A standard approach to modeling non-rigid scenes (e.g., due to relative motion of multiple dynamic objects in addition to ego-motion) is to split the scene into a number of layers where each layer has a separate motion model [39]. For example, Zhou *et al.* use a binary "explainable mask" [44] to exclude outlying motions, and Vijayanarasimhan *et al.* segment images into K regions based on motion [37]. However, in the later-case, there is no distinction between object motion and ego motion making it inappropriate for odometry.

We use a similar strategy in order to separate motion into two layers corresponding to static background and dynamic objects (outliers). We feed a pair of images and their predicted optical flow into a u-net-like segmentation network [28] to predict this separation which then defines the weights used for camera motion estimation using pseudo inverse function $MF^{\dagger}(\cdot)$ described in Section 3.1.

Consider a scene divided into K regions corresponding to moving objects and rigid background. Let $Seg_i(x) \in \{0, 1\}$ denote a mask that indicates the image support of region i and $\mathcal{U}^i$ denote the corresponding rigid motion field for that object considered in isolation. The composite motion field for the whole image $\mathcal{U}$ can be written as:

$$\mathcal{U} = \sum_{i}^{K} Seg_i \cdot \mathcal{U}^i,$$

In the odometry setting, we are only interested in the motion of the camera relative to static background. We thus collect any dynamic objects into a single motion field and consider a single binary mask:

$$\mathcal{U}(\boldsymbol{x}_i) \approx Seg_s(\boldsymbol{x}_i)\mathcal{U}^s + Seg_d(\boldsymbol{x}_i)\mathcal{U}^d.$$

In our training with this segmentation network, we use the approximated motion field $\mathcal{U}$ for the photometric warping loss described below. For simplicity, we refer our single layer model as CeMNet[1] and dual layer model as CeMNet[2]

In Figure 3, we illustrate intermediate results demonstrating how the 2 layer model can better estimate camera motion in the presence of dynamic objects. Since the single layer model cannot distinguish background and foreground, the quality of predicted camera pose is bad. Excluding the dynamic scene components from the camera motion estimation provides substantially better pose estimation as seen in panels (i) and (l) which show less photometric warping error on the scene background relative to the single layer model shown in (f).

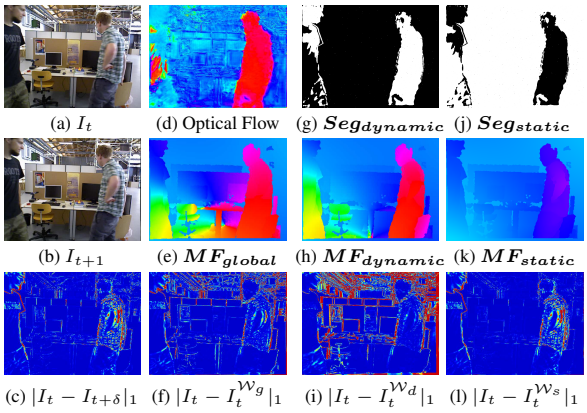


Figure 3. A sample result on a dynamic sequence from TUM [31]. From an input frame pair (a) and (b), $\boldsymbol{Net}_{OF}$ predicts optical flow (d). Both camera and object motion are visible in the frame difference (c). A single motion field (e) is dominated by large object motions and yields poor warping error (f), particularly on the background. Our model includes a segmentation network $\boldsymbol{Net}_{seg}$ that divides the image into dynamic and static masks (g,j) and fits corresponding motion fields (h,k). These provide better warping error on the objects (i) and background (l) respectively.

Hard assignment to layers: Previous work such as [37] uses a soft probabilistic prediction of layer membership (i.e., using a softmax function to generate layer weights). However, such an approach introduces degeneracy since it can utilize weighted combinations of two motions to match the flow (e.g., even in a completely rigid scene). We find that using hard assignment of motions to layers yields superior camera motion estimates. We utilize the "Gumbel sampling trick" described in [36] to implement hard assignment while still allowing differentiable end-to-end training of both the flow and segment networks.

