Inductive Biases for Predicting Deformation and Stress in Deformable Object Grasps with Graph Neural Networks

Frederik Heller[1], Alap Kshirsagar[1], Tim Schneider[1,2], Guillaume Duret[1,2] and Jan Peters[1,3,4]

***Abstract*— Humans handle and manipulate soft, deformable objects effortlessly, but robot skill in this domain lags behind. One of the major challenges in robotic manipulation of deformable objects arises from the difficulty of predicting the deformation and stress fields. The gold standard for modeling the physics of deformable objects, and predicting deformation and stress fields, is the computation-heavy Finite Element Method (FEM). Recent advances such as Graph Neural Networks (GNNs) enable learning such fields with high accuracy. We base our work on the DefGraspNets model, of which we identify key limitations: First, the network predicts stress values at mesh vertices, which is not in line with the physical model of FEM. Second, high mesh resolution and low number of message passing rounds prohibit propagation of information through the entire graph, which hurts performance in edge cases. To overcome these limitations, we propose two modifications as inductive biases to the GNN: Tetrahedron features for predicting values directly at tetrahedrons, and a global feature as shortcut for information relevant to the whole graph. Our results, evaluated with FEM-simulated datasets of grasps on different objects, show that our method outperforms the baseline on nearly all objects, enabling more accurate and physically more realistic predictions.**

We release our codebase: fheller1.github.io/tetgraspnets

