Self-supervised Object Motion and Depth Estimation from Video

Qi Dai[1,3] Vaishakh Patil[1] Simon Hecker[1] Dengxin Dai[1] Luc Van Gool[1,2] Konrad Schindler[1,3]

[1]Computer Vision Lab, ETH Zurich [2]VISICS, ESAT/PSI, KU Leuven
[3]Institute of Geodesy and Photogrammetry, ETH Zurich

daiq@ethz.ch {patil, heckers, dai, vangool}@vision.ee.ethz.ch schindler@geod.baug.ethz.ch

Abstract

We present a self-supervised learning framework to estimate the individual object motion and monocular depth from video. We model the object motion as a 6 degree-of-freedom rigid-body transformation. The instance segmentation mask is leveraged to introduce the information of object. Compared with methods which predict pixel-wise optical flow map to model the motion, our approach significantly reduces the number of values to be estimated. Furthermore, our system eliminates the scale ambiguity of predictions, through employing the pre-computed camera ego-motion and the left-right photometric consistency. Experiments on KITTI driving dataset demonstrate our system is capable to capture the object motion without external annotation, and contribute to the depth prediction in dynamic area. Our system outperforms earlier self-supervised approaches in terms of 3D scene flow prediction, and produces comparable results on optical flow estimation.

1. Introduction

Imagining a driving scenario in real world. The driver may encounter many dynamic objects (*e.g.* moving vehicles). The knowledge of their movements is of vital importance for the driving safety. We aim to solve the motion of individual object from video in the context of autonomous driving (*e.g.* the video is taken by a camera installed on a moving car). However, due to the entanglement of object movement and camera ego-motion, it is difficult to estimate the individual object motion from video.

This difficulty can be tackled by introducing the information of surrounding structure, *e.g.* a per-pixel depth map. Depth estimation from image is a fundamental problem in computer vision. Recently the view-synthesis based approach provides a self-supervised learning framework for depth estimation, without supervision of depth annotation. Strong baselines of depth prediction have been established in [9, 20, 36], most of which jointly train a depth and a pose network (for predicting camera ego-motion).

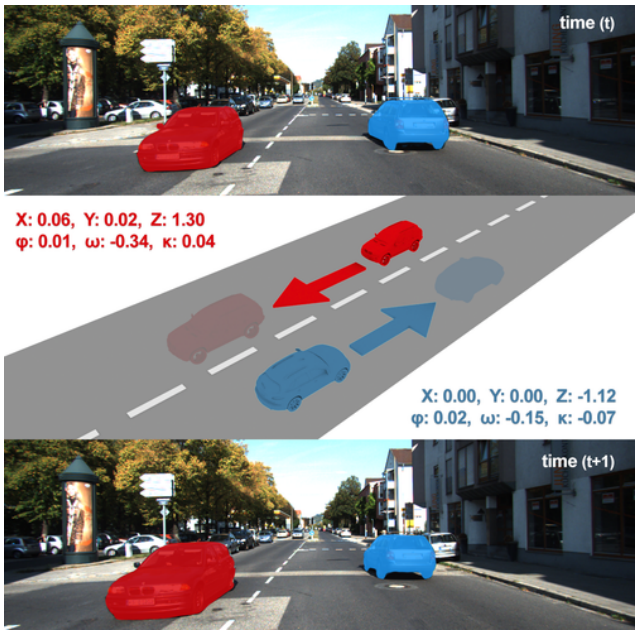

Figure 1. Our system predicts individual object motion by leveraging the instance-level segmentation mask. For each segmented object, three translation (X, Y, Z) and three rotation elements (φ, ω, κ) are predicted. The prediction describes the object movement during the capture of two consecutive frames (I_t and I_{t+1}), within the camera coordinate system of I_t. The unit of translation and rotation elements are meter and degree respectively. More details about the coordinate system and rotation order are provided in supplementary materials.

The depth and camera ego-motion can only explain the pixel displacement in static background. To model the dynamic object motion, Yin *et al*. [35] proposes to jointly estimate the optical flow. Similar pipeline is used in EPC++ [20] and achieves impressive performance.

In this paper, we propose a self-supervised learning framework for estimating the individual object motion and the monocular depth from video. Our first contribution is to model and predict the object motion in the form of 6 *dof*

(degree-of-freedom) rigid-body transform. Previous self-supervised approaches use dense 2D optical flow, or 3D scene flow [2] to model the motion, meaning a pixel-wise flow map is predicted. By contrast, our approach predicts a 6 *dof* rota-translation for the motion of individual object. The number of values to be estimated is significantly reduced from a pixel-wise prediction to 6 scalars per instance.

Our second contribution is to solve the scale ambiguity in the view-synthsis based framework, using the pre-computed absolute camera ego-motion and the left-right photometric loss. Self-supervised learning framework based on view synthesis suffers from scale ambiguity, meaning the predicted motion and depth are only up-to-scale. We solve this problem by estimating the absolute camera ego-motion in advance, and synthesizing a view together with the depth prediction. In this process the scale information is introduced. We also impose the left-right photometric loss which encodes scale information from the stereo image pair.

We perform evaluations of our framework on KITTI dataset. The result manifests the effectiveness of our system to predict individual object motion. Our system outperforms other self-supervised approaches in scene flow prediction, and improve the depth prediction in dynamic area of the image. We also produce optical flow prediction which is comparable to results of earlier approaches.

2. Related work

Our system is developed to solve the individual object motion from video, and provide monocular depth estimation. In this section we firstly present works related to depth estimation from image. Then some methods which address the object motion are introduced.

