SE3-Nets: Learning Rigid Body Motion using Deep Neural Networks

Arunkumar Byravan and Dieter Fox
Department of Computer Science & Engineering
University of Washington, Seattle

***Abstract*—We introduce SE3-Nets which are deep networks designed to model rigid body motion from raw point cloud data. Based only on pairs of depth images along with an action vector and point wise data associations, SE3-Nets learn to segment effected object parts and predict their motion resulting from the applied force. Rather than learning point wise flow vectors, SE3-Nets predict SE3 transformations for different parts of the scene. Using simulated depth data of a table top scene and a robot manipulator, we show that the structure underlying SE3-Nets enables them to generate a far more consistent prediction of object motion than traditional flow based networks.**

I. Introduction

The ability to predict how an environment changes based on forces applied to it is fundamental for a robot to achieve specific goals. For instance, in order to arrange objects on a table into a desired configuration, a robot has to be able to reason about where and how to push individual objects, which requires some understanding of physical quantities such as object boundaries, mass, surface friction, and their relationship to forces. A standard approach in robot control is to use a physical model of the environment and perform optimal control to find a policy that leads to the goal state. For instance, extensive work utilizing the MuJoCo physics engine [31] has shown how strong physics models can enable solutions to optimal control problems and policy learning in complex and contact-rich environments [11, 22]. A fundamental problem of such models is that they rely on very accurate estimates of the state of the system [37]. Unfortunately, estimating values such as the mass distribution and surface friction of an object using visual information and force feedback is extremely difficult. This is one of the main reasons why humans are still far better than robots at performing even simple tasks such as pushing an object along a desired trajectory. Humans achieve this even though their control policies and decision making are informed only by approximate notions of physics [5, 28]. Research has shown that these mental models are learned from a young age, potentially by observing the effect of actions on the physical world [4]. Learning such a model of physical intuition can help robots reason about the effect of their actions, which is a critical component of operating in complex real-world environments.

In this work, we explore the use of deep learning to model this concept of "physical intuition", learning a model that predicts changes to the environment based on specific actions. Our model uses motion cues to learn to segment the scene into "salient" objects (much akin to a saliency map [17]) and jointly predicts the motion of these objects under the effect of applied actions. We restrict our attention primarily to modeling the motion of systems of rigid bodies, though our model can represent arbitrary 3D motion (mixing multiple rigid motion components is a standard way of modeling deforming objects [25]).

Many objects we interact with in the physical world are rigid or can be modeled as systems of rigid bodies. Rigid body motion is a well understood field of physics and is widely used in robotics, computer vision, and graphics. We borrow tools from rigid body physics to design a deep network, **SE3-Net**, that represents motion in the environment as a set of $\mathbf{SE3}$ transforms ($\mathbf{SE3}$ refers to the Special Euclidean Group that represents 3D rotations and translations: $\{R, t | R \in \mathbf{SO}(3), t \in \mathbf{R}^3\}$). Our network takes a raw 3D point cloud from a calibrated kinect sensor and a continuous action vector as input and predicts a transformed point cloud by applying a set of rigid body transformations. Key to our model is the notion of disentangling the motion of the objects (the **What?**) from their location in the environment (the **Where?**). Our network does this by explicitly predicting a set of "k" $\mathbf{SE3}$ transforms to encode the motion and "k" dense pointwise masks that specify the contribution of each $\mathbf{SE3}$ towards a 3D point. Finally, our model combines the $\mathbf{SE3}$s, their masks, and the 3D input through a differentiable *transform* layer that blends the motions to produce the output point cloud.

In the absence of any constraints, our network can represent an arbitrary per-point 3D motion. To restrict the network to model rigid body motion, we adapt a weight sharpening technique used in [35] and show that this results in a segmentation of the environment into distinct objects along with their rigid motion. Finally, we show results on three simulated scenarios where our system predicts the motion of a varying number of rigid objects under the effect of applied forces and a 14-DOF robot arm with 4-actuated joints. Our model predicts consistent motion, performing better than two 3D optical-flow baselines (measured on flow prediction error), while also learning a notion of "objects" without explicit supervision.

This paper is organized as follows. After discussing related work, we introduce **SE3-Nets** in Section III, followed by experimental evaluation and discussion.

II. Related work

Robotics: As mentioned before, many optimal control techniques require a "dynamics" model that predicts the effect

of actions on the state of the system [30, 32]. Early work on learning dynamics models from data mostly focused on low-dimensional state and control representations [8]. In contrast to these methods, Boots et al. [6] learn a model using Predictive State Representations to predict a depth image given a history of prior images and control. Unlike their work, which operates in a Hilbert-space embedding of depth images, **SE3-Nets** explicitly encode the dynamics as rigid body motion, also learning a notion of object segmentation.

