Motion-based Object Segmentation based on Dense RGB-D Scene Flow

Lin Shao, Parth Shah∗, Vikranth Dwaracherla∗, and Jeannette Bohg

Abstract—Given two consecutive RGB-D images, we propose a model that estimates a dense 3D motion field, also known as scene flow. We take advantage of the fact that in robot manipulation scenarios, scenes often consist of a set of rigidly moving objects. Our model jointly estimates (i) the segmentation of the scene into an unknown but finite number of objects, (ii) the motion trajectories of these objects and (iii) the object scene flow. We employ an hourglass, deep neural network architecture. In the encoding stage, the RGB and depth images undergo spatial compression and correlation. In the decoding stage, the model outputs three images containing a per-pixel estimate of the corresponding object center as well as object translation and rotation. This forms the basis for inferring the object segmentation and final object scene flow. To evaluate our model, we generated a new and challenging, large-scale, synthetic dataset that is specifically targeted at robotic manipulation: It contains a large number of scenes with a very diverse set of simultaneously moving 3D objects and is recorded with a commonly-used RGB-D camera. In quantitative experiments, we show that we significantly outperform state-of-the-art scene flow and motion-segmentation methods. In qualitative experiments, we show how our learned model transfers to challenging real-world scenes, visually generating significantly better results than existing methods.

I. INTRODUCTION

Semantic and functional scene understanding is a crucial capability of manipulation robots. In the Computer Vision community, this challenging problem is often approached given only a single image. However, a robot is able to physically interact with the environment and thereby autonomously induce motion in the scene. This motion creates a rich, visual sensory signal that would otherwise not be present, thus facilitating better scene understanding. The benefit for perception that arises from interaction is often referred to as Interactive Perception (IP) [1]. In this paper, we are providing the robot with a model to process the visual effect of its interaction. Given two consecutive RGB-D images, we are interested in estimating a dense 3D motion field of the environment, also known as scene flow. We show how this result helps to segment the finite, but unknown number of moving objects in the scene. This can provide input to tasks such as for example grasp (re-)planning or 3D object reconstruction. It can also help to learn the dynamic parameters of a scene.

We propose a model that takes advantage of the fact that in a common household scenario, scenes often consist of a set of rigidly moving objects. Our model jointly estimates (i) the segmentation of a scene into a finite number of rigidly moving object, (ii) the motion trajectories of these objects and (iii) the resulting object scene flow as defined by Menze and Geiger [25]. We propose to use a deep neural network architecture that takes as input a pair of consecutive RGB-D images. See Fig. 1 for an overview of the approach. In a first stage, features are extracted from each of the four input images. The RGB features are then correlated and the resulting values are used to weight the feature encoding of the depth data. Intuitively, this favors correspondences between points in the depth data that also have a strong similarity in the RGB images. The result is then decoded to produce three images containing the object positions, their translation, and their rotation. From this, we can infer the object scene flow and segmentation.

As will be detailed in the next section, our work differs from previous approaches in scene flow estimation in two aspects. First, we propose a learning-based method that mitigates the limitation of traditional methods in challenging situations of insufficient texture, occlusions and large displacements. Second, our model assumes input from RGB-D cameras which are the most commonly used cameras on manipulation robots and for which no data set or learning-based method has yet been proposed.

Summarizing, our primary contributions are: (1) generating a challenging, large-scale dataset for scene flow estimation with ground-truth annotated RGB-D images, (2) treating rotational symmetry of objects in scene flow prediction, (3) estimating object scene flow with a deep neural network architecture, and (4) predicting rigid body transformations
to segment a finite, but unknown number of moving objects.

II. RELATED WORK

Estimating scene flow has a long-standing history in the research community with Vedula et al. [31] coining the term. In this section, we briefly review the most recent approaches that are related to our work in terms of several aspects: input sensor, data sets, learning-based methods and motion segmentation.

A. Scene Flow based on RGB-D or Stereo Images

Gottfried et al. [12] were the first to use an RGB-D sensor for scene flow estimation. Their work also addresses the necessary calibration process. Herbst et al. [17] generalize the two-frame variational optical flow algorithm (2D) to scene flow (3D). The resulting dense scene flow was then used for rigid motion segmentation. Jaimez et al. [19] presented the first real-time method for computing dense scene flow from RGB-D images. Their method is based on a variational formulation that imposes brightness and geometric consistencies. The minimization problem is efficiently solved with a GPU and a primal-dual algorithm. Vogel et al. [32] were the first to propose the estimation of piecewise rigid scene flow were oversegmentation into superpixels constrains the scene flow estimation. The authors obtain a new level of accuracy that may run in real-time. Inspired by this work, Golyani et al. [11] propose a multi-frame scene flow approach which jointly optimizes the consistency of the patch appearances and their local motions from RGB-D image sequences. However, the reliance on bottom up cues for segmentation may lead to oversegmentation of objects. Menze and Geiger [25] defined object scene flow as the 3D motion associated with a set of pixels that constitute an object. The authors encourage superpixels in the same region to have similar 3D motion. The inference process is computationally very expensive, taking 2-50 minutes per image pair.