Supervised Depth Estimation The depth estimation is formulated as a regression problem in most supervised approaches, where the difference between the predicted depth and its ground truth is minimized. The manually defined feature is used in early work. Saxena *et al.* [27] propose to estimate the single-view depth by training Markov random field(MRF) with hand-crafted features. Liu *et al.* [19] integrate semantic labels with MRF learning. Ladicky *et al.* [16] improve the performance by combining the semantic labeling with the depth estimation.

Deep convolutional neural network (CNN) is good at extracting features and inspires many other methods. Eigen *et al.* [5] propose a CNN architecture to produce dense depth map. Based on this architecture, many variants have been proposed to improve the performance. Li *et al.* [18] improve the estimation accuracy by combining the CNNs with the conditional random filed(CRF), while Laina *et al.* [17] use the more robust *Huber loss* as the loss function. Patil *et al*. [29] produce a more accurate depth estimation by exploiting spatiotemporal structures of depth across frames.

Self-supervised Depth Estimation However, it is expensive and time-consuming to collect large amounts of depth ground truth. One alternative is to build a self-supervised learning framework where the network is supervised by the image reconstruction loss. This loss is a function of depth prediction. Zhou *et al*. [36] proposed to jointly train two networks for estimating dense depth and camera ego-motion. The image is synthesized from the network output, following the traditional *Structure-from-motion* procedure. Various constraints have been introduced to improve the performance, like the surface normal consistency [34], the edge consistency [33] or the temporal depth consistency [22]. Godard *et al*. [10] achieved a significant improvement by compensating for image occlusion .

Besides estimating depth from the monocular video, [10, 20] have proposed to synthesize stereo image pairs for depth estimation. Here the stereo image pairs have been calibrated in advance, the pose network is thus no longer necessary. Depth prediction from this set-up is free of scale ambiguity issue, since the scale information is introduced from the calibrated stereo image pairs.

Compensation for Object Motion Most self-supervised monocular depth estimation approaches are subject to rigid scene assumption: scene captured by the video are assumed to be rigid. This assumption is generally not true in autonomous driving scenario, where many moving objects are presented.

The object motion can be solved by introducing the optical flow map. Yin *et al*. [35] proposed to estimate the residual flow on top of the rigid flow, which is computed from the predicted depth and camera ego-motion. This residual flow can only correct for small error but generally fail for big pixel displacement, *e.g.* when the object is moving fast. Luo *et al*. [20] proposed to jointly train networks for depth, camera ego-motion, optical flow and motion segmentation, with enforcing the consistency between each prediction. In [25] a similar architecture is adopted, while the system is trained in a competitive collaboration manner. Both approaches have achieved *SoTA* performance on KITTI dataset.

Beyond the scope of self-supervised learning, the estimation of optical flow has been addressed through end-to-end deep regression based methods [4, 12]. PWC-Net [28] further improves the efficiency by integrating the pyramid processing and cost volume into their system. Besides optical flow, scene flow has been introduced to solve the object motion. Scene flow [30] describes the 3D motion of a point. [31, 32, 24] estimate the scene flow by fitting a piece-wise rigid representations of motion. They decompose the scene into small rigidly moving plane and solve their motion by enforcing some constraints, like appearance or constant velocity consistency in [31]. DRISF [21] formulates the scene flow estimation as energy minimization in a deep structured

model, which can be solved efficiently and outperforms all other approaches.

In this work we try to solve the object motion by modelling it as a rigid-body transform. Similar idea is proposed in [3], where the pre-computed instance segmentation masks are utilized for individual object-motion prediction. We proposes a self-supervised learning framework inspired from their work, with significant modifications summarized as following: (1) Our system estimates the combined transformation, encapsulating both the camera ego-motion and the object motion. By contrast, Casser *et al*. [3] estimate the object motion on top of the camera ego-motion, predicted by the Pose-net. Thus the accuracy of their object motion prediction is dependent on the performance of their Pose-net. (2) Our system predicts absolute object motion which can be used to transform the object in 3D space directly. By contrast, Casser *et al*. only predict an up-to-scale object motion. This means the magnitude of object movement is missing in their prediction.

3. Method

Here we propose a framework for jointly training an object-motion network (ObjMotion-net) and a depth network (Depth-net). We firstly explain the view synthesis for dynamic objects, then provide an overview of our framework. Our networks are supervised by four losses, which are detailed in Sec. 3.3.

3.1. Theory of View Synthesis

View synthesis is to synthesize the target frame I_{tgt} from the source frame I_{src}. For each pixel p_{tgt} in I_{tgt}, its correspondence p_{src} in I_{src} is required. The photometric consistency between the synthesized view $\hat{I}_{tgt}$ and its reference I_{tgt} serves as the primary supervision in our system.

Synthesis for Static Area Suppose two consecutive frames from a video are given: the target frame I_{tgt} captured at time *t*, and the source frame I_{src} captured at time *t+1*. For pixel p_{tgt} in the static area of I_{tgt}, its correspondence p_{src} in I_{src} is computed from Eq. 1:

$$\begin{aligned} h(p_{src}) &\sim K T_{t\to s} X^t(p_{tgt}) \\ X^t(p_{tgt}) &= \hat{D}(p_{tgt}) K^{-1} h(p_{tgt}) \quad p_{tgt} \in S_0(I_{tgt}) \end{aligned} \tag{1}$$

where $h(p)$ denotes the homogeneous pixel coordinates, K is the camera intrinsics, $T_{t\to s}$ is the camera ego-motion for the reference system C_{tgt} and C_{src} (see Fig. 2), $X^t(p_{tgt})$ is the projected 3D point of p_{tgt} in the reference system C_{tgt}, $\hat{D}(p_{tgt})$ denotes the depth prediction scalar at p_{tgt}, $S_0(I_{tgt})$ refers to the static area of I_{tgt}.