Deep learning in robotics: Deep learning models have recently been very successful on a wide variety of tasks in computer vision such as classification, semantic labeling and object recognition. Deep models have also been used for learning dynamics models in robotics and reinforcement learning, by mapping raw pixel images to low-dimensional (latent) encodings on top of which standard optimal control methods are applied [34, 33, 9]. **SE3-Nets** can also be used similarly and have the added advantage of a generative model that operates in the interpretable 3D physical world.

Physics prediction: Our model is related to recent work in deep learning for physics based prediction, such as predicting the stability of a tower of blocks from images [19, 20], predicting the motion of a billiards ball under forces [10] and predicting the dynamics of objects in images [23, 24]. These methods mostly predict low-dimensional outputs like ball velocity [10] or the probability of falling [20]. One exception is the work by Lerer et al. [19], which predicts images of a tower of blocks, but their network operates on RGB images and has no specific notion of an action or forces.

Predicting 3D rotation: Related work in the computer vision literature has looked at the problem of predicting 3D motion, primarily rotation between pairs of images [13, 18, 2, 26]. Perhaps, the most similar work to ours is by Oh et al. [26], who proposed a deep model that predicts the change in an encoded pose vector through the effect of a one-hot action vector to render rotated color images of objects. Differing from their work, we operate on 3D data, use a continuous action vector, and explicitly predict rigid body motion and object masks.

Attentional models and disentangling representations: A related line of work to ours is the idea of attentional mechanisms [12, 36, 3] which focus on parts of the environment related to task performance and the concept of disentangling representations [7, 18, 26], which aim to separate variations in the environment. Our model has a differentiable, dense-pointwise attender that learns to focus on parts of the environment where motion occurs, using these cues to segment objects. Also central to our model is the idea of disentangling the motion of the object from its location. Finally, our model is related to the Spatial Transformer network [15], though we model the effect of actions, use dense attention and restrict ourselves to $\mathbf{SE3}$ transformations.

III. SE3-NETS

Fig.1 shows the general architecture of **SE3-Nets** . Our network takes a 3D point cloud (X) shaped as a 3-channel image and an n-dimensional continuous vector as input and generates a transformed 3D point cloud (Y) as output. There are three major components to our network: an encoder, a decoder and a final transform layer. The encoder is a straightforward late-fusion convolutional/fully-connected architecture that processes the input point cloud and controls separately to produce low-dimensional encoded vectors. We concatenate the encoded vectors to produce a single joint encoding which is used by the rest of the network.

A. Decoder

The concatenated encoding vector is used by the decoder to predict the motion in the scene by separating it into two parts: the mask, which attends to where motion occurs, and the $\mathbf{SE3}$ transformation parameters, which specify the motion itself.

What motion is happening? We represent motion in the environment using 3D rigid body transforms. A rigid body transform in 3D $[R, t] \in \mathbf{SE3}$ can be specified by a rotation $R \in \mathbf{SO(3)}$ and a translation $t \in \mathbf{R}^3$. We include a pivot term $p \in \mathbf{R}^3$ to model an arbitrary change of the point around which rotation happens, in case it differs from the camera's viewpoint. A 3D point x under the action of this transformation moves to:

$$x' = R(x - p) + p + t \tag{1}$$

We choose to represent rotations using a 3-parameter axis-angle transform $a \in \mathbf{R}^3$, with $||a||_2 = \theta$, the magnitude of rotation. This gives us a total of 9 parameters per transform[1].

Given the encoder output, the first module of the decoder is constrained to predict k $\mathbf{SE3}$ transforms, where k is a pre-specified network parameter which limits the number of distinctly moving objects or parts (including background which has no motion) that the network can model. We use a simple three-layer fully-connected network for this prediction.

Where is the motion happening? The second module of the decoder is a dense-pointwise attender trained to focus on parts of the environment that exhibit motion. Under the assumption that the environment has k distinct motions, we can formulate this as a k-class labeling problem where each point can belong to one of the k motion-classes. Unfortunately, this formulation is non-differentiable due to the discreteness of the labeling. Instead, we relax the formulation to predict a per-point probability distribution M_i over the k motion classes:

$$M_i = \{m_{i1}, m_{i2}, \ldots, m_{ik}\} \mid \sum_{j=1}^{k} m_{ij} = 1; \tag{2}$$

allowing each point to smoothly interpolate between multiple motions.