Although traditional methods have achieved excellent performance in RGB-D based scene flow estimation, they still suffer from problems like insufficient texture, occlusions and large displacements. Insufficient texture will result in huge errors during the matching process across different frames. Occlusion commonly exists in complex scenes, especially those with multiple moving objects. It violates the consistency assumption in traditional methods and leads to mismatch. Large displacements arise when an object is moving or rotating fast or under a low frame rate. Traditional approaches heavily rely on brightness constancy and smoothness within a small region and may lack the power to tackle such problems. These issues can be mitigated by using learning methods.

B. Datasets

Several large scale datasets exist for benchmarking and learning optical and scene flow. Different from our data set, they are all under a binocular setting with flow and disparity ground truth. KITTI [10] consists of 194 training and 195 test scenes recorded from a calibrated pair of cameras mounted on a car. Ground truth annotations are obtained by combining data from a 3D laser scanner with the car’s ego motion. Menze and Geiger [25] annotated the dynamic scenes with 3D CAD models for all moving vehicles and modified the dataset with 200 training scenes and 200 test scenes. KITTI contains valuable real world data. However, ground truth contains some approximation error. Mayer et al. [23] created a synthetic dataset called FlyingThings3D containing about 25000 stereo frames with ground truth scene flow annotations. We generated a dataset consisting of 35540 pairs of consecutive RGB-D images containing scenes with 1-30 simultaneously moving objects. In total, we use 31594 distinct object mesh models from ShapeNet [5] (see Section V).

C. Learning Based Flow Prediction

Learning-based method have up till now been mainly applied to optical flow estimation. Dosovitskiy et al. [8] posed this problem as a supervised learning problem and were the first to solve it with Convolutional Neural Networks (CNNs). They compare two architectures called FlownetS and FlownetC: a generic architecture and an architecture that includes a layer that correlates feature vectors at different image locations. These two FlowNets were tested on datasets like Sintel [4] and KITTI [10] achieving competitive accuracy at frame rates of 5-10 fps. Ilg et al. [18] extend FlowNet by developing a stacked architecture. It includes warping of the second images with intermediate optical flow. They also created a subnetwork specializing in small displacements resulting in state-of-the-art results while running at real-time. For learning-based scene flow estimation, Hadfield and Bowden [15] introduced a novel cost function. In this new formulation only a limited portion of the parameters from the entire pipeline are learned, leading to limited improvements. Mayer et al. [24] utilized a CNN to estimate scene flow based on stereo images. They embed a disparity estimation network called DispNet into FlowNet [8].

In this paper, we propose a deep architecture that estimates scene flow from two consecutive RGB-D frames. We evaluate our approach on a new and very difficult data set with multiple moving objects that may have uniform color and symmetries. The data set is also characterized by strong occlusions and large displacements. We show how our learning-based approach outperforms traditional methods on this data set by not relying on assumptions on smoothness and brightness constancy.

D. Motion-based Segmentation

Bohg et al. [11] extensively reviewed the variety of work towards motion-based segmentation within robotics. Here, we discuss a few representative examples. Many works use over-segmentations and connect superpixels over time using optical flow and/or clustering methods [2,13]. However, the reliance of bottom-up cues often results in remaining oversegmentation. Cheriyadat and Radke [6], Fushing et al. [9], Brox and Malik [3], Zografos et al. [35] formulated the problem as clustering of point trajectories across different
frames. The authors usually use spectral clustering method. Instead, Rahmati et al. [27] utilized multi-label graph cuts. Ji et al. [20] defined an unbalanced energy to model both, motion segmentation and point matching. Keuper et al. [22] formulate motion-based segmentation based on point trajectories as a minimum cost, multi-cut problem. The minimum cost multi-cut formulation allows for varying cluster sizes. We propose a model where each pixel directly predicts the center and trajectory of the object that it is associated with. We achieve accurate motion-based segmentation by clustering in this space. This in turn helps to refine the scene flow estimate.

III. PROBLEM FORMULATION & NOTATION

Given the manipulation and robotics motivation of this work, the input to the proposed model are two consecutive RGB-D images. We assume that the environment consists of a finite, but unknown, number of rigidly moving objects. The network outputs (i) a pixel-wise segmentation of each object, (ii) the rigid body motion of each objects, and (iii) the scene flow of each pixel in a reference frame.

