From Geometries to Contact Graphs*

Martin Meier[1], Robert Haschke[1], and Helge J. Ritter[1]

Neuroinformatics Group, CITEC, Bielefeld University
{mmeier,rhaschke,helge}@techfak.uni-bielefeld.de

Abstract. When a robot perceives its environment, it is not only important to know what kind of objects are present in it, but also how they relate to each other. For example in a cleanup task in a cluttered environment, a sensible strategy is to pick the objects with the least contacts to other objects first, to minimize the chance of unwanted movements not related to the current picking action. Estimating object contacts in cluttered scenes only based on passive observation is a complex problem. To tackle this problem, we present a deep neural network that learns physically stable object relations directly from geometric features. The learned relations are encoded as contact graphs between the objects. To facilitate training of the network, we generated a rich, publicly available dataset consisting of more than 25000 unique contact scenes, by utilizing a physics simulation. Different deep architectures have been evaluated and the final architecture, which shows good results in reconstructing contact graphs, is evaluated quantitatively and qualitatively.

Keywords: Physical reasoning · graph generation · robotics.

1 Introduction

Having knowledge about the relations of objects is important for everyday activities. For example, when taking a plate from a stack of dishes, we intuitively pick the one that is on top of the stack, since it has only contact with one of the other objects in the stack. This, for humans intuitive, knowledge is also crucial in the robotic domain when manipulation actions have to be planned and carried out. One way of specifying this knowledge that allows to infer relations between objects and their mutual contacts is in terms of a contact graph. To generate a contact graph, two major strategies can be distinguished. Either an agent actively explores its environment, for example by pushing objects around and track changes in their movements, or passively by capturing a snapshot of the environment and using an analysis pipeline to extract contact information based on rules. In this setting, the snapshot would be, for example, a 3D point cloud from a depth sensor. This point cloud needs to be further processed to extract segments from it, which in turn have to be fitted to object models, to

* The research reported in this paper has been supported by the German Research Foundation DFG, as part of Collaborative Research Center 1320 "EASE - Everyday Activity Science and Engineering". The research was conducted in subproject R05.

finally infer contact information from these objects by reproducing them in a virtual environment, e.g. a physics simulation, and extract their mutual contacts from the simulation. Alternatively, object hypotheses can also be generated by means of a neural network [11].

By creating a contact graph from object geometries and a set of predefined rules the work presented in [10] follows the latter, rule based line of approaches to generate a contact graph. They propose a formalism to generate motion plans from a sequence of transitions in this graph. Rosman and Ramamoorthy [13] used SVMs to segment point clouds into objects based on geometric separability and generated rule based relations between these objects. In [1], the authors pursue the idea of active exploration. They use contact graphs to plan planar pushing actions of a robot to arrange boxes in a desired pattern. Here the contact graph is constructed by letting the robot actively move the boxes around and register contacts between them. The work presented in [15] used a learned representation of contact events in a manipulation sequence that was carried out by human demonstrators and successfully applied it to a real robot.

Exploiting relational properties of data has recently been of major interest in the machine learning and robotics community. The authors in [4] used graphs constructed from spatial relations between objects in a classification task to predict actions performed by humans. The input graphs here are constructed based on prior knowledge. In [6], the authors used the graph structure in the underlying datasets with graph convolutional layers to facilitate semi supervised classification. They were able to outperform different graph clustering methods. The model presented in [2] introduced the term of interaction networks. These networks are able to predict future states in a 2D physics simulation by learning from object properties an their physical relations expressed in graph structures. By using the kinematic tree of different robots directly as an input graph to a network, the authors in [14] were able to learn physics based controllers for different robots. Exploiting the underlying graph structure of data also gained interest in the area of activity recognition. In [17], a networks was presented which is able to reconstruct the relations between different team members in a game of volleyball from video sequences. Here, the prediction of the relations is formulated by optimizing multiple cross entropies to obtain the most probable relations. The work in [12] internally uses a graph structure to keep track of activity descriptions over time when trying to estimate when and where in a video an actions has taken place.