This dense object mask is computed from the joint encoding through a de-convolutional pipeline. We follow recent work in semantic labeling [21] and add the outputs of our convolutional layers to the inputs of the de-convolutional layers. In practice, this gives us much sharper reconstructions of the object

[1]For the rest of this paper, we will misuse notation to refer to this 9 parameter transform as an $\mathbf{SE3}$ transform, though it is actually $\mathbf{SE3} \times \mathbf{R}^3$

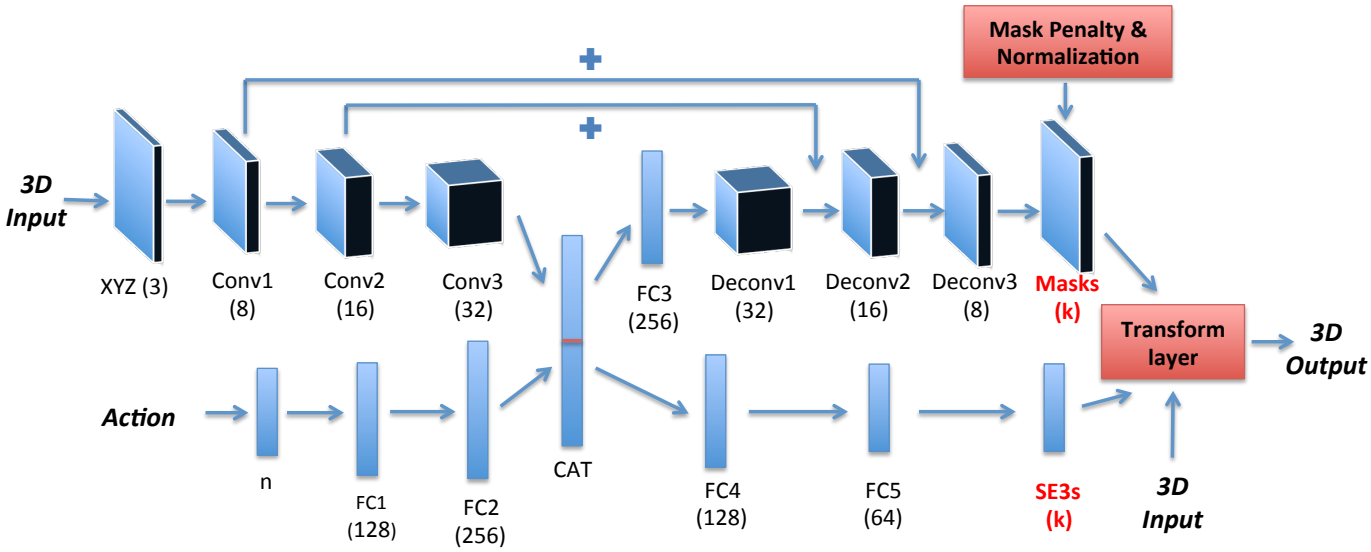

Fig. 1: **SE3-Net** architecture. Input is a 3D point cloud along with an n-dimensional action vector (bold-italics), both of which are encoded and concatenated to a joint feature vector. Decoder uses the encoded vector to predict object masks and **SE3** transforms which are applied to the input cloud via the transform layer to generate the output. Mask weights are sharpened and normalized before use for prediction. Conv = Convolution, FC = Fully Connected, Deconv = Deconvolution, CAT = Concatenation

shapes and contours in the environment, improving overall performance.

B. Transform layer

Given the predicted the **SE3** parameters and the mask weights, the transform layer produces a blended output point cloud from the input points:

$$y_i = \sum_{j=1}^{k} m_{ij} \left(R_j \left(x_i - p_j \right) + p_j + t_j \right) \tag{3}$$

where y_i is the 3D output point corresponding to input point x_i. (3) computes a convex combination of transformed input points, transformed by each of the k **SE3** transforms with weights given by the object mask. As a consequence of our relaxation for the mask prediction, the effective transform on a given point is generally not in **SE3** as (3) blends in 3D space rather than in the space of **SE3** transforms. On the other hand, we now have the flexibility to represent both rigid and non-rigid motions through a combination of the transforms and the object masks. Additionally, we avoid having to blend in **SE3** space which can lead to singularities. In spite of the advantages, using the current framework to model rigid motion without any explicit regularization can lead to overfitting (as evidenced in our results). We now show how we encourage the network to predict rigid motions through a form of regularization on the object mask.

Enforcing Rigidity: A simple way to restrict the network (3) to predict rigid motions is to force the mask probability vector M_i to make a binary decision over the k predicted transforms instead of blending. As mentioned before, a naive formulation can lead to non-differentiability. Instead, we smoothly encourage the mask weights towards a binary decision using a form of weight sharpening [35]:

$$m_{ij}{}' = \frac{(m_{ij} + \mathcal{N}(0, \sigma^2))^{\gamma}}{\sum_k m_{ik}{}^{\gamma}} \tag{4}$$

where γ and σ are proportional to the training epoch. In practice, the combination of the noise and growing exponent forces the network to separate its decisions apart, resulting in nearly binary distributions at the end of training.

IV. Evaluation

We evaluate **SE3-Nets** to predict rigid body motion on different tasks simulated using the Gazebo physics simulator. The task of the network is to predict the motion of a robot manipulator and rigid objects based on control commands and a moving ball, respectively. The object motion scenarios require the network to learn to segment effected objects from the scene and predict their motion, which depends on which location on the object is being hit by the ball. The other scenarios test how well the network is able to learn the articulation of a robot manipulator resulting from input controls. All our datasets contain sequences of simulated point clouds taken from a depth camera looking at the environment from a fixed viewpoint (for each task) along with the state of all the objects and dense 3D optical flow between point cloud pairs. In all our tasks, the network predicts a target 5 timesteps into the future (0.15 sec) given the current point cloud (3 x 240 x 320) and control. We assume that the control is held fixed for this duration.