More formally, let \( I^t \) and \( P^t \) denote an RGB image and a point cloud from a single RGB-D image at time \( t \). Time \( t \) and \( t−1 \) refer to the current and previous frames, respectively. To calculate scene flow of each point \( P^t_i \in P^t \) in a reference frame, we predict its 3D displacement by estimating its corresponding position \( P_{i,t−1} \) in the previous frame. This estimate is denoted by \( P_{i,t−1} \).

Let \( O \) denote the set of rigidly moving objects in the scene. The rigid body motion between two consecutive frames for \( O_k \) is described by an SE(3) transform consisting of a rotation \( R_k \) and translation \( T_k \). Our model directly outputs three images \( Q, T \) and \( X \) where each pixel contains an estimate of the rotation, translation and center of the object that the pixel belongs to. Therefore, if point \( P^t_i \) is generated by \( O_k \) then the correct value at the projected image coordinates \((u, v)\) in the respective output images will contain the ground truth rotation, translation and center of object \( O_k \).

We denote the rotation of a point \( P^t_i \) based on the axis-angle representation \( Q_k \) as \( r(P^t_i, Q_k) = R_kP^t_i \). Therefore, the corresponding point in frame \( t−1 \) can be computed by

\[
P_{i,t−1} = r(P^t_i - X_k, Q_k) + X_k + T_k
\]

with per-pixel scene flow \( S_t = P_{i,t−1} - P^t_i \). Note, that our model outputs an estimate of the ground truth variables \( Q_k, T_k \) and \( X_k \) which results in \( P_{i,t−1} \) instead of \( P^t_{i,t−1} \) and therefore only in an estimate \( S_i \) of the ground truth scene flow. During training, we aim to minimize the error between these estimates and the ground truth.

Let \( \xi_k = [X_k, X_k + T_k] \) denote the trajectory feature of an object \( O_k \). \( X_k \) and \( X_k + T_k \) are the object centers at frame \( t \) and \( t−1 \), respectively. Unless two objects are moving together, each \( \xi_k \) is unique per object. Therefore, we can use it as a cue for motion-based, object segmentation.

IV. TECHNICAL APPROACH

A. Rigid Motion and Object Scene Flow

The first stage of the proposed model, displayed in Fig. 2, consists of two Siamese networks that takes RGB images \( I^t−1, I^t \) and point clouds \( P^t−1, P^t \) as inputs, each with resolution \((W, H, 3)\). The pair of point clouds is fed into the first of these networks that outputs a new feature encoding denoted by \( P_f^{t−1} \) and \( P_f^t \), respectively. We use the VGG architecture [30] for this purpose. The shape of the output feature tensor is \((W/8, H/8, 64)\).

The pair of RGB images is fed into the second Siamese network that outputs a new feature encoding denoted by \( I_f^{t−1} \) and \( I_f^t \), respectively. We use the ResNet50 architecture and its weights for initialization [16]. The shape of the output feature tensor is \((W/8, H/8, 256)\).

The RGB image features are fed into a correlation layer similar to the one used in FlowNetC [8]. A high correlation between patches in consecutive RGB images indicates that they contain a projection of the same physical object part. This correlation layer parallels the brightness constancy assumption in traditional optical and scene flow methods.

Fig. 3 visualizes the correlation process encoded in the layer. Let \( I_f^{t−1} \) denote a feature of RGB image \( I^t \) at pixel \((u, v)\). Each feature is correlated with a patch of features denoted by \( P_f^{t−1} \). The patch is centered at \( I_f^{t−1} \) and has a side length of \( 2L+1 \), i.e. the dimension of the patch encoding is \((2L + 1, 2L + 1, 256)\). The correlation operation between features \( I_f^{t−1} \) and \( I_f^{t−1} \) inside the patch \( P_f^{t−1} \) is defined as

\[
c(\mathcal{I}_f^{t−1}, \mathcal{I}_f^{t−1}) = \langle \mathcal{I}_f^{t−1}, \mathcal{I}_f^{t−1} \rangle \text{ if } |u−k| \leq L, |v−l| \leq L
\]

(2)

The output vector of correlation between the single feature \( I_f^{t−1} \) and corresponding patch \( R_f^{t−1} \) has a dimension of \((2L + 1)^2\). The correlation is performed at each pixel within \( I_f^{t−1} \) with a stride of \((W/8, H/8)\). The final output shape of the correlation layer is \((W/8, H/8, (2L + 1)^2)\).