Synthesis for Dynamic Area Pixel correspondence for dynamic object is computed from Eq. 2. Here the 3D point

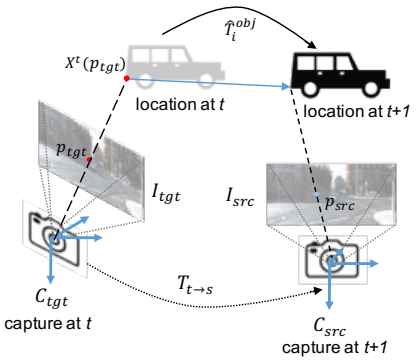

Figure 2. View synthesis for dynamic object. Firstly p_{tgt} is projected into the target camera reference system C_{tgt}, denoted as a 3D point $X^t(p_{tgt})$. This point is then transformed by $\hat{T}_i^{obj}$ (for object motion) and $T_{t\to s}$ (for camera ego-motion), and is finally projected onto I_{src} as the correspondence p_{src}.

$X^t(p_{tgt})$ is further transformed by a rigid-body transform $\hat{T}_i^{obj} \in \mathcal{SE}(3)$ (6 *dof*, 3 translations and 3 Euler angles). This process is illustrated in Fig. 2.

$$h(p_{src}) \sim K\, T_{t\to s}\, \hat{T}_i^{obj}\, X^t(p_{tgt}) \quad p_{tgt} \in S_i(I_{tgt}) \tag{2}$$

Here $S_i(I_{tgt})$ refers to pixels in the dynamic area of I_{tgt}, whose 3D motion is described as $\hat{T}_i^{obj}$. Suppose there are n moving objects in the scene, we estimate $\hat{T}_i^{obj}$ $(i = 1, \ldots, n)$ for each individual object. Then the target frame I_{tgt} is synthesized separately for static pixels according to Eq. 1, and for dynamic pixels according to Eq. 2.

Note we only focus on objects whose movement can be described by a rigid-body transform. These include cars, buses and trucks, etc. Objects like pedestrians are not considered since their movement is too complicated to be described by a rigid-body transform.

3.2. Framework Overview

Fig. 3 provides an overview of our framework. It illustrates how the image is synthesized from the network output: depth and object-motion for all instances in the scene. We distinguish between the static and dynamic area based on image segmentation mask. Acquired from the pre-trained Mask R-CNN [11] model, the segmentation mask highlights instances which are potentially, but not necessarily dynamic (e.g when vehicle stops for the traffic light).

The segmentation mask also distinguishes between different instances in the scene. We align the instance mask across time, and segment the temporal image sequence by this instance-aligned mask. One masked sequence example is shown as I_{t-1}, I_t and I_{t+1} in Fig. 3. This serves as the network input to predict the motion $\hat{T}_{t\to t-1}^{obj}$ and $\hat{T}_{t\to t+1}^{obj}$ for this specific object.

In implementation, the actual network prediction is the product of the camera ego-motion and the object-motion:

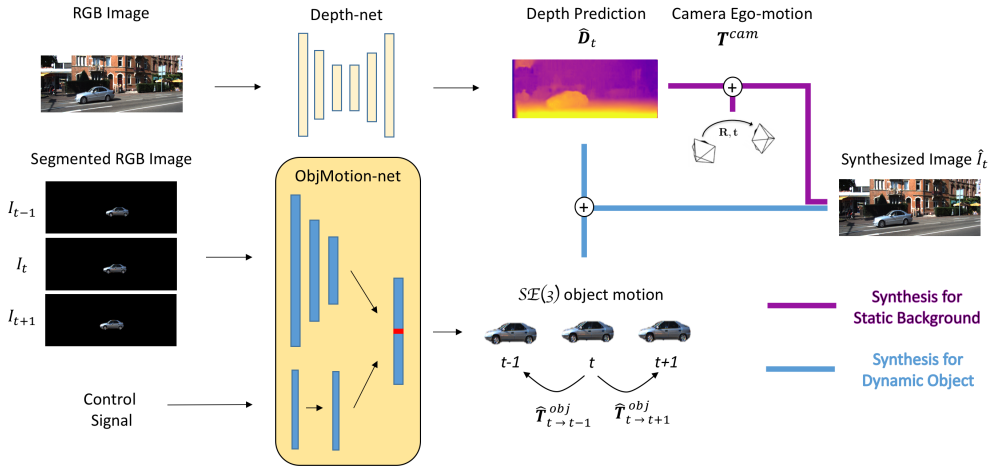

Figure 3. Framework overview. For view synthesis, each pixel is distinguished as either dynamic or static pixel. Dynamic pixel is synthesized from the individual object-motion and depth prediction, while static pixel is reconstructed from the depth and camera ego-motion. The camera ego-motion is pre-computed from the visual odometry library [8]. The distinguish of static/dynamic pixel is based on the segmentation mask, provided by Mask R-CNN [11].

$T^{cam}_{t\to t-1} \times \hat{T}^{obj}_{t\to t-1}$ and $T^{cam}_{t\to t+1} \times \hat{T}^{obj}_{t\to t+1}$. The object-motion can then be obtained based on the pre-computed camera ego-motion.