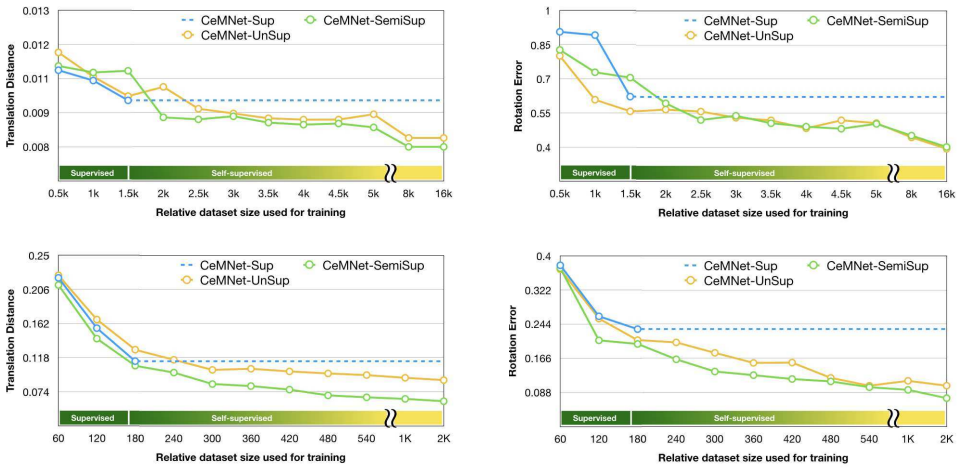
4. Training Losses

Losses for Self-supervision As described in Section 3.2, there are several different losses which can be applied to predicted flows. Here we adapt them to the self-supervised setting. The basic building block is to check if a predicted flow is photometrically consistent with the input image pairs.

For a given optical flow $\mathcal{U}^{OF}$ and source image $I_{t+\delta}$ then we can synthesize warped image $I_t^{\mathcal{W}_{OF}}$ and check if it matches I_t. As described in [13], this type of spatial transformation can be carried out in a differentiable framework using bilinear interpolation:

$$I_t^{\mathcal{W}^{OF}}(x_i) = \sum_{i\in\{t,b\},j\in\{l,r\}} w^{ij} I_{t+\delta}(x_i + \mathcal{U}^{OF}(x_i)),$$

where w^{ij} denotes the bilinear weighting of the four sample points. For simplicity, we write $I_t^{\mathcal{W}^{OF}}(x_i) = \mathcal{W}(I_{t+\delta}, \mathcal{U}^{OF})$ to denote the warping of $I_{t+\delta}$ using flow



(a) Translation Error (b) Rotation Error

Figure 4. Camera motion error on held-out test data as a function of training set size for TUM (top) and Virtual KITTI (Bottom) RGBD datasets. The blue line denotes training a supervised model that can't exploit unlabeled data. Introducing self-supervised warping losses yields much better performance when either using only unsupervised training (yellow) or semi-supervised training (green). Surprisingly, unsupervised training is actually competitive with supervised training for estimating rotation (b) but performs worse for translation (a).

$\mathcal{U}^{OF}$. We then define the self-supervised flow loss using the photometric error over all pixels:

$$\mathcal{L}_{OF} = \sum_{i=1}^{N} ||I_t(x_i) - I_t^{\mathcal{W}^{OF}}(x_i)||_1$$

This loss serves as an approximation of $\mathcal{L}_2$ when the predictions are far from the true motion field.

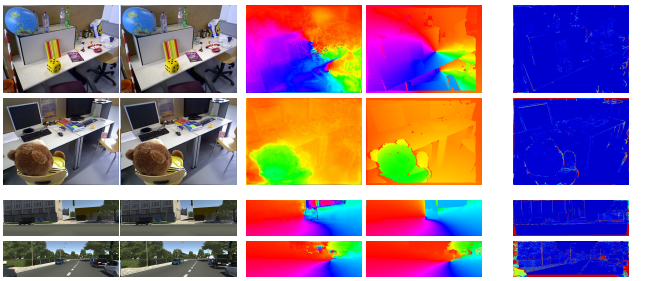
We can similarly apply warping loss is possible to the reconstructed motion field rather than the initial prediction. If the motion field we found is correct, then again, the warped image should be matched with the target image. We can build motion field loss by using motion-field warped image $I_t^{\mathcal{W}^{MF}} = \mathcal{W}(I_{t+\delta}, \mathcal{U}^{MF})$ as:

$$\mathcal{L}_{MF} = \sum_{i=1}^{N} P_t(x_i) ||I_t(x_i) - I_t^{\mathcal{W}^{MF}}(x_i)||_1$$

where the mask $P_t(x_i)$ is 1 when the depth at x_i is valid, 0 otherwise. This is necessary when using a depth sensor which doesn't provide depths at every image location. This loss acts as a proxy for minimizing the camera motion estimation error by lifting the prediction back to the flow space. When we predict camera motion for static scene, we use the global motion field, and for the dynamic scene, we use composite motion field $\mathcal{U}$.