Highly correlated RGB patches also indicate which parts in consecutive point clouds correspond to each other. We therefore multiply the correlation value tensor with the corresponding \( P_f^{t−1} \) features to get a weighted XYZ feature encoding \( \hat{P}_f^{t−1} \). Then we apply max pooling to this results along the feature dimension as follows:

\[
\hat{P}_f^{t−1} = \max_{|u−k| \leq L} (c(\mathcal{I}_f^{t−1}, \mathcal{I}_f^{t−1}) | P_f^{t−1} \rangle)
\]

(3)

We concatenate \([P_f^{t}, P_f^{t−1}, \hat{P}_f^{t−1}]\) and feed this into another encoder until reaching a feature map with size \((W/60, H/60, 512)\) before feeding it into a decoder. Skip links are created between encoder and decoder. The decoder generates three images \( Q, T \) and \( X \) representing per-pixel estimates of rotation, translation and center position of the object projected to that pixel. Per-pixel scene flow can then be computed through Eq. 1.
Therefore, their correlation example will output $P$ values are used to weight corresponding cells of the XYZ feature map with every cell within a patch of the feature map.

Maps (black) are generated, each cell in the feature map $I$ contains the probability $\xi_{uv}$ of teh moving object. Thereafter, the segmentation ID is determined using the center of the object and its predicted translation. For predicting scene flow, the translation, rotation, and input XYZ data is utilized. The final output is presented as a segmentation mask and scene flow predictions. Note that the blue, red, and green arrows do not have gradient flow.

Fig. 3. Process of correlation and max-pooling. After two RGB feature maps (black) are generated, each cell in the feature map $I_f^t$ is correlated with every cell within a patch of the feature map $I_f^{t-1}$. Let us assume that the yellow cells $F$ and $I$ contain features corresponding to the same object. Therefore, their correlation $c(I_f^t, I_f^{t-1})$ will be high. These correlation values are used to weight corresponding cells of the XYZ feature map $P_f^{t-1}$ (gray). The result is fed into a max-pooling layer which in this example will output $c(I_f^t, I_f^{t-1})P_f^{t-1}$. The final feature $t''$ containing object XYZ information at frame $t-1$ will be placed at the same location as feature $F$ at frame $t$.

To segment moving objects, we propose the following inference process. Let $B$ be an additional output image of our model. A pixel at $(u, v)$ contains a scalar value $B_{uv}$. This value is a radius estimate of the sphere that encloses all pixels which belong to the same moving object, i.e. have a similar trajectory. The sphere is centered at $\hat{\xi}_{uv}$. Any pixel at coordinates $(o, p)$ whose $\hat{\xi}_{op}$ falls inside the sphere centered around $\hat{\xi}_{uv}$ will be segmented as the same object $O_k$. Any pixel at $(m, n)$ whose $\hat{\xi}_{mn}$ falls outside the sphere will be part of the background or a different object. In addition to $B$, we also learn a mask layer to discard pixel in this segmentation process that belong to the background.

To generate the ground truth of $B^gt$, each pixel $(u, v)$ representing object $O_k$ is annotated by half of the minimum distance between $\xi_k$ and the trajectories $\xi_l$ of all the other objects in the image pair:

$$B_{uv}^{gt} = \frac{1}{2} \min_{k \neq l} ||\xi_k - \xi_l||_2$$  \hspace{5cm} (4)

Inspired by region proposals [28], our model also outputs an image denoted by $\eta$. Each pixel in this image at $(u, v)$ contains the probability $\eta_{uv}$ that it is the projection of the object centroid.

To generate the ground truth of $\eta$, we sort pixels representing object $O_k$ by their distance to the object’s centroid in ascending order. The top 300 pixels per object in the input image $I$ will be annotated as 1, the rest will be annotated as 0. If the total number of pixels representing object $O_k$ is less than 300, all of them are annotated as 1.

Given the predicted $B$ and $\eta$, we can now perform multi-object segmentation as visualized in Fig. 4. Pixel $(u, v)$ with the maximum predicted probability $\hat{\eta}_{uv}$ is proposed first. Given a sphere centered at $\hat{\xi}_{uv}$ with radius $\hat{B}_{uv}$, all pixels $(m, n)$ with a trajectory $\hat{\xi}_{mn}$ enclosed by this sphere are assigned to object $O_k$. All pixel assigned to $O_k$ are removed from the set of unsegmented pixels before segmenting the next object. The remaining pixel at $(o, p)$ with the highest
\( \bar{\hat{\xi}}_{uv} \) is used as the seed for segmenting \( O_2 \). This process is repeated until all foreground pixels are assigned an object id \( k \). The final object translation \( T_k \) and rotation \( R_k \) is computed by averaging over all pixels with the same id. Based on this, also the scene flow can be recomputed.

C. Loss Function

We use the following training loss:

\[
L = \lambda_m L_m + \lambda_{center} L_{center} + L_p \\
+ \lambda_{var} L_{var} + \lambda_{vio} L_{vio}.
\]  

In the following, we define each term. Note that all pixel-wise loss terms \( L_p, L_{center}, L_{var} \) and \( L_{vio} \) are only computed on the ground truth foreground pixel.