Object Motion Network The ObjMotion-net is designed to predict individual object movement. The idea is inspired by the Pose-net. Both networks take image sequence as input, and output 6 motion parameters. As shown in [36], Pose-net is capable to infer the camera ego-motion. This indicates Pose-net can conduct feature extraction and matching which are indispensable procedures for motion inference. We suppose ObjMotion-net, which adopts a similar architecture, also has the capability to extract and match features, and can infer the individual object motion based on these information.

The ObjMotion-net takes the masked image sequence (shown in Fig. 3) as input. All information irrelevant with the target object is excluded. Besides, one control signal is fed into the network. This control signal encodes the information of the object movement magnitude, which is missing in the monocular image sequence. The control signal $F_i^{t\to t+1}$ for the i-th object between time t to $t+1$ is computed as Eq. 3:

$$\begin{aligned} F_i^{t\to t+1} &= \bar{X}_i^{t+1} - \bar{X}_i^t \\ \bar{X}_i^m &= |\sum X_i^m(p)| \quad p \in S_i(I_m), m \in \{t, t+1\} \\ X_i^m(p) &= \hat{D}^m(p) K^{-1} h(p) \quad m \in \{t, t+1\} \end{aligned} \tag{3}$$

$F_i^{t\to t+1}$ is actually the vector from the 3D object center $\bar{X}_i^t$ to $\bar{X}_i^{t+1}$. Here $|\cdot|$ refers to the mean operator. $X_i^m(p)$ is the projected 3D point of pixel p in the reference system C_m, while $S_i(I_m)$ is the i-th object area of image I_m, with $m \in \{t, t+1\}$ denoting the image capture time.

3.3. Loss Function

Our framework employs four loss terms: photometric loss L_p, left-right photometric loss L_{lrp}, control constraint loss L_{cc} and disparity smoothness loss L_{disp}.

The photometric loss L_p penalizes the photometric inconsistency between the synthesized view $\hat{I}$ and its reference view I. $\hat{I}$ is synthesized based on the prediction from Depth-net and ObjMotion-net, thus L_p provides gradient on both networks. We adopt a robust image similarity measurement SSIM for L_p as formulated in Eq. 4, with $\alpha = 0.85$. The depth is predicted and supervised at multi-scale level to overcome the gradient locality [36].

$$L_p = \alpha \frac{1 - SSIM(I, \hat{I})}{2} + (1-\alpha)\|I - \hat{I}\|_1 \tag{4}$$

Note we distinguish the static and dynamic area for the synthesized view $\hat{I}$ when we compute its photometric loss. Instead of averaging the per-pixel photometric difference over the whole image, we average the difference in static and dynamic area separately, and formulate the L_p by summing them. Separation of photometric loss can compensate the unbalance between the static and dynamic image area, and contribute to the training of network.

The left-right photometric loss L_{lrp} is imposed to solve the scale ambiguity of the monocular depth prediction. The direct output of our Depth-net is actually the disparity. It can be used to synthesize the left image from its right counterpart, or vice versa. L_{lrp} penalizes the photometric difference of the synthesized stereo images. This provides supervision to solve the scale ambiguity of disparity predictions.

During experiments we find the translation of object-motion tends to be predicted as small values. This issue can be fixed by imposing the control constraint loss L_{cc}. L_{cc} is the L1-norm of the difference between the predicted translation of object-motion $\hat{t}_i$ and the computed control signal F_i (defined in Eq. 3).

$$L_{cc} = \sum_i^n \|\hat{t}_i - F_i\|_1 \quad \hat{t}_i = [\hat{x}_i, \hat{y}_i, \hat{z}_i] \tag{5}$$

Here n is the number of instances appeared in the input image sequence. L_{cc} encodes the magnitude information of

the object movements. The issue of the small translation prediction can then be fixed.

The disparity smoothness loss L_{disp} is enforced to penalize a fluctuated disparity prediction. An edge-aware smoothness term is imposed as formualted in Eq. 6. Here the disparity smoothness ($\partial_x d$ and $\partial_y d$) is weighted by the exponential image gradient ($e^{\|-\partial_x I\|}$ and $e^{\|-\partial_y I\|}$). x and y refers to the gradient along the horizontal or vertical direction.

$$L_{disp} = |\partial_x d| e^{\|-\partial_x I\|} + |\partial_y d| e^{\|-\partial_y I\|} \tag{6}$$

Our final objective is a sum of all loss terms stated above, weighted by their corresponding weight:

$$\begin{aligned} L_{final} = \lambda_p \cdot L_p + \lambda_{lrp} \cdot L_{lrp} \\ + \lambda_{cc} \cdot L_{cc} + \lambda_{disp} \cdot L_{disp} \end{aligned} \tag{7}$$

4. Experiments

In this section, we firstly describe the implementation details, and demonstrate evaluation results on object motion, scene flow, optical flow and depth estimation. Experiments are mainly conducted on KITTI [7], a dataset provides driving scenes in real-world scenario. Synthetic dataset Virtual KITTI [6] is also used to test our system.

4.1. Implementation Details

Dataset and preprocessing KITTI raw dataset provides videos which cover various scenes. We resize all images into a fixed size 192×640, and format a temporal image sequence by concatenating I_{t-1}, I_t and I_{t+1} together. Scenes covered by the test set are excluded during training. Finally 23310 training samples and 2658 validation samples are formatted for training the depth network, where the Eigen split [5] is adopted to determine scenes to be excluded.