Single Box: We generated 9,000 scenes where a ball is made to collide with a box placed in a random position on a table. At the start of each scenario, the ball is placed at a random position and a randomly chosen constant force is continuously applied to the ball, directing it to collide with the box. Each scene lasts for a second, and we stop recording if

the box falls off the table. The control vector for this dataset is 10-dimensional, with the 6D pose of the ball represented as a position and quaternion, and the applied force. We vary the starting pose of the ball, box, the table size and the applied forces while the size and mass of the ball and box are held fixed. In total, this dataset has around 170,000 examples, which we split 70:30 for train/test.

Multiple Boxes: To test the generalization of the system to different object sizes, masses and number of objects, we generate another dataset that varies all three at random. Each scene has anywhere from 1-3 objects of varied sizes with mass proportional to their size, and the ball is forced to collide with a randomly chosen box. We restrict ourselves to cases where only a single box and the ball are in collision and discard examples that involve multiple collisions, since it is extremely hard for the system to disambiguate the motion without any velocity or other temporal information. This dataset has 12,000 scenarios, with a total of 210,000 examples.

Baxter: Our third dataset consists of sequences of depth images of a simulated Baxter robot being controlled to move its arms in front of the camera. For each scenario, we command a constant, randomly chosen velocity to 1-4 randomly chosen joints on the robot's right arm. Each scenario lasts for a second, at the end of which we zero the velocity of the arm. We command the arm to a random pose every 20 scenarios. In total, this dataset has around 11,000 scenes with 220,000 examples. Our controls for this task are the commanded joint velocities, a 14-dimensional vector of which up to 4 values are non-zero.

A. Training

We implemented our system using the deep learning package Torch [1]. We train our networks using standard backpropagation and the ADAM optimization method [16] along with Google's Batch Normalization technique [14] to speed up training. We use the PReLU non-linearity throughout the network. At the start of training, we initialize the layer predicting **SE3** transforms to predict identity which we found to improve convergence. We initially set the weight scheduling penalty to zero and slowly ramp up the noise parameter σ and the exponent γ till they reach a maximum. We set $k = 3$ for the box datasets and $k = 5$ for the baxter dataset. We use the Huber loss with a delta of 0.05m for the Baxter dataset and 0.1m for the box datasets.

Training targets: We assume that ground truth data-associations are given to us at training time, meaning that the mapping between input points and target are known apriori. For our simulated datasets, each data point in the input has a one-to-one mapping to the point at the same pixel location on the output - we compute this target by adding our pre-recorded dense 3D optical flow to the input point cloud. This gets rid of any potential occlusions, missing data and reappearing data. We discuss ways to overcome this assumption later in Sec. V.

B. Comparison networks

We show results for the following networks: *Ours:* **SE3-Net** from Fig.1. *Flow:* Network trained to predict dense 3D optical flow directly. Uses a Conv/Deconv architecture almost identical to our network except for the **SE3** prediction module and the transform layer. *No Penalty:* **SE3-Net** without any weight sharpening to enforce binarization of the object masks. *Ours (Large):* Bigger version of the **SE3** network with 5-Conv/Deconv layers and 6x as many parameters as our original network. *Flow (Large):* Bigger version of the flow network, again with 5-Conv/Deconv layers. Has 8x as many parameters as the original **SE3-Net**. All networks were trained with the same parameters for the same number of epochs (per task).

C. Results

Table I shows the results from all the networks on the different tasks, as compared on average 3D flow error of all moving points. Our networks significantly outperform their counterpart small and large flow networks (respectively) on all tasks. Interestingly, we noticed that the flow networks perform quite poorly on examples where only a few points exhibit motion such as when just the ball moves in the scene while our networks are able to predict the ball's motion quite accurately.

Fig. 2 shows the projected depth images based on the predictions from the different networks on the box datasets. While the flow error for the large flow network is low, it is clear that the predictions given by the network are much noisier, as can be seen from the re-projections. On the other hand, both our networks are able to label the box and the ball as separate objects (see Fig. 5), predicting distinct rigid transformations that are consistent with the actual motion. This results in a crisper re-projection with almost no noise that matches the ground truth targets closely. In practice, we found that it is crucial to give the network datasets where the ball moves independently as this provides implicit knowledge that the ball and the box are distinct objects. In cases where the ball was always in contact with the box, the network had a hard time disambiguating between the objects, often labeling them together as a single object.