1) Mask Loss: \( L_m \) is the cross-entropy loss between the ground-truth and estimated foreground/background segmentation. If a pixel is the projection of an object point, we assign 1 as ground truth; otherwise 0.

2) Cluster Center Loss: Cross-entropy loss \( L_{center} \) is used to learn the probability \( \bar{\hat{\xi}}_{uv} \) of a pixel \( (u,v) \) to be the object center as described in Sec. [IV-B].

3) Pixel-wise Loss: We use a pixel-wise loss \( L_p \) on the predicted object rotation \( Q_{uv} \), translation \( T_{uv} \), centroid \( X_{uv} \), scene flow \( S_{uv} \), enclosing sphere radius \( B_{uv} \) and trajectory \( \xi = [X_{uv} + T_{uv}] \). For each attribute, we use the L2-norm to measure and minimize the error between predictions and ground truth. Note that the loss on each attribute is also differently weighted. We denote their corresponding weights \( \lambda_Q, \lambda_T, \lambda_X, \lambda_S, \lambda_B \) and \( \lambda_\xi \).

4) Variance Loss: We use \( L_{var} \) to encourage pixels \( (u,v) \) belonging to the same object \( O_k \) to have similar trajectories \( \hat{\xi}_{uv} \) and thereby to reduce their variance.

\[
L_{var} = \sum_k \frac{1}{N_k} \sum_{(u,v) \in O_k} \| \hat{\xi}_{uv} - \bar{\hat{\xi}}_{uv} \|_2^2
\]

where \( \bar{\hat{\xi}}_{uv} \) is the mean value of \( \hat{\xi}_{uv} \) over all \( N_k \) pixels belonging to \( O_k \).

5) Violation Loss: \( L_{vio} \) penalizes pixels \( (u,v) \) that are not correctly segmented. Any predicted trajectory \( \hat{\xi}_{uv} \) that is more than \( \frac{1}{5} B_{uv} \) away from the ground truth \( \xi_{uv} \) will be pushed towards the ground truth trajectory by the violation loss. Note that \( B_{uv} \) refers to the radius of the enclosing sphere.

\[
L_{vio} = \sum_k \sum_{(u,v) \in O_k} \mathbb{1}\{\| \hat{\xi}_{uv} - \xi_{uv} \|_2 > \frac{1}{5} B_{uv} \} \| \hat{\xi}_{uv} - \xi_{uv} \|_2
\]

V. Dataset

We generated a new dataset that consists of RGB-D image pairs showing dynamic scenes. These scenes contain a large variety of rigid objects which are moving randomly between frames. Our dataset differs from existing datasets by being much more relevant to robotic manipulation research: it contains moving, graspable objects and is recorded with an RGB-D camera that is very common on manipulation robots. See Fig. 9 for some example frames. To ensure a diverse data set, we used 31594 3D object mesh models from ShapeNet [5] covering 28 categories. We split these models into a training, validation and test set with 21899, 3186 and 6509 objects respectively. Model sizes are adjusted to simulate their real world sizes [29]. For each scene, 1-30 object models are randomly selected. For simulating realistic object motion, we use Bullet [7] as physics engine. The objects are put close to each other at 0.2 meter above the ground. After simulation begins, they start to fall down to the ground and collide with each other in the process. Two frames are extracted from the simulated image sequence and used as RGB-D image pair. 24994, 3360, 7186 frame pairs are synthesized using models from training, validation and test sets respectively. In total, we generated 35540 pairs of consecutive RGB-D frames using Blender [14] to ensure realistic depth data. For each rendered RGB image pair, we randomly sample an image from the SUN397 dataset [33] to simulate textured floor or we use a single color. We also randomly change the lighting conditions (number of light sources, their positions and energies) and camera viewpoint.

Annotating Objects with Rotational Symmetry

Some of the objects in ShapeNet [5] are rotationally symmetric, e.g. bottles and bowls. Rotational symmetry is a common object attribute especially for human-made objects. However, the rotation of such an object around its symmetry axis cannot be estimated from an image pair (especially when the object is uniformly colored) as there might be multiple or even infinite solutions. There are different orders of rotational symmetry denoted by \( C_2, C_3, C_4, \cdots, C_n, \cdots, C_\infty \). An object with \( C_n \) means that it will remain the same after rotating about the rotation axis by \( \pm 360/n \) degrees. An object might contain several different rotational symmetries. Fig. 5 illustrates an example.