In this paper, we present a neural network that is able to infer complex contact graphs only from geometric properties of physically stable object configurations. This extends rule-based approaches such as [10, 13], that rely on geometry alone, e.g. without considering the embedding in a physical situation with gravity and friction. We include such physical information through the way our training data set is generated, relying on the power of deep neural nets to implicitly extract the resulting correlations between physics and geometry for constructing accurate contact predictions. In our setting, we do not have a-priori knowledge of the underlying graph structure, but we want to predict it given

physical realistic geometric object configurations. We decided on an approach similar to [17], to let the network generate contact hypotheses. To this end, we created a novel dataset[1] of physically realistic object relations and corresponding contact graphs and evaluated different network architectures to facilitate contact graph generation from single examples.

Since one of the most tedious tasks when generating a dataset is acquiring high quality label information, we will describe our approach to automatically generate object relations and corresponding contact graphs in the following section. After that we lay out the design decisions for the network architectures we chose and evaluate them.

2 Data Generation and Preprocessing

To generate a sufficiently large dataset for training a neural network and also to obtain ground truth data, we employed the open source robot simulation gazebo [7]. The goal of our network is to realize a mapping:

$$\hat{\mathbf{A}} = f(\mathbf{F}, \theta) \tag{1}$$

where θ denote the network weights, $\mathbf{F}$ is a $n \times d$ matrix of input features (cf. below) and $\hat{\mathbf{A}}$ an estimate of the true contact graph $\mathbf{A}$, represented in the form of an undirected binary adjacency matrix of size $n \times n$ to accommodate contacts between up to n objects ($n = 10$ in our simulations).

To this end, we randomly generated scenes which contain between six and ten objects. Each scene is generated as the result of a physical process, modeling how an initially random configuration of physical objects above a planar support surface ("ground") comes to rest under its natural dynamics and the influence of gravity. Therefore, each of these objects is initialized with a randomized 6D position in space and random size along its x,y and z dimension. Mean and upper/lower bound of the uniformly distributed initialization values are shown in Table 1. We chose a smaller z dimension compared to x and y to assure that one dimension is small enough to be easily graspable by a robotic manipulator, so that the dataset could possibly be used in future experiments. Having one smaller dimension is also a common feature in most household objects, for example books or various kind of storage boxes for food. The friction between the simulated objects is set to a rather high value in simulation, a real world analogy would be bricks made of clay, to facilitate reaching a stable state in a shorter period of time by reducing unwanted sliding motions. The density of each object is also set to a value similar to bricks with $2 \frac{g}{cm^3}$.

It is important to note that the objects are initialized above the ground and in a consecutive fashion to prevent an overlap of objects during initialization. If two objects are initialized in an overlapping or penetrating state, physics simulators tend to estimate extremely high forces and the simulated scene has a high chance to "explode", being in an unrecoverable state afterwards.

[1] Dataset is available at https://pub.uni-bielefeld.de/record/2943056

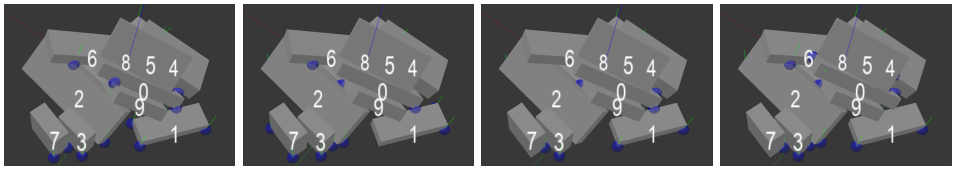
property	X	Y	Z
position	$0.0 \pm 0.1\,\mathbf{m}$	$0.0 \pm 0.1\,\mathbf{m}$	$0.4 \pm 0.1\,\mathbf{m}$
orientation	$\pi \pm \pi\,\mathbf{rad}$	$0.0 \pm 0.1\,\mathbf{rad}$	$\pi \pm \pi\,\mathbf{rad}$
size	$0.25 \pm 0.15\,\mathbf{m}$	$0.125 \pm 0.05\,\mathbf{m}$	$0.05 \pm 0.01\,\mathbf{m}$

Table 1. Mean and upper/lower bound of the uniformly distributed initialization parameter of generated objects. Position and size are in meters while the orientation is given in Euler xyz angles.