For the ObjMotion-net, we evaluate on the training split of KITTI flow 2015 dataset, which provides the ground truth for disparity, optical flow and scene flow. We format 40820 samples and 2070 samples for training and validation respectively. Besides the raw dataset, we format another training set from the test split of the multi-view extension of KITTI flow 2015 dataset. Scenes in this split contain more moving vehicles, which benefit the training of ObjMotion-net. There are 6512 training examples and 364 validation examples in this split.

The camera ego-motion is required for view synthesis. Instead of training a pose network, we use the Libviso2 [8], an library of visual odometry, to estimate the camera ego-motion. The segmentation mask for image is generated from the pre-trained Mask R-CNN model [11]. We segment objects which move rigidly (their movement can be described by a 6 *dof* rota-translation) in the scene, like car, bus or truck. The segmented mask is aligned across the temporal image sequence. This means the mask index for the same instance appeared on I_{t-1}, I_t and I_{t+1} are identical. More details of the alignment is given in supplementary materials.

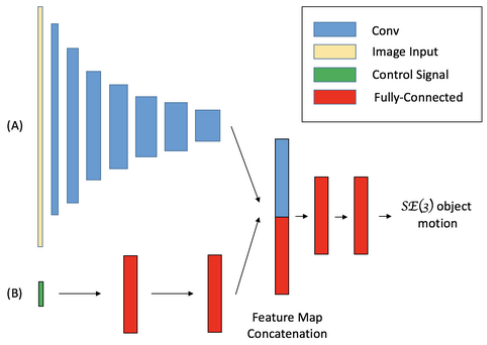

Figure 4. Structure of ObjMotion-net. The input of part A is the segmented I_{t-1}, I_t and I_{t+1}, concatenated along the color channel. Part B takes the control signal as input. Feature maps from part A and B are flatted and concatenated, then is convoluted through two fully-connected layer to obtain the object motion prediction.

Network Architecture Our system contains two subnetworks, the ObjMotion-net and the Depth-net. The ObjMotion-net is designed based on the pose network in [35]. As shown in Fig. 4, this structure constitutes two parts. Part A takes the concatenation of segmented RGB image as input, while part B takes the control signal as input. As explained in Sec. 3, the control signal is integrated to solve the ambiguity of object motion. We adopt batch normalization (BN) [13] and ReLUs for all convolutional layers. While we drop BN for fully-connected and the prediction layer.

For the Depth-net, we adopt the architecture in [35] as backbone. This structure consists of the encoder and the decoder part. The basic structure of ResNet50 is adopted for the encoder. While in decoder the combination of convolution and upsampling is used for upscaling the feature map. Skip connections between the encoder and the decoder are added to integrate global and local information.

4.2. Training Details

Our system is implemented using TensorFlow framework [1]. Color augmentation is performed on the fly. The network is optimized using Adam optimizer, with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively. Our system is trained on a single TitanXP GPU. A stage-wise training strategy is adopted, with training Depth-net alone at the beginning, and then jointly training Depth-net and ObjMotion-net.

Training Depth-net We firstly train the Depth-net, since an accurate control signal (necessary for the training of ObjMotion-net) requires accurate depth prediction. In the first stage the Depth-net is trained on the formatted

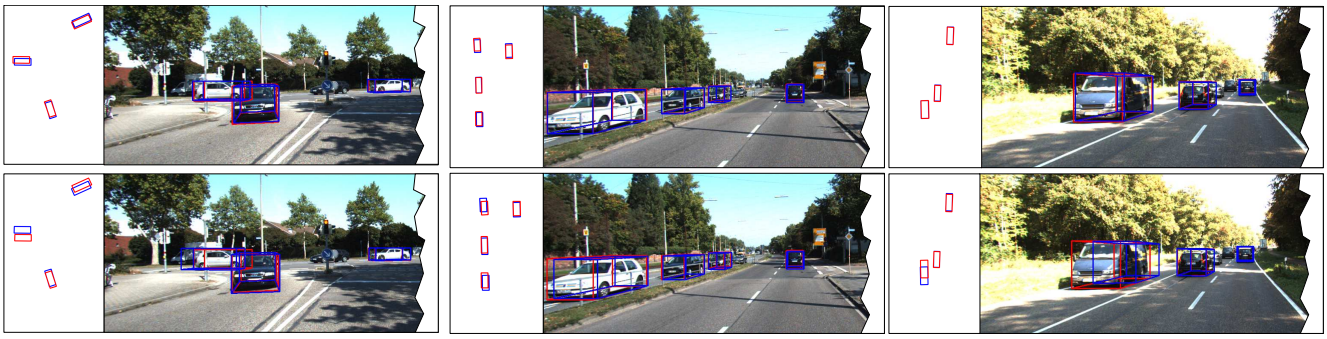

Figure 5. Visualization of Bird's View Box and 3D Bounding Box. Our results (top row) and results from GeoNet [35] (bottom row) are presented. The ground truth is in red while the prediction is in blue. Our predictions have a larger overlapping with the ground truth box.

KITTI raw dataset, supervised by the left-right photometric consistency L_{lrp} and disparity smoothness L_{disp}. Their weights are set to be $\lambda_{lrp} = 1.0$ and $\lambda_{disp} = 0.5$. The learning rate and the batch size are 0.0002 and 4 respectively. After training 300K iterations, the model is trained on Cityscapes dataset for 200K steps (for better generalization as indicated in [9]), and is fine-tuned on KITTI again for 100K steps. The learning rate is 0.0001 and $\lambda_{disp} = 50$ in this stage.