This has implications for the ground truth annotation of our dataset. If we directly use the ground truth rotation provided by the simulator, the network might not converge during training as more than one rotation might lead to the
same RGB-D data. In the following, we describe a procedure to map the ground truth rotation of an object about its symmetry axis to the rotation with minimum angular displacement. Consider an object with $C_n$ rotational symmetry. Let $\bar{r}_{t-1}$ and $\bar{r}_t$ denote this axis of symmetry at frame $t-1$ and frame $t$, respectively. Let the rotation provided by the simulator be given as a quaternion $q = [q_0, q_x, q_y, q_z]^T$. We decompose the rotation $q$ into a rotation $\alpha$ about $C_n(\bar{r}_{t-1})$ and a rotation $\theta$ perpendicular to $C_n$: $(\bar{r}_\perp = \bar{r}_{t-1} \times \bar{r}_t)$.

$$\alpha = 2\tan^{-1}\left(\frac{r_{t-1},xq_x + r_{t-1},yz + r_{t-1},zq_z}{q_0}\right)$$

$$\theta = 2\cos^{-1}(q_0 / \cos(\alpha/2))$$  \hspace{1cm} (7)  \hspace{1cm} (8)

$\alpha$ is then adjusted to be $\hat{\alpha} \in (-\pi/n, \pi/n)$. This corresponds to the minimum angular displacement leading to the same observation as the original angle. From this, we can construct a new quaternion $\hat{q}$ which corresponds to the rotation of $\hat{\alpha}$ about $\bar{r}_{t-1}$ and rotation of $\theta$ about $\bar{r}_\perp$. Note that if $\alpha = \hat{\alpha} \in (-\pi/n, \pi/n)$ then $q = \hat{q}$. This operation is performed on all the rotational axis of symmetry. With this procedure, we reduced the ambiguous cases to a very small number, e.g. to uniformly-colored objects with non-orthogonal axes of symmetry (of which there exists one among our models) or rotations as shown in Fig. 5 where the minimum angular displacement can either refer to a rotation in the positive or negative direction.

VI. EXPERIMENTS

In this section, we quantitatively report the performance of the proposed model on the synthesized dataset and qualitatively on real data. We evaluate accuracy in scene flow prediction by comparing to PD-Flow [19]. We evaluate motion-based segmentation performance by comparing to Higher-Order Minimum Cost Lifted Multicuts (HOMC) [21].

Furthermore, we compare to variants of the proposed architecture. We refer to the network in Fig. 2 as OurC and propose a simpler neural net architecture denoted by OurS. It concatenates all four input images and feeds it into the encoder. Most importantly, it drops the correlation and max pooling layer. The remaining model architecture is the same. OurC+vL denotes added variance and violation loss compared to training OurC. Our model OurC+vL is simultaneously predicting pixel-level segmentation IDs and scene flow. Given all pixels with the same, predicted object ID, we compute the mean object center $\bar{X}_k$, translation $\bar{T}_k$ and rotation $\bar{Q}_k$. OurC+vL+Rig denotes the model with added rigidity constraints for improved scene flow estimation.

Our experiments are conducted on an NVIDIA P100 with TensorFlow. For training, we use the Adam optimizer [23] with its suggested default parameters of $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [23] along with a batch size of 12 image pairs. The learning rate started at $\lambda = 0.0001$ and is consistent across the experiments. The input RGB-D images have a resolution of $240 \times 320$. The loss weights, as defined in Sec. IV-C, are set to $\lambda_m = 1.0$, $\lambda_{var} = 0.1$, $\lambda_{vio} = 0.1$, $\lambda_Q = 0.1$, $\lambda_T = 100.0$, $\lambda_X = 10.0$, $\lambda_S = 10.0$, $\lambda_B = 1.0$ and $\lambda_\xi = 1.0$.

A. Evaluation of Scene Flow Performance

We compare the proposed method with PD-Flow [19] using standard evaluation metrics as defined in [34]: end point error (EPE) and 4D average angular error (AAE) error. Both of these metrics, EPE and AAE, are calculated as averages over the entire image and are reported in meters and degrees, respectively. Because it is not possible to calculate the scene flow for an object that is only present in one of the two frames, we also report masked EPE and masked AAE which only calculates the desired metrics on objects that are in both frames. The results are presented in Fig. 6.

All our proposed models outperform PD-flow by a large margin. OurC and its variants perform better than the simple model version OurS without the correlation layer. This comes at the expense of a higher processing time. However, the most complex model OurC+vL+Rig can still run at 8.3 frames per second.

B. Evaluation of Motion-based Segmentation

We evaluate our model’s ability to perform motion-based segmentation by comparing to HOMC, the state-of-the-art technique by Keuper [21]. Unlike our method which requires only two RGB-D images, this method requires a sequence of RGB images. To satisfy this requirement, we repeat our two images, $A$ and $B$, five times to form a 10 image sequence, [ABABABABABAB]. Given this sequence of images, HOMC phrases the clustering of motion trajectories as a minimum cost multi-cut problem. The result forms the basis to propose segmentations. As suggested in the original minimum cost multi-cut paper [22], the sampling is set to 4 and the prior is set to 0.5. Keuper [21] provided an executable file for the

| Method   | EPE all | EPE masked | AAE all | AAE masked | Runtime (second) |
|----------|---------|------------|---------|------------|------------------|
| PD-flow  | 0.02830 | 0.08041    | 1.6055  | 4.572      | 0.046            |
| OurS     | 0.01643 | 0.05324    | 0.9282  | 3.020      | 0.059            |
| OurC     | 0.01330 | 0.04333    | 0.7499  | 2.457      | 0.078            |
| OurC+vL  | 0.01315 | 0.04333    | 0.7415  | 2.457      | 0.078            |
| OurC+vL+Rig | 0.01303 | 0.04290    | 0.7343  | 2.432      | 0.121            |