Similar results can be observed on the baxter dataset (Fig. 3), where the network consistently predicts the motion of the robot arm. We found that the network usually segments the arm into two distinct object classes (Fig.5), with a separation along the elbow. This is consistent with the fact that most of the apparent motion is due to the effect of two joints, one at the shoulder and the other at the elbow. Once again, we see that the **SE3-Net** results are sharper than those predicted by the flow networks, again highlighting the strength of our networks in modeling rigid motion. It is worth noting that the "No Penalty" version of our algorithm performs quite poorly across all tasks due to a slight overfit to the training data. This again highlights the fact that forcing the network to predict the rigid motion of individual parts greatly improves prediction performance.

To test how well our rigid motions and object models carry through across time, we tested our network on a simple sequential prediction task on the baxter dataset. We did a sequence of 5-step predictions where the network was input an initial point cloud and control vector (which was kept constant) and predicted a sequence of outputs by using its own

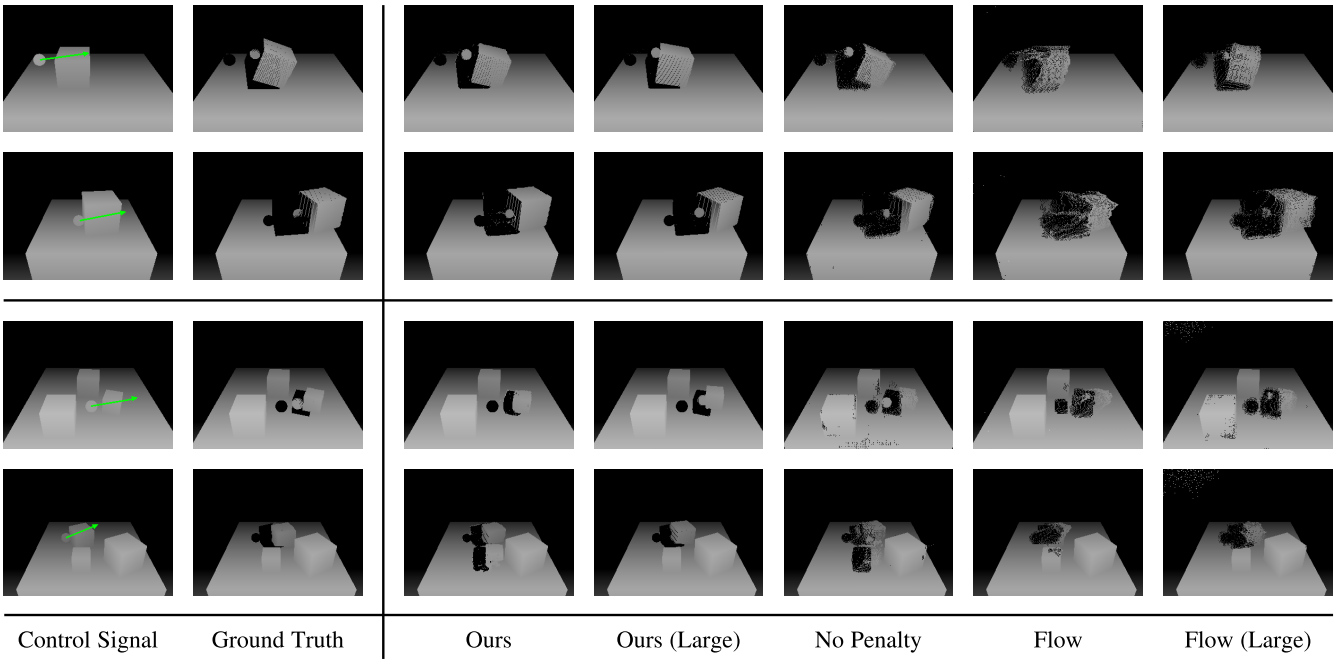

Fig. 2: Prediction results on the two box datasets. Top two rows show results from the "Single Box" dataset on two separate examples while the bottom two are example results from the "Multiple Boxes" dataset. All images (except first column on the left) shown were rendered by projecting the predicted 3D point cloud onto the 2D image using the camera parameters and rounded off to the nearest pixel without any interpolation. (left to right) Input point cloud with ball control flow vector shown in green; ground truth point cloud prediction; predictions generated by different networks. 3D point clouds for the flow networks were computed by adding the predicted flow to the input point cloud. Image best viewed in high resolution.

Task	Ours	Ours (Large)	No Penalty	Flow	Flow(Large)
Single Box	0.041	**0.019**	0.253	0.101	0.028
Multiple Boxes	0.036	**0.011**	0.197	0.062	0.019
Baxter	0.00070	**0.00059**	0.00074	0.00114	0.00071

TABLE I: Average per-point flow prediction error (m) across tasks and networks. Our (large) network achieves the best flow error compared to baselines even though it is not directly trained to predict flow.

output as input for the next timestep. Results for this test are shown in Fig. 4 which compares the ground truth target images across the sequence to the predictions from the large **SE3-Net** and flow models. Our predictions are very consistent across time with little noise (though the motion error does slightly cascade) while the flow model predictions become much noisier. Overall, we see that the notion of rigidity and object saliency significantly improves performance across multiple tasks we tested on, while also providing the capability to identify distinctly moving objects in the scene just using motion cues. We highly encourage our readers to view the video (https://www.youtube.com/channel/UCM5JZPr1pLWEgbFVM-PyG0w) of our results which clearly shows the strengths of our approach.