Training ObjMotion-net with Depth-net We then jointly optimize the Depth-net and ObjMotion-net. Besides the L_{lrp} and the L_{disp}, the photometric consistency L_p is imposed on the dynamic warped image. The object motion prediction is constrained by the control signal through imposing L_{cc}. The loss weights are set to be $\lambda_p = \lambda_{lrp} = 1.0$, $\lambda_{disp} = 50$, and $\lambda_{cc} = 0.1$. The learning rates for Depth-net and ObjMotion-net are 0.0001 and 0.0002 respectively, and the batch size is 2.

After training on the KITTI raw dataset for 180K iteration, we fine-tune the ObjMotion-net on the test split of KITTI flow dataset 2015, with fixing the parameters of Depth-net. All hyper-parameters remain the same. The ObjMotion-net is trained for 100K iterations in this stage.

4.3. Individual Object Motion Evaluation

Our system predicts individual object motion in 3D space. To demonstrate the effectiveness of our system, we present the Intersection over Union (IoU) of the bird's view box and 3D bounding box. Take the example of 3D bounding box: the 3D bounding box for object i at time t, denoted as B_t^i, is transformed by its object motion prediction $\hat{T}_{t+1}^i$. Then the predicted location of box at time $t+1$, $\hat{B}_{t+1}^i$ is obtained. We compute the IoU between $\hat{B}_{t+1}^i$ and its ground truth B_{t+1}^i. This IoU can serve as a performance indicator for our ObjMotion-net.

The ground truth of bird's view box and 3D bounding box is provided in KITTI tracking dataset [7]. We evaluate on 80 temporal image pairs (I_t and I_{t+1}), which are included in both the training split of KITTI tracking and flow dataset. Objects in scenes are segmented by pre-trained Mask R-CNN model [11]. We do not compute the IoU for objects that the Mask R-CNN fails to segment. In total 204 objects are selected from these 80 image pairs.

Method	Bird View	3D Box
Rigid	45.02%	43.67 %
GeoNet [35]	57.54%	56.00%
Ours	**70.69 %**	**69.10 %**

Table 1. Average Intersection over Union of bird's view box and 3D bounding box. The number is computed by averaging instance-level IoU of all identified (segmented) objects in test images. Method *Rigid* shows the averaged IoU of bounding box between time t and $t+1$, where the bounding box is not transformed by any rota-translation.

In Fig. 5, we compare our qualitative results with results from GeoNet [35]. It can be seen that our predicted bounding boxes have a higher overlapping with the ground truth. Quantitative results in Table 1 also show our system has a higher average IoU, with 70.69% for bird's view box and 69.10% for 3D bounding box. This demonstrates the effectiveness of our system to predict the individual object motion.

Different with our system, traditional self-supervised approaches like GeoNet does not predict a 6 *dof* rota-translation. Nevertheless, these approaches provide disparity and optical flow predictions, from which we can compute the pixel-wise scene flow vector in 3D space. The transformation of 3D bounding box (also the bird's view box) from time t to $t+1$ can then be inferred from the scene flow prediction.

4.4. Scene Flow Evaluation

We present the evaluation results on 3D scene flow in Table 2. Scene flow vector encapsulates the information of object motion. The evaluation conducts on the predicted disparity for two consecutive frames: $\hat{D}(I_t)$ and $\hat{D}(I_{t+1})$, and the 2D optical flow map $\hat{F}_{t\to t+1}$. In our system, we do not have a component to explicitly predict the pixel-wise

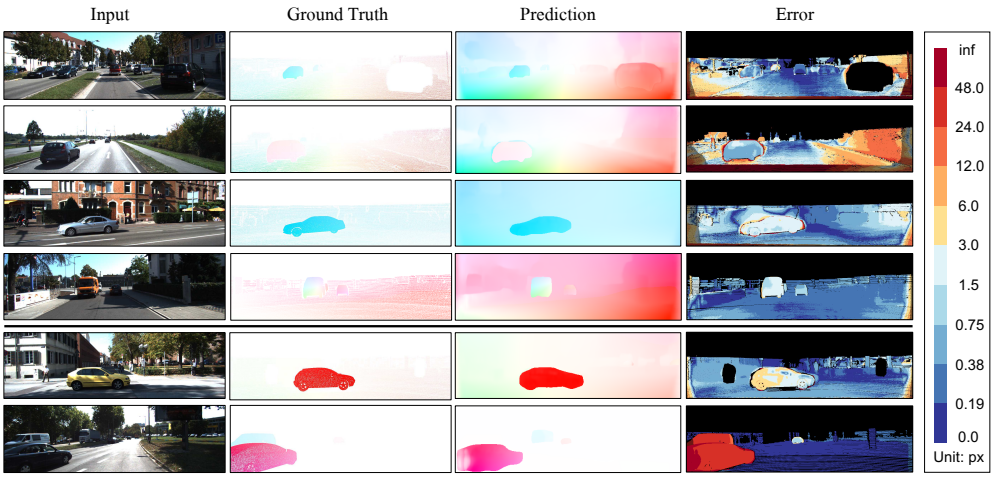

Figure 6. Visualization of optical flow on images of KITTI flow 2015 training split. The error magnitude of the last column is encoded into different colors according to the legend at the right-hand side. Basically good pixel is in blue while bad pixel is in orange/red. The top four rows show some successed samples, where the error of most pixels in dynamic region are below 3px. The bottom two rows show two imperfect examples, with bad pixel percentages are relatively high.