To obtain a stable state in the simulation, after initializing the objects, we let the simulation run for 15 seconds at a rate of 1000 simulation steps/second and evaluated the object movements afterwards. If the center of none of the objects moves further than $0.1mm$ in any direction within 100 simulation steps, we recorded ten simulation steps consisting of object locations and contact pairs between objects. These contact pairs were accumulated over the ten simulation steps. Accumulating contacts over multiple steps was necessary due to unstable contacts generated by the simulation. An example for a stable configuration, e.g. no object is moving, of ten objects is shown in Fig. 1. Although the objects are not moving, not all contacts are recognized during all time steps, for example the contact between object two and six is not present in each of the four consecutive frames of the simulation shown in Fig. 1. This is due to the technique of simulation engines to add small noise internally to all object states, to avoid running into numerical instabilities. Recording ten consecutive steps proved to be sufficient to collect all contact pairs between objects. We ran simulations for nearly a week and were able to gather 25116 samples of training data, consisting of object locations and their pairwise contacts.

From the recorded samples of contacts, we generate the binary object adjacency matrices $\mathbf{A}$ by checking if there exists a contact for a given object pair within the recorded ten simulation steps. If this is the case, we set the corresponding element in the adjacency matrix to 1, 0 otherwise. Contacts between objects and the ground plane in the simulation were omitted, since most of the objects have contacts with the ground and these connections would be over represented compared to object-object contacts.

As our object features we used shape (3D elongation along x,y,z object edges), spatial position (3D) and orientation (represented as quaternions with 4 parameters). By having at most ten objects, we obtain a 10×10 feature matrix $\mathbf{F}$ consisting of object features $\times$ the number of objects. Since we generated the scenes with a varying number of objects, rows with a higher row index have a higher probability to consist only of zeroes. To circumvent this bias and to augment the dataset, we applied random permutations to the rows of the feature matrix $\mathbf{F}$ and the same permutations to the rows and columns of the corresponding adjacency matrix $\mathbf{A}$, to preserve the symmetry of $\mathbf{A}$. We applied 32 permutations to each pair $\{\mathbf{F}, \mathbf{A}\}$ which, together with the original pair, leads to a total of 828828 training samples.



(a) Sim step 1. (b) Sim step 2. (c) Sim step 3. (d) Sim step 4.

Fig. 1. Contacts calculated by the physics simulation for a stable configuration of ten objects. Contacts between objects are depicted as blue spheres. During four consecutive simulation steps, not always the same contacts are calculated. This is due to numerical instabilities and added noise by the simulation engine.

3 Learning Contact Relations

The goal of this work is to learn a mapping from a set of physical object configurations to a binary adjacency matrix that indicates their mutual contacts. To this end, the natural idea is to express our learning problem as a multi-label classification task and use binary cross entropy

$$\mathcal{L}(\mathbf{A}, \hat{\mathbf{A}}) = -\frac{1}{n * n} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(a_{ij} \log \hat{a}_{ij} + (1 - a_{ij}) \log(1 - \hat{a}_{ij})\right) \tag{2}$$

as loss function, where a_{ij} are the elements of $\mathbf{A}$ and $\hat{a}_{ij}$ are the elements of the predicted adjacency matrix $\hat{\mathbf{A}}$. We also investigated using Focal Loss[8] and reconstruction with *mean squared error* as losses, but these did not lead to better results than binary cross entropy, details are in the next section.

Since our generated dataset contains, on average, 20% positive entries in the adjacency matrices while the remaining 80% entries are zero, it is not sensible to use a straightforward accuracy metric to evaluate our model, because the dataset is not sufficiently balanced. Therefore, we are using the *Area under the ROC curve* (AUC) as metric in the following evaluation [3]. Here, *ROC* is the receiver operating characteristic, which is defined by the ratio of the true positive rate to the true negative rate of a classifier.

To obtain a network architecture which is able to reconstruct an adjacency matrix from our dataset and to ensure it learns some kind of dense representation, we performed a reverse ablation study. Since we have a 100 dimensional output with sigmoid activation to match the binary cross entropy loss, we want the second to last layer to contain less neurons than the output, to facilitate the learning of a denser representation than the original input. We started with a single hidden layer with 64 neurons and ReLU (rectified linear unit) activation, and successively added hidden layers of increasing size with ReLU activation to the model, while keeping the 64 unit layer always as the second to last layer, to enforce the learning of a dense representation in this layer. The final reconstruction of the adjacency matrix from the last layer is done by applying a threshold

# hidden layers	1	2	3	4	5	6	7
units in first hidden layer	64	96	128	192	256	384	512
AUC score	0.6623	0.7393	0.8146	0.8425	0.8629	0.8653	0.8647