V. Discussion

We introduced **SE3-Nets** a deep learning model that learns to predict changes in the environment based on applied actions, parameterized as a series of rigid body motions applied to 3D points in the environment. **SE3-Nets** selectively learn to focus on parts of the scene where motion occurs, segmenting the scene into objects and predicting **SE3** motions for each distinct object. We showed that this separation works well in practice and results in strong performance on three simulated tasks with rigid bodies in motion. **SE3-Nets** are able to generalize across different scenarios and produce results that are very consistent with the observed rigid motion, as compared to traditional flow networks.

There are several promising directions to focus on towards improving these networks. First, we tested on simulated tasks in this paper, but it should be straightforward to extend the method to work on real world data - given dense data-associations. While we can obtain these type of associations from traditional tracking frameworks such as DART [29] or dense scene-flow based approaches [27], this might not be applicable in all scenarios. This brings us to the next area for improvement - learning data associations which is currently a limitation of **SE3-Nets** . Under the assumption of small motions, algorithms such as ICP can align pairs of similar point clouds together. We can extend this approach to **SE3-Nets** formulating a loss function based on ICP, which we believe is a simple first step in learning data-associations. Third, **SE3-Nets** cannot handle occlusions or fill-in data based on prior knowledge. An intuitive solution to handle this can be to extend the current paradigm of disentangled motion/mask computation by adding

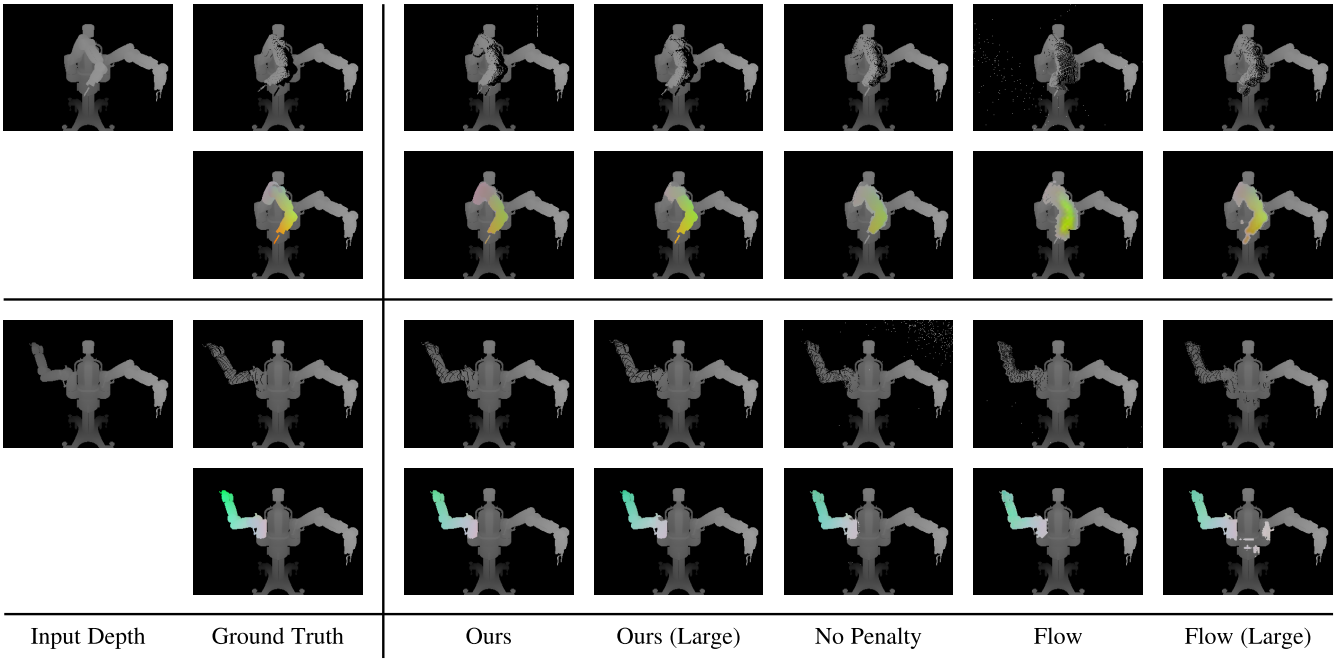

Fig. 3: Prediction results on the Baxter dataset. First and third rows show 2D perspective projections of the 3D point clouds for two separate examples. Second and fourth rows show the predicted 3D optical flow rendered on top of the ground truth input depth map for the same examples. Optical flow was computed for our networks by subtracting the predictions from the input point cloud. Image best viewed in high resolution.