Method	bg	fg	all
GeoNet [35]	66.8%	90.4%	70.7%
Mono + Geo	39.4%	70.9%	44.7%
EPC++ [20]	$>$ 22.8%	$>$ 70.4%	$>$ 60.3%
Ours	35.3%	**58.7%**	**39.3%**

Table 2. Bad pixel percentage of scene flow prediction, evaluated on the training split of KITTI flow 2015. *Fg* and *bg* are short for foreground (dynamic area) and background (static area) respectively. For the evaluation of *Mono + Geo*, the disparity from Monodepth2 [10] and the flow prediction from GeoNet [35] are used.

optical flow. The optical flow prediction is obtained through view synthesis with taking the object motion into account, as illustrated in Fig. 3 and in Sec. 3. This is fundamentally different with all other approaches, where the optical flow map is a direct network output.

The evaluation metrics in KITTI scene flow benchmark is the bad pixel percentage. A pixel is considered as bad if its prediction error $\geq$ 3px or $\geq$ 5%. Besides, the segmentation masks for moving objects are provided. This makes it possible to evaluate within dynamic region (*fg* in Table 2), which in our case is the *Region-of-Interest*.

Our results outperform all other three methods. In foreground we achieve a percentage of 58.7%, compared with 70.9% from Mono+Geo, and 90.4% from GeoNet. EPC++ [20] does not provide the scene flow evaluation, so we can only provide a lower bound based on their disparity and optical flow results. It is important to note that both GeoNet and EPC++ predict a pixel-wise optical flow map. By contrast, our system models the individual object motion using a 6 *dof* rota-translation. Our system significantly reduces the number of values to be estimated, while still yields a better scene flow evaluation result.

4.5. Optical Flow Evaluation

Optical flow describes dense pixel correspondence between two images. The 3D motion of object is also reflected in the 2D optical flow map. We provide quantitative (Table 3) and qualitative (Fig. 6) results of optical flow prediction. We evaluate on the training split of KITTI flow 2015 dataset. The average End-point error (EPE) is reported. EPE is the L2-norm of the optical flow prediction error, which is a 2-dimensional vector representing the error along horizontal and vertical direction.

Method	End-point Error	
	Noc	All
GeoNet [35]	8.05	10.81
DSTFlow [26]	6.96	16.79
SIGNet [23]	7.66	13.91
Janai *et al.* [14]	-	7.04
EPC++ [20]	-	**5.43**
CC-uft [25]	-	5.66
Ours	**5.14**	7.00

Table 3. Average End-point error (EPE) of optical flow prediction on the training split of KITTI flow 2015 dataset. Noc is short for evaluation in Non-occluded area, while All refers to evaluation over the whole image.

Our results achieve comparable performance without a direct optical flow output. As shown in Table 3, the EPE in non-occluded area is 5.14, compared with 8.05 in GeoNet and 6.96 in DSTFlow. In terms of the EPE over all region, our system achieves a 7.00 error, which is inferior to EPC++ (5.43) and CC-uft (5.66).

We provide the visualization of optical flow prediction and error in Fig. 6. Examples at the top four rows show the effectiveness of our system, where the error of most pixels in dynamic region are low. Example at the fifth row shows one typical imperfect case: good and bad pixels are half-and-half in dynamic region, and the error magnitude of bad pixel is small. We argue this imperfection results from small bias in view synthesis. As mentioned before, our flow prediction is obtained from view synthesis. Any small error in camera intrinsics, depth or object motion may result in an error bigger than the bad pixel threshold (3px). Nevertheless, our system can still capture the general motion of objects, since most bad pixels in this example are below 6px, which is barely over the threshold.

Example at the bottom row of Fig. 6 shows the other im-

Method	Train	Lower the better				Higher the better		
		Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Zhou [36]	M	0.183	1.595	6.709	0.270	73.4%	90.2%	95.9%
GeoNet [35]	M	0.153	1.328	5.737	0.232	80.2%	93.4%	97.2%
EPC++ [20]	S	0.127	0.936	5.008	0.209	84.1%	94.6%	**97.9%**
Monodepth [9]	S	0.124	1.076	5.311	0.219	84.7%	94.2%	97.3%
Monodepth2 [10]	S	**0.106**	**0.818**	**4.750**	**0.196**	**87.4%**	**95.3%**	**97.9%**
Ours	S	0.123	0.921	5.050	0.205	85.4%	94.9%	97.7%

Table 4. Self-supervised monocular depth results evaluated on the Eigen split of KITTI. The best results are in **bold**, the second best are underlined. Here M and S in *Train* are short for monocular or stereo supervision during training.

Method	Bad Pixel Percentage		
	bg	fg	all
Monodepth2 [10]	**18.60%**	44.47 %	22.50%
EPC++ [20]	22.76%	26.63%	23.84 %
Ours	22.84%	**18.33 %**	**22.15%**

Table 5. Bad pixel percentage of disparity prediction, evaluated on KITTI stereo split.

perfect case: all pixels in dynamic region are bad pixels, and the error magnitude is big. This usually happens when only part of the object appears in the image. The incompleteness of object appearance makes it difficult for ObjMotion-net to predict the motion.

4.6. Monocular Depth Evaluation

We present depth evaluation on KITTI dataset using two different test splits: Eigen split in Table 4, and KITTI stereo split in Table 5. The ground truth of Eigen split is obtained by projecting Velodyne laser points onto the image plane. KITTI stereo split provides 200 high-quality disparity maps in the training split of scene flow 2015 dataset.