Fig. 6. Performance of scene flow prediction. EPE in meters. AAE in degrees. The learned models greatly outperform the baseline PD-Flow. OurC and its variants perform better than the simple model version OurS without the correlation layer.
state-of-the-art technique upon request and it takes roughly 35 seconds to produce the segmentation.

To evaluate the different segmentation results, we focus on a set of four metrics that are frequently presented in segmentation papers: precision, recall, F-measure, and extracted objects [22]. Following conventions that were presented in [20], the metrics were calculated on the segmentations produced by the HOMC technique and our three network architectures. The F-measure threshold of 0.75 was used for object extraction. The results are reported in Fig. 7.

HOMC [21] achieves a better precision but a very low recall which indicates under-segmentation. It only extracted 6 percent of the objects in the test set. All our proposed methods show a significant improvement on recall, F-measure and number of extracted objects while still retaining a high precision score. Note, however, that HOMC relies on a longer sequence of images, it does not have access to the strong depth cue. Some example results are shown in Fig. 9. These results highlight another advantage of our approach. It provides dense segmentation.

C. Architecture Design Analysis

1) Effects of correlation layer: We report the training and validation loss curve in Fig. 8. OurC has a significantly lower training and validation loss than OurS. We also showed that OurC outperforms OurS both in scene flow prediction and motion-based segmentation. This demonstrates the impact of adding correlation layers in OurC.

2) Effects of using variance and violation loss: Comparing with OurC, OurC+vL improves motion-based segmentation, but only gains small improvements on scene flow prediction. It indicates that the variance and violation loss are effective for motion-based segmentation.

3) Effects of using rigid motion cues: The best scene flow prediction performance is achieved by adding rigid constraints (OurC+vL+Rig). However the improvement over OurC is only marginal. The difference to OurS remains significant, underlining the importance of the correlation layer.

D. Results on Real World Data

Finally, we demonstrate the networks ability to perform in a real world setting. We recorded real RGB-D data with the Intel RealSense SR300 Camera. The data includes large displacements, occlusions, and collisions. It was captured using a diverse set of objects with varying geometries, textures, and colors. Note that we do not have any ground truth annotations and that the model is not fine-tuned to transfer from synthetic to real data. Some example images and corresponding outputs are displayed in Fig. 10.

Because the real data produces one long sequence, we apply HOMC [21] on the stream of real data as one long sequence. The resulting segmentations appear to be slightly more accurate than the ones produced from image pairs.

The real world RGB-D images are fed into our OurC+vL+Rig model. The results are also presented in Fig. 10. The accurate real world segmentation and scene flow prediction results strongly indicate the small sim-to-real transfer gap of our deep nets. However there are still some failure cases that persist, such as: noisy sensor data, varied lighting conditions during an experiment, and multiple neighboring objects moving along similar trajectories.

VII. CONCLUSION

We proposed a deep neural network architecture that given two consecutive RGB-D images can accurately estimate object scene flow and motion-based object segmentation. We demonstrated this on a new and challenging, synthetic data set that contains a large variety of graspable objects moving simultaneously. We showed that the correlation layer makes a crucial difference to training time and accuracy and outperforms the state-of-the-art baselines in scene flow prediction and motion-based segmentation. Additionally, we showed how our approach performs on real RGB-D data when only trained on synthetic data. The results look qualitatively more accurate than baseline methods. Overall, we demonstrated the power of learning based methods over traditional methods in situations of large displacements and strong occlusions. This has previously only been demonstrated by [24] in a binocular setting. We were interested in data that is relevant to robotic manipulation. In future work, we will explore how this approach enables agile, robotic manipulation in cluttered scenes.