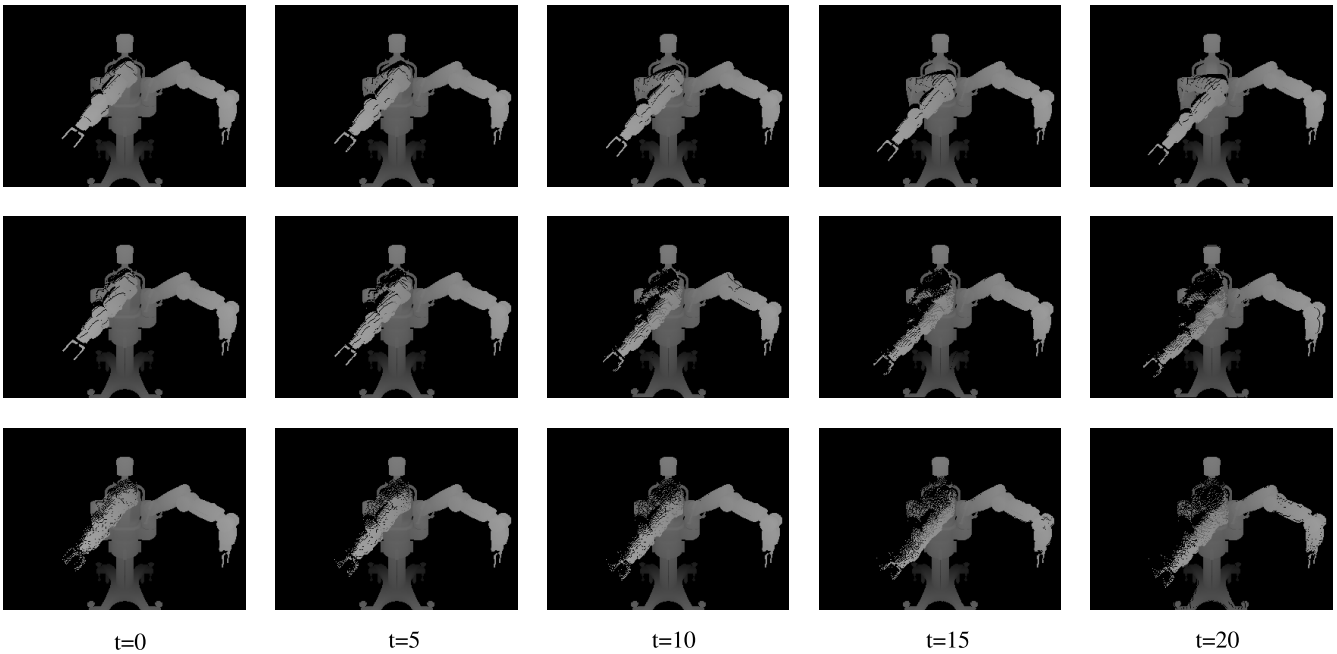

Fig. 4: Multi-step prediction results obtained by feeding back the output of the network to the input for 4-steps into the future. First row shows ground truth, second row the prediction of the large **SE3** network, and third shows the prediction of the larger flow network.