Table 2. Evaluation of different deep architectures. The table is read column wise. For example, a network with three hidden layers has the architecture of input-128-96-64-output and a validation AUC score of 0.8164 after training.

of 0.5 to the output of the sigmoid units, to obtain binary values. Since our dataset contains more than 800k samples, we followed the idea to increase the batch size instead of decreasing the learning rate [16], to facilitate faster training. Starting from a batch size of 64, we double the batch size every 20 epochs up to a final batch size of 2048. All trainings have been carried out using the Adam optimizer [5]. From the 800k samples, 20% were kept from training to perform the evaluation. Table 2 shows the seven deep architectures we tested. The table is read column wise. For example, if the network has four hidden layers, the first hidden layer has 192 ReLU units and the network achieved a final AUC score of 0.8425. As can be seen in the table, the AUC score increases with each additional hidden layer until it reaches a plateau at five hidden layers. We therefore decided to do a thorough evaluation on a network with five hidden layers with 256-192-128-96-64 units.

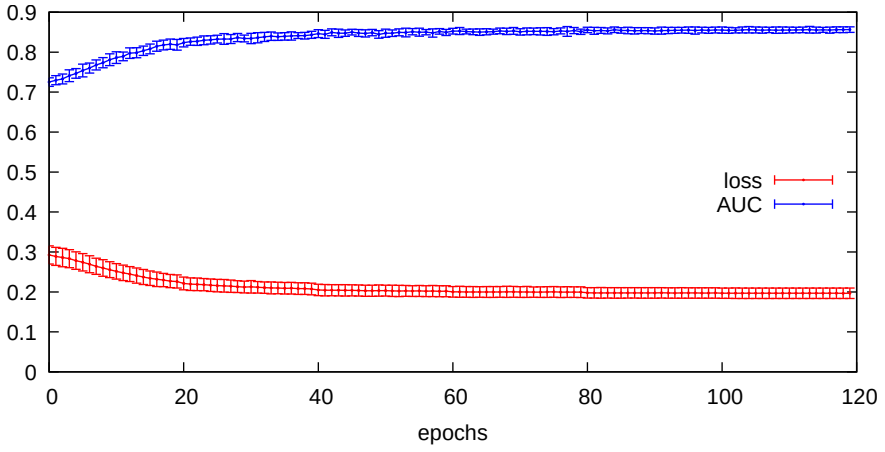
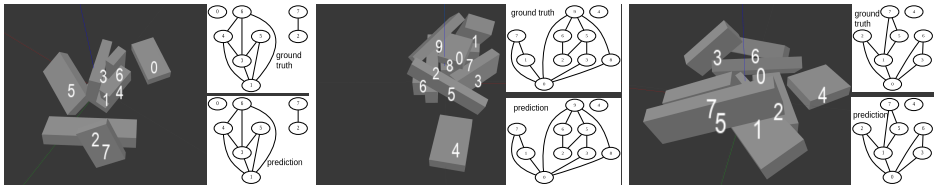


Fig. 2. Results of a 5-fold cross validation using the proposed model from section 3. Starting from an initial batch size of 64, the batch size is doubled every 20 epochs. The values shown are mean and standard deviation for AUC score and loss at the end of each epoch, for the evaluation sets of the five runs.



(a) Sample 24745. (b) Sample 334. (c) Sample 14566.

Fig. 3. Example scenes from the dataset where our network achieved perfect results. Each panel shows a screenshot of the scene and the corresponding contact graphs.

4 Evaluation

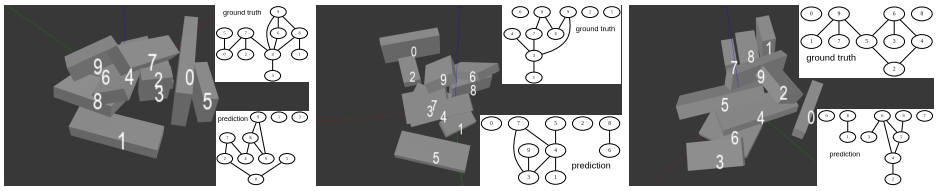
We evaluated the selected model in a 5-fold cross validation. The AUC score reaches a final value of $\mu = 0.8564, \sigma = 0.007$, with a minimum of 0.8495 and a maximum of 0.8709, the training progress is shown in Fig. 2. The increase in AUC score reaches a plateau around epoch 60, where the batch size is increased to 512. As a comparison, using focal loss with these settings leads to a final AUC score of 0.7719. This could be due to the fact that focal loss is tailored towards shaping the underlying binary cross entropy towards rewarding very rare occurring features, which could be too extreme in the dataset at hand. Using mean squared error as the loss function leads to a final AUC score of 0.8146.