Finally, we can utilize the orthogonal projection loss to minimize the distance between predicted optical flow and its projection onto the space of motion fields via:

$$\mathcal{L}_{OP} = \sum_{i}^{N} ||\mathcal{U}^{OF} - \mathcal{U}^{MF}||_1$$



(a) I_t (b) $I_{t+\delta}$ (c) Optical flow (d) Motion field (e) $|I_t - I_t^{\mathcal{W}_{MF}}|_1$

Figure 5. Visualizations of our single layered model. Top three rows come from TUM [31] dataset and bottom three come from Virtual KITTI [6]. From the input images (a) and (b), the predicted flow, and recovered motion field are displayed in (c) and (d) respectively. Since motion field is derived from camera pose estimate, the error between I_t and motion field based warped image $I_t^{\mathcal{W}_{MF}}$ reflects the accuracy of predicted camera motion. If the predicted camera pose and depth is ideal, then the error in (e) should be zero.

By combining three above losses, we can define the final self-supervised loss function

$$\mathcal{L}_{Final} = \lambda_{OF}\mathcal{L}_{OF} + \lambda_{MF}\mathcal{L}_{MF} + \lambda_{OP}\mathcal{L}_{OP},$$

where λ_{OF}, λ_{MF} and λ_{OP} weigh relative importance (we use 1, 0.1 and 0.1 respectively in our experiments).

Semi-supervision for symmetry breaking In our segmentation network, we predict two layers corresponding to static and dynamic parts. However, in the unsupervised setting, the loss is symmetric with respect to which segment label is considered background. This symmetry problem can

Seq.	DVO-SLAM [16]	Kintinuous [41]	ElasticFusion [40]	ORB2 [23]	CeMNet(RGBD)
fr1/desk	0.021	0.037	0.020	0.016	**0.0089**
fr1/desk2	0.046	0.071	0.048	0.022	**0.0129**
fr1/room	0.043	0.075	0.068	0.047	**0.0071**
fr2/xyz	0.018	0.029	0.011	0.004	**0.0009**
fr1/office	0.035	0.030	0.017	0.010	**0.0041**
fr1/nst	0.018	0.031	0.016	0.019	**0.0117**
fr1/360	0.092	-	-	-	**0.0088**
fr1/plant	0.025	-	-	-	**0.0061**
fr1/teddy	0.043	-	-	-	**0.0139**

Table 1. Relative translation error on TUM [31] static dataset. Most of the methods in this table use RGBD frames camera for pose prediction. Our model is trained without any supervised data.

interfere with training of the model and affect final performance. To break this symmetry, we found it most effective to utilize a small amount of supervised data where camera motion is known. For the supervised data we use an additional loss term on the camera motion estimated for the background layer.

Our network predicts camera motion in an axis-angle representation that includes translation part $\boldsymbol{t} \in \mathbb{R}^3$ and rotation $\boldsymbol{w} \in \mathbb{R}^3$. For supervised loss, we treat these two components separately in order to match the criteria typically used in benchmarking pose estimation performance.

Following [31], we compute the difference between our predicted camera motion and the ground truth $\boldsymbol{Q}^d = (\boldsymbol{Q}^p)^{-1}\boldsymbol{Q}^{gt}$ where $\boldsymbol{Q} \in \mathbb{R}^{4\times4}$ and penalize the translation and rotation components respectively by:

$$\mathcal{L}_{trans} = ||Q_t^d||_2$$

$$\mathcal{L}_{rot} = \arccos\left(\min\left(1, \max\left(-1, \frac{Tr(Q_r^d)-1}{2}\right)\right)\right)$$

5. Experimental Results

For the following experiments, we use the synthetic Virtual KITTI dataset [6] depicting street scenes from a moving car, and the TUM RGBD dataset [31] which has been used to benchmark a variety of RGBD odometry algorithms. To measure performance, we use relative pose error protocol proposed in [31].

Self-supervised learning improves model performance: To show the benefits of self-supervision, we assume that only 10% of each dataset has ground-truth available. We use 11 different sequences from the TUM dataset as training, choose a random ordering of frame pairs over the whole dataset and train models with increasingly large subsets of the data and test on a separate held-out collection of frames. This allows us to evaluate the effect of growing the amount of supervised/unsupervised training data in a consistent way across models.