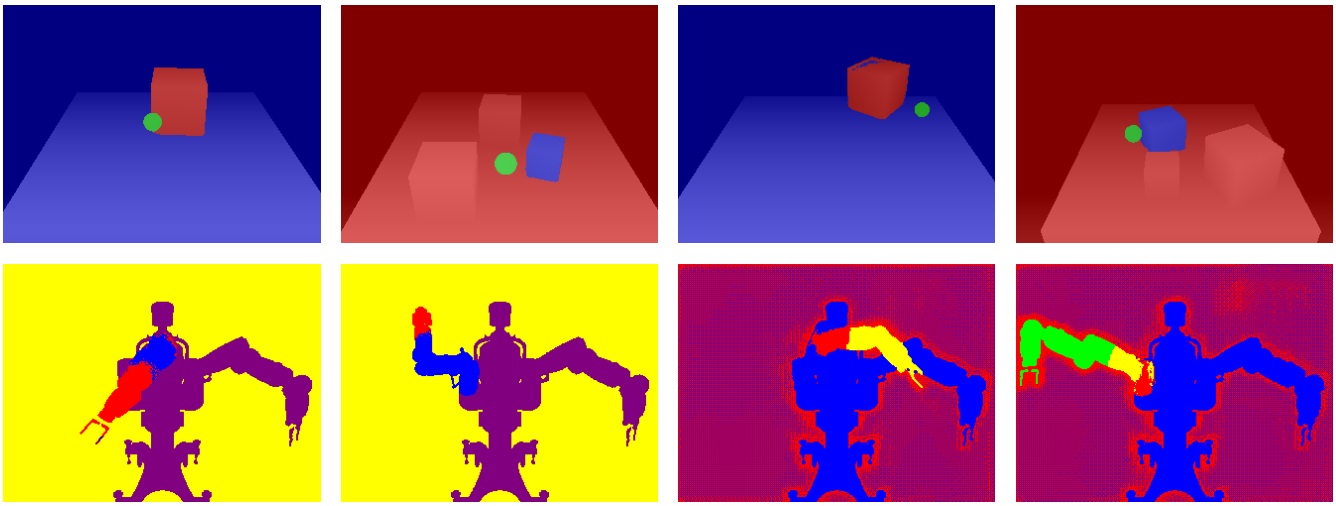

Fig. 5: Object labels predicted by the network rendered by an argmax across the k mask channels (in practice, the masks are close to binary). Left two columns show ground truth masks. First row shows combined results from the box datasets overlaid on the input depth images. Second row shows labels from the Baxter dataset. The system usually segments the robot arm into two distinct parts with a split near the elbow. Background is split across multiple classes, all of which predict an identity transform.

an explicit component that learns to fill-in/remove data. Fourth, a natural next step for **SE3-Nets** seems to be an extension to use temporal data. For example, disambiguating the ball from the box becomes simple given a sequence of point clouds where the ball first moves towards the box, hits it and moves away. This could also provide interesting solutions to solve all of the previously discussed problems. And last, we motivated this paper as a dynamics model to be used in conjunction with control. We believe that the current formulation of **SE3-Nets** is quite amenable to this and are looking towards integrating this system with a real-world robotic platform to perform interesting control tasks.

References

[1] Torch. URL http://torch.ch/.

[2] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015.

[3] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[4] Renée Baillargeon. Infants' physical world. *Current directions in psychological science*, 13(3):89–94, 2004.

[5] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45): 18327–18332, 2013.

[6] Byron Boots, Arunkumar Byravan, and Dieter Fox. Learning predictive models of a depth camera & manipulator from raw execution traces. In *IEEE International Conference on Robotics and Automation*, pages 4021–4028. IEEE, 2014.

[7] Brian Cheung, Jesse A Livezey, Arjun K Bansal, and Bruno A Olshausen. Discovering hidden factors of variation in deep networks. *arXiv preprint arXiv:1412.6583*, 2014.

[8] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[9] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. *arXiv preprint arXiv:1509.06113*, 2015.

[10] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015.

[11] Perle Geoffroy, Nicolas Mansard, Maxime Raison, Sofiane Achiche, and Emo Todorov. From inverse kinematics to optimal control. In *Advances in Robot Kinematics*, pages 409–418. Springer, 2014.

[12] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[13] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 44–51. Springer, 2011.

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

[15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2008–2016, 2015.

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Christof Koch and Shimon Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. In *Matters of intelligence*, pages 115–141. Springer, 1987.

[18] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2530–2538, 2015.

[19] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. *arXiv preprint arXiv:1603.01312*, 2016.

[20] Wenbin Li, Seyedmajid Azimi, Aleš Leonardis, and Mario Fritz. To fall or not to fall: A visual approach to physical stability prediction. *arXiv preprint arXiv:1604.00066*, 2016.

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[22] Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popovic, and Emanuel V Todorov. Interactive control of diverse complex characters with neural networks. In *Advances in Neural Information Processing Systems*, 2015.

[23] Roozbeh Mottaghi, Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. Newtonian image understanding: Unfolding the dynamics of objects in static images. *arXiv preprint arXiv:1511.04048*, 2015.

[24] Roozbeh Mottaghi, Mohammad Rastegari, Abhinav Gupta, and Ali Farhadi. " what happens if..." learning to predict the effect of forces in images. *arXiv preprint arXiv:1603.05600*, 2016.

[25] R. Newcombe, D. Fox, and S. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, 2015.

[26] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pages 2845–2853, 2015.

[27] Julian Quiroga, Thomas Brox, Frédéric Devernay, and James Crowley. Dense semi-rigid scene flow estimation from rgbd images. In *Computer Vision–ECCV 2014*, pages 567–582. Springer, 2014.

[28] Adam N Sanborn, Vikash K Mansinghka, and Thomas L Griffiths. Reconciling intuitive physics and newtonian mechanics for colliding objects. *Psychological review*, 120(2):411, 2013.

[29] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Dart: Dense articulated real-time tracking.

[30] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference, 2005. Proceedings of the 2005*, pages 300–306. IEEE, 2005.

[31] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[32] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pages 1049–1056. ACM, 2009.

[33] Niklas Wahlström, Thomas B Schön, and Marc Peter Deisenroth. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251*, 2015.

[34] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, pages 2728–2736, 2015.

[35] William Whitney. Disentangled representations in neural models. *arXiv preprint arXiv:1602.02383*, 2016.

[36] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*, 2015.

[37] Jiaji Zhou, Robert Paolini, J Andrew Bagnell, and Matthew T Mason. A convex polynomial force-motion model for planar sliding: Identification and application. 2016.