We further evaluated the impact of our dataset augmentation to the achievable AUC score. To this end, we started with the original dataset and successively doubled the number of permuted samples, which are added to the training set. The results are shown in table 3. Here we can see that the augmentation proves to be beneficial for the task. Especially the comparison of no permutation to one is a strong hint that it is important to permute the feature matrices to reduce the bias which originates from possible empty feature vectors. Also in this case, the gain from adding more permutations becomes neglectable around 64 permutations, which is a good indicator that our chosen settings of 32 is a good trade-off between training duration and achieved AUC score. To get a qualitative insight into the results of our trained model, we randomly selected successful reconstructions of contact graphs and the corresponding scenes from our dataset as well as reconstructions that partially failed, i.e. contain missing or wrong links between objects. The successful reconstructions are shown in Fig. 3. Here the prediction contact graph always matches the ground truth. The partially failed examples are shown in Fig. 4. Here it can be seen that our model has problems with corner to surface contacts, for example between objects 7 and 0 in the left panel, 8 and 7 in the center panel and 7 and 9 in the rightmost panel. Also, edge to surface contacts seem to be slightly over estimated in some cases, there is a non existing contact added between object 8 and 1 in the rightmost panel and between object 5 and 4 in the central panel.

We further investigated the behavior of our model in a successively changing simulation. To this end, we manually constructed a scene that resembles a

permutations	0	1	2	4	8	16	32	64	128	256
AUC score	0.6647	0.7503	0.7693	0.7805	0.8177	0.8253	0.8511	0.8532	0.8534	0.8506

Table 3. Evaluation of the dataset augmentation in terms of added permutations to the dataset. The initial increase in performance is compelling in the lower numbers of permutations but reaches a plateau around 64, using the proposed network architecture.

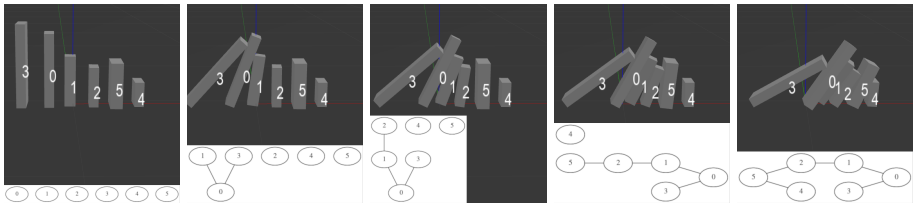


(a) Sample 23710. (b) Sample 414. (c) Sample 327.

Fig. 4. Example scenes from the dataset where our network partially failed to reconstruct the contact graph. In the top right of each panel, the ground truth contact graph is shown. The reconstruction is in the lower right. For discussion please refer to the text.

domino effect, as shown in Fig. 5. We pushed the leftmost box to tip over the remaining five boxes and took snapshots of the object poses when other blocks started to fall over. The resulting contact graphs are shown at the bottom of each panel. The graphs represent the sequence of the blocks contacting their neighbors and create a chain that connects all blocks, eventually.

Additionally, we investigated the activations in the last hidden layer in our network. Fig. 6 shows the same TSNE [9] embedding of activations for each sample in our dataset, colored with respect to two different properties in our dataset. In the left panel, the embedded activations are colored according to the number of objects in the input feature. Here clear clusters are visible, where the

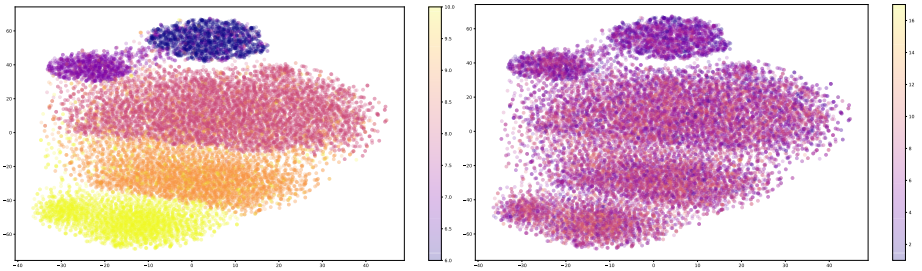


(a) Snapshot 1. (b) Snapshot 2. (c) Snapshot 3. (d) Snapshot 4. (e) Snapshot 5.