In Figure 4, we plot the relative translation/rotation errors as a function of training data size. The supervised version of the model (CeM-Sup) can only be trained on the first 10% of the dataset and makes no use of the unsupervised data. In this setting it outperforms the unsupervised model (CeM-Unsup). However, as the amount of unsupervised training data continues to grow, CeM-Unsup eventually outperforms the supervised model. For a clear comparison, the unsupervised losses are not used in training (CeM-Sup). We also compare a model which uses both supervised and unsupervised loss (CeM-SemiSup) which generally yields even better performance. We note that because the real world depth data in TUM is incomplete, limiting performance of the supervised model while the supervised model shows expected decreasing errors on Virtual KITTI.

Seq.	TUM [31]		SfM-Net [37]		CeMNet(RGB)	
	Trans	Rot	Trans	Rot	Trans	Rot
fr1/desk	**0.008**	0.495	0.012	0.848	0.0113	**0.6315**
fr1/desk2	0.099	0.61	**0.012**	0.974	0.0133	**0.7548**
fr1/360	0.099	0.474	**0.009**	1.123	0.0091	**0.5455**
fr1/plant	0.016	1.053	0.011	0.796	**0.0083**	**0.5487**
fr1/teddy	0.020	1.14	0.0123	0.877	**0.0113**	**0.6460**

Table 2. To compare our model to RGB odometry methods, we use an off-the-shelf monocular depth estimator [18].

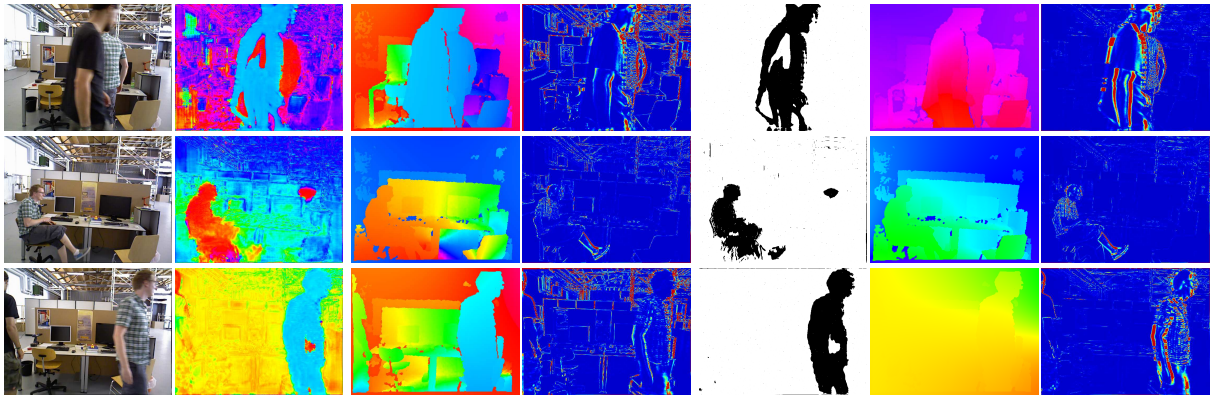
	Training		Testing		
	GT Depth	GT Cam	GT Depth	Trans	Rot
Geometric [14]	-	-		0.4579	0.3423
AIGN-SfM [35]	✓	✓		0.1247	0.3333
CeMNet(RGBD)	✓		✓	**0.0878**	**0.0781**
CeMNet(RGB)				0.0941	0.1079

Table 3. Relative pose error comparison using Virtual KITTI [6]. Both with (CeMNet(RGBD)) and without (CeMNet(RGB)) depth inputs, our models outperform previous methods.

Motion field and warping: In Section 4, we describe how a predicted camera pose is used to generate motion field and used in the warping loss. In Figure 5, we plot the per-pixel warping loss for several inputs. Left two (a-b) show the input RGB frames, (c) shows predicted optical flow. (d) is regenerated motion field. (e) shows differences between the target image and warped image. Note that blue color means lower differences between those two images.

Camera motion error comparison: To measure the quality of predicted camera pose, we compare our single layer model (CeMNet) with previous RGBD SLAM methods on the TUM dataset in Table 1. CeMNet(RGBD) shows the best average performance among tested methods in terms of relative translation error. Several previous methods of interest, including [44, 37] do not utilize depth as an input, instead predicting it directly from input images.