ACKNOWLEDGMENTS

We would like to thank Margret Keuper for providing an executable of her motion-based segmentation method [22] and her support in evaluating on our data set.
Fig. 9. Performance comparison of the proposed method on our synthetic data set. First two columns show RGB inputs, next columns are Scene Flow from PD-flow, our method, and the ground truth, the final three columns correspond to segmentation results from multi-cut, our method, and the ground truth. In scene flow images, green, blue, red intensities are proportional to the velocities along X, Y, Z respectively. In the HOMC segmentation, colored pixels (not gray or white) have been successfully clustered with longer trajectories to produce valid segmentations.

| RGB Inputs | Scene Flow | Segmentation |
|------------|------------|--------------|
| frame $I^{t-1}$ | PD-flow | OurC+V+L+Rig | Ground Truth | HOMC | OurC+V+L+Rig | Ground truth |
| frame $I^t$   |           |              |              |               |              |               |

Fig. 10. Performance comparison of the proposed method on the real-world data set. First two columns show RGB inputs, next columns are Scene Flow from PD-flow and our methods, last two columns correspond to segmentation results from multi-cut and our method.

| RGB Inputs | Scene Flow | Segmentation |
|------------|------------|--------------|
| frame $I^{t-1}$ | PD-flow | OurC+V+L+Rig | HOMC | OurC+V+L+Rig |
| frame $I^t$   |          |              |       |               |

REFERENCES

[1] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. IEEE Transactions on Robotics, 2017.

[2] W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In Computer Vision, 2009 IEEE 12th Int. Conf. on, 2009.

[3] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. Computer Vision—ECCV 2010, 2010.

[4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In European Conf. on Computer Vision. Springer, 2012.

[5] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[6] A. M. Cheriyadat and R. J. Radke. Non-negative matrix factorization of partial track data for motion segmentation. In Computer Vision, 2009 IEEE 12th Int. Conf. on. IEEE, 2009.

[7] E. Coumans. Bullet physics simulation. In ACM SIGGRAPH 2015 Courses, SIGGRAPH ’15. ACM, 2015.

[8] A. Dosovitskiy, P. Fischery, E. Ilg, P. Husser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In 2015 IEEE Int. Conf. on Computer Vision (ICCV), 2015.

[9] H. Fushing, H. Wang, K. VanderWaal, B. McCowan, and P. Koehl. Multi-scale clustering by building a robust and self correcting ultrametric topology on data points. PloS one, 2013.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. Int. Journal of Robotics Research (IJRR), 2013.

[11] V. Golyani, K. Kim, R. Maier, M. Niefner, D. Stricker, and J. Kautz.
Multiframe scene flow with piecewise rigid motion. In Proceedings of the Int. Conf. on 3D Vision (3DV), 2017.

[12] J. Gottfried, J. Fehr, and C. Garbe. Computing range flow from multimodal kinect data. In Int. Symposium on Visual Computing, 2011.

[13] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conf. on. IEEE, 2010.

[14] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree. Blensor: blender sensor simulation toolbox. Advances in Visual Computing, 2011.

[15] S. Hadfield and R. Bowden. Scene flow estimation using intelligent cost functions. In British Conf. on Machine Vision, 2014.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[17] E. Herbst, X. Ren, and D. Fox. Rgb-d flow: Dense 3-d motion estimation using color and depth. In Robotics and Automation (ICRA), 2013 IEEE Int. Conf. on. IEEE, 2013.

[18] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In IEEE Conf. on Computer Vision and Pattern Recognition, 2017.

[19] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers. A primal-dual framework for real-time dense rgb-d scene flow. In Robotics and Automation (ICRA), 2015 IEEE Int. Conf. on. 2015.

[20] M. Keuper. Higher-order minimum cost lifted multicuts for motion segmentation. 2017 IEEE Int. Conf. on Computer Vision (ICCV), 2017.

[21] M. Keuper, B. Andres, and T. Brox. Motion trajectory segmentation via minimum cost multicuts. In IEEE Int. Conf. on Computer Vision (ICCV), 2015.

[22] D. Kinga and J. Ba Adam. A method for stochastic optimization. In Int. Conf. on Learning Representations (ICLR), 2015.

[23] N. Mayer, E. Ilg, P. Hauser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition, 2016.

[24] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2015.

[25] P. Ochs, J. Malik, and T. Brox. Segmentation of moving objects by long term video analysis. IEEE transactions on pattern analysis and machine intelligence, 2014.

[26] H. Rahmati, R. Dragon, O. Aamo, L. Van Gool, and L. Adde. Motion segmentation with weak labeling priors. In German Conf. on Pattern Recognition. Springer, 2014.

[27] C. Vogel, K. Schindler, and S. Roth. Piecewise rigid scene flow. In 2013 IEEE Int. Conf. on Computer Vision, 2013.

[28] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In Computer vision and pattern recognition (CVPR), 2010 IEEE Conf. on. IEEE, 2010.

[29] Z. Yan and X. Xiang. Scene flow estimation: A survey. CoRR, abs/1612.02590, 2016.

[30] V. Zografos, R. Lenz, R. Ringaby, M. Felsberg, and K. Nordberg. Fast segmentation of sparse 3d point trajectories using group theoretical invariants. In Asian Conf. on Computer Vision. Springer, 2014.