Fig. 5. Example for a manually constructed "domino" sequence. In the beginning, six objects are arranged in a straight line. The leftmost object is tilted to initiate tipping over the other objects. The corresponding contact graph generated by our neural network is shown at the bottom of each panel.

yellow color at the bottom of the figure indicates six objects in the input while the dark blue at the top indicates ten objects.

The right panel is colored according to the number of contacts in the output contact graphs. Here only a slight tendency is visible, the lighter coloring at the bottom indicates less contacts. This correlates with the number of objects in the left panel, since samples with less objects have a tendency to also have less contacts.



(a) Colored wrt. number of objects. (b) Colored wrt. number of contacts.

Fig. 6. 2D TSNE embedding of the activations in the last hidden layer. This figure shows the same embedding colored with respect to two different properties of our dataset. The left panel is colored based on the number of objects in the input feature while the right panel is colored wrt. the number of contacts in the input feature.

5 Conclusion

We considered the task of predicting contact graphs from object configurations through a deep neural network and presented a dataset which represents rich physical contact situations. We compared network architectures of different depths for solving this task, considering it as a multi-label classification task from geometric object features to the elements of an adjacency matrix that describes the contact graph. We showed that for the resulting, unbalanced multi-label classification task with our dataset, an optimization based on binary cross entropy is superior to least squares or focal loss in term of the respective AUC score and present typical examples and TSNE embeddings to provide some insight into the properties of the network solution. The used input representation is well tailored to become part of a point cloud processing pipeline leading from a depth map of a cluttered arrangement of 3D objects to a high-level contact representation of their physical configuration. Such a pipeline can enable a recording of rich contact episodes and help to guide every day manual actions of robots in unstructured environments.

References

1. Anders, A.S., Kaelbling, L.P., Lozano-Perez, T.: Reliably arranging objects in uncertain domains. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 1603–1610. IEEE (2018)
2. Battaglia, P., Pascanu, R., Lai, M., Rezende, D.J., et al.: Interaction networks for learning about objects, relations and physics. In: Advances in neural information processing systems. pp. 4502–4510 (2016)
3. Chawla, N.V.: Data mining for imbalanced datasets: An overview. In: Data mining and knowledge discovery handbook, pp. 875–886. Springer (2009)
4. Dreher, C.R., Wächter, M., Asfour, T.: Learning object-action relations from bimanual human demonstration using graph networks. IEEE Robotics and Automation Letters **5**(1), 187–194 (2019)
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
7. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). vol. 3, pp. 2149–2154. IEEE (2004)
8. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
9. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
10. Najafi, E., Shah, A., Lopes, G.A.: Robot contact language for manipulation planning. IEEE/ASME Transactions on Mechatronics **23**(3), 1171–1181 (2018)
11. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)
12. Rashid, M., Kjellstrom, H., Lee, Y.J.: Action graphs: Weakly-supervised action localization with graph convolution networks. In: The IEEE Winter Conference on Applications of Computer Vision. pp. 615–624 (2020)
13. Rosman, B., Ramamoorthy, S.: Learning spatial relationships between objects. The International Journal of Robotics Research **30**(11), 1328–1342 (2011)
14. Sanchez-Gonzalez, A., Heess, N., Springenberg, J.T., Merel, J., Riedmiller, M., Hadsell, R., Battaglia, P.: Graph networks as learnable physics engines for inference and control. In: International Conference on Machine Learning. pp. 4470–4479 (2018)
15. Scherzinger, S., Roennau, A., Dillmann, R.: Contact skill imitation learning for robot-independent assembly programming. In: IEEE International Conference on Intelligent Robots and Systems. pp. 4309–4316. IEEE (2019)
16. Smith, S.L., Kindermans, P.J., Ying, C., Le, Q.V.: Don't decay the learning rate, increase the batch size. arXiv preprint arXiv:1711.00489 (2017)
17. Wu, J., Wang, L., Wang, L., Guo, J., Wu, G.: Learning actor relation graphs for group activity recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9964–9974 (2019)