For fair comparison, we also test our model with predicted depth (CeMNet(RGB)) using off-the-shelf the monocular depth prediction model introduced by Iro *et al.* [18] which was trained using NYU Depth dataset



(a) I_t (b) Optical flow (c) $\boldsymbol{MF}$ (all) (c) $\boldsymbol{WE}$ (all) (d) $\boldsymbol{Seg}$ (static) (e) $\boldsymbol{MF}$ (static) (f) $\boldsymbol{WE}$ (static)

Figure 6. Intermediate results of two layered model for dynamic scene camera pose prediction. Without separating static and dynamic components, it is difficult to get good camera motions (high error in (c)). However, as shown in (f), it is possible to predict camera motion for background by fitting only the static segment (d).

Seq.	Baseline		CeMNet[1]		CeMNet[2]		CeMNet[2](Semi)	
	Trans	Rot	Trans	Rot	Trans	Rot	Trans	Rot
fr3/sit_static	0.0134	0.5724	0.0025	0.1667	0.0016	0.1573	**0.0010**	**0.1527**
fr3/sit_xyz	0.0179	0.7484	0.0070	0.2645	0.0068	0.2653	**0.0064**	**0.2612**
fr3/sit_halfsph	0.0104	1.0135	0.0081	**0.5272**	0.0080	0.5820	**0.0074**	0.5552
fr3/walk_static	0.0149	0.5703	0.0103	0.2107	0.0030	0.1610	**0.0019**	**0.1583**
fr3/walk_xyz	0.0174	0.7952	0.0128	0.3338	0.0079	**0.2915**	**0.0078**	0.2921
fr3/walk_halfsph	0.0166	0.9426	0.0147	0.4698	0.0107	0.4120	**0.0102**	**0.3989**

Table 4. Relative pose error comparison using TUM dynamic dataset [31]. Generally, the two layered model shows better performance than single layered model. Including a small amount of supervision (CeMNet$^2(Semi)$) yields equivalent or better performance by breaking the symmetry of the unsupervised loss.

V2 [24]. We rescale the predictions by 0.9 to match the range of depths in TUM (presumably due to differences in focal length) but otherwise leave the model fixed. As shown in Table 2, our method continues to outperform others in terms of rotation and shows comparable translation errors.

Additionally, we show performance on the Virtual KITTI dataset in Table 3. We specify how each method uses the available ground truth depth and camera pose data available for train and test. Using the true depth at test time results in strong performance from our model. For fair comparison, we also evaluate our model using the monocular depth prediction model of [9] trained with KITTI [8] dataset and converted from the predicted disparity to depth[1]. The results show better performance than previous self-supervised approaches even without using ground-truth depth.

Static/Dynamic segmentation: In Figure 6, we visualize the results of breaking the input into static and dynamic layers. From the RGB input pair at I_t (a) and $I_{t+\delta}$, predicted optical flow is shown in (b). While single layered model generates motion field using the complete flow, the two layer model fits separate motions which segments moving objects and yields reduced warping error ((c) vs (f)), especially in the static background region.

We perform a quantitative comparison on the TUM dynamic dataset which includes both object and camera motion. The results results are shown in Table 4. While single layered models such as the baseline direct prediction model and CeMNet[1] are sensitive to dynamic objects, two layered model CeMNet[2] shows less pose error. However, as noted previously, the unsupervised loss suffers from a symmetry as to which layer correspond to ego-motion. We evaluate the use of a small amount of supervised data (10%) to break this symmetry in the segmentation prediction network. This yields the the lowest resulting motion errors across nearly all test sequences.

6. Conclusion

In this paper, we have introduced a novel self-supervised approach for ego-motion prediction that leverages a continuous formulation of camera motion. This allows for linear projection of flows into the space of motion fields and (differentiable) end-to-end training. Compared to direct prediction of camera motion (both our own baseline implementation and previously reported performance), this approach yields more accurate two-frame estimates of camera motions for both RGBD and RGB odometry. Our model exploits self-supervised training, allowing it to make effective use of "free" unsupervised data. Finally, by utilizing a two-layer segmentation approach makes the model further robust to the presence of dynamic objects in a scene which otherwise interfere with accurate ego-motion estimation.

Acknowledgements: This work was supported by NSF grants IIS-1813785, IIS-1618806, IIS-1253538 and a hardware donation from NVIDIA.