We compare our depth results with other self-supervised approaches in Table 4. Our result achieves a comparable performance with EPC++ [20], but is inferior to Monodepth2 [10]. In Table 5 we present the bad pixel percentage of disparity prediction on KITTI stereo split. Our results outperform Monodepth2 and EPC++ in terms of foreground and overall bad pixel percentage. We achieve a lower value of 18.33% in foreground, which is better than percentage of Monodepth2 (44.47%) and EPC++ (26.63%). This suggests our ObjMotion-net can contribute to the depth prediction for dynamic objects.

4.7. Evaluation on Virtual KITTI Dataset

Virtual KITTI [6] is a synthetic video dataset. It contains 50 monocular videos, varying in their scenes, weather conditions (fog, rain, *etc.*) and imaging conditions (camera orientation). We formulate a test split with 200 images, by randomly sampling 4 images from each video.

We present the average end-point error of the optical flow prediction in the foreground (dynamic) area of the image in Table 6. We do not fine-tune our model on virtual KITTI dataset. Our system achieves a 12.90 error compared with 13.23 of GeoNet. However, our performance is inferior to the results of [15]. Some qualitative results are shown in Fig. 7. Our system produces visually similar results with the ground truth. This shows our system is able to solve the general object motion in virtual KITTI, but performs poorly on capturing pixel-level displacement.

Method	Foreground EPE
GeoNet [35]	13.23
Janai *et al*. [14]	**7.55**
Ours	12.90

Table 6. Foreground EPE of optical flow prediction on Virtual KITTI dataset.

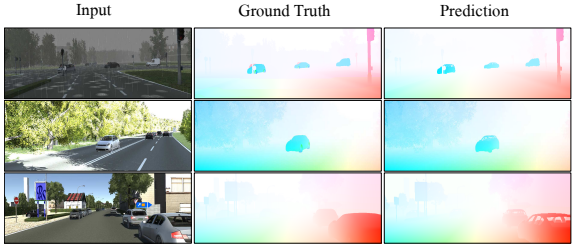

Figure 7. Visualization of optical flow on Virtual KITTI dataset.

5. Conclusion

We have presented a self-supervised learning framework for individual object motion inference and depth estimation. The object motion is modelled as a rigid-body transformation. Our system is able to infer the object motion from video. This contributes to scene flow and optical flow prediction, and improve the depth estimation in dynamic area of the scene.

It would be interesting to explore the following questions in future: 1) Integrate scale information from other sources. In our system the scale information of object motion is extracted from the depth prediction (the *control signal*). It would be difficult to apply our system in case where the depth prediction is not reliable. In future we can try to integrate scale information from other sources, like stereo image pairs, or sparse depth *ground truth* from Lidar. 2) Estimate the motion of pedestrians. Currently we focus on the object motion which can be described by a rigid-body transformation. While non-rigid motion, like the movement of pedestrians, is also common in driving scenario. We may dissect pedestrians into smaller parts which is moving rigidly, or try to model its motion in a different way.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Zhe Cao, Abhishek Kar, Christian Häne, and Jitendra Malik. Learning independent object motion from unlabelled stereoscopic videos. 2019.

[3] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. *arXiv preprint arXiv:1811.06152*, 2018.

[4] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[5] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

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

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012.

[8] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV)*, 2011.

[9] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[10] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation. *arXiv preprint arXiv:1806.01260*, 2018.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[14] Joel Janai, Fatma Guney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 690–706, 2018.

[15] Joel Janai, Fatma G"uney, Anurag Ranjan, Michael J. Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *European Conference on Computer Vision (ECCV)*, volume Lecture Notes in Computer Science, vol 11220, pages 713–731. Springer, Cham, Sept. 2018.

[16] Lubor Ladicky, Jianbo Shi, and Marc Pollefeys. Pulling things out of perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–96, 2014.

[17] 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.

[18] Bo Li, Chunhua Shen, Yuchao Dai, Anton Van Den Hengel, and Mingyi He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1119–1127, 2015.

[19] Beyang Liu, Stephen Gould, and Daphne Koller. Single image depth estimation from predicted semantic labels. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1253–1260. IEEE, 2010.

[20] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *arXiv preprint arXiv:1810.06125*, 2018.

[21] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3614–3622, 2019.

[22] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, 2018.

[23] Yue Meng, Yongxi Lu, Aman Raj, Samuel Sunarjo, Rui Guo, Tara Javidi, Gaurav Bansal, and Dinesh Bharadia. Signet: Semantic instance aided unsupervised 3d geometry perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9810–9820, 2019.

[24] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015.

[25] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12240–12249, 2019.

[26] Zhe Ren, Junchi Yan, Bingbing Ni, Bin Liu, Xiaokang Yang, and Hongyuan Zha. Unsupervised deep learning for optical flow estimation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[27] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in*

neural information processing systems, pages 1161–1168, 2006.

[28] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2018.

[29] Patil Vaishakh, Gansbeke Wouter Van, Dai Dengxin, and Gool Luc Van. Dont forget the past: Recurrent depth estimation from monocular video. 2020.

[30] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 722–729. IEEE, 1999.

[31] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1377–1384, 2013.

[32] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *European Conference on Computer Vision*, pages 756–771. Springer, 2014.

[33] Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. Lego: Learning edge with geometry all at once by watching videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 225–234, 2018.

[34] Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. *arXiv preprint arXiv:1711.03665*, 2017.

[35] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018.

[36] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017.