[1]We use 0.54 as baseline distance and 725 for focal length

References

[1] V Babu, Anima Majumder, Kaushik Das, Swagat Kumar, et al. A deeper insight into the undemon: Unsupervised deep network for depth and ego-motion estimation. *arXiv*, 2018.

[2] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, pages 611–625, 2012.

[3] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE TPAMI*, 38(2):295–307, 2016.

[4] A Dosovitskiy, P Fischery, E Ilg, P Häusser, C Hazirbas, V Golkov, P v d. Smagt, D Cremers, and T Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015.

[5] Jakob Engel and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, 2014.

[6] A Gaidon, Q Wang, Y Cabon, and E Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.

[7] Ravi Garg, Vijay Kumar B.G., Gustavo Carneiro, and Ian Reid. *Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue*, pages 740–756. 2016.

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[9] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[10] Ankur Handa, Michael Bloesch, Viorica Pătrăucean, Simon Stent, John McCormac, and Andrew Davison. gvnn: Neural network library for geometric computer vision. In *ECCV*, pages 67–82. Springer, 2016.

[11] Zhiwu Huang, Chengde Wan, Thomas Probst, and Luc Van Gool. Deep learning on lie groups for skeleton-based action recognition. *CVPR*, pages 1243–1252, 2017.

[12] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *CVPR*, 2016.

[13] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and koray kavukcuoglu. Spatial transformer networks. In *NIPS*, pages 2017–2025. Curran Associates, Inc., 2015.

[14] A Jaegle, S Phillips, and K Daniilidis. Fast, robust, continuous monocular egomotion computation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 773–780, 2016.

[15] Michael Janner, Jiajun Wu, Tejas Kulkarni, Ilker Yildirim, and Joshua B Tenenbaum. Self-Supervised Intrinsic Image Decomposition. In *NIPS*, 2017.

[16] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *IROS*, 2013.

[17] D. H. Kim and J. H. Kim. Effective background model-based rgb-d dense visual odometry in a dynamic environment. *IEEE Transactions on Robotics*, 32(6):1565–1573, 2016.

[18] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[19] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. UnDeepVO: Monocular visual odometry through unsupervised deep learning. *arXiv*, 2017.

[20] Shile Li and Dongheui Lee. Rgb-d slam in dynamic environments using static point weighting. *IEEE Robotics and Automation Letters*, 2(4):2263–2270, 2017.

[21] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *CVPR*, June 2018.

[22] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. Relative camera pose estimation using convolutional neural networks. *arXiv*, 2017.

[23] R. Mur-Artal and J. D. Tards. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[24] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[25] Tomás Pajdla and Jiří Matas, editors. *The Least-Squares Error for Structure from Infinitesimal Motion*, 2004.

[26] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. 2016.

[27] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *AAAI*, 2017.

[28] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351, pages 234–241, 2015.

[29] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *TPAMI*, 39(4), Apr. 2017.

[30] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *CVPR*, 2017.

[31] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, Oct. 2012.

[32] Yuxiang Sun, Ming Liu, and Max Q.-H. Meng. Improving rgb-d slam in dynamic environments: A motion removal approach. *Robotics and Autonomous Systems*, 89:110 – 122, 2017.

[33] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. *CVPR*, 2017.

[34] H. Tung, H. Wei, E. Yumer, and K. Fragkiadaki. Self-supervised learning of motion capture. In *NIPS*, 2017.

[35] Hsiao-Yu Fish Tung, Adam W. Harley, William Seto, and Katerina Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. *ICCV*, 2017.

[36] Andreas Veit and Serge J. Belongie. Convolutional networks with adaptive computation graphs. *CoRR*, 2017.

[37] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *CoRR*, 2017.

[38] S. Wang, R. Clark, H. Wen, and N. Trigoni. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *ICRA*, pages 2043–2050, 2017.

[39] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *CVPR*, pages 520–526, Jun 1997.

[40] Thomas Whelan, Stefan Leutenegger, Renato Salas Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. In *Proceedings of Robotics: Science and Systems*, 2015.

[41] Thomas Whelan, John McDonald, Michael Kaess, Maurice Fallon, Hordur Johannsson, and John J. Leonard. Kintinuous: Spatially extended kinectfusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, July 2012.

[42] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *ECCV*, pages 817–833, 2018.

[43] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

[44] Tinghui Zhou, Matthew Brown, Noah Snavely, and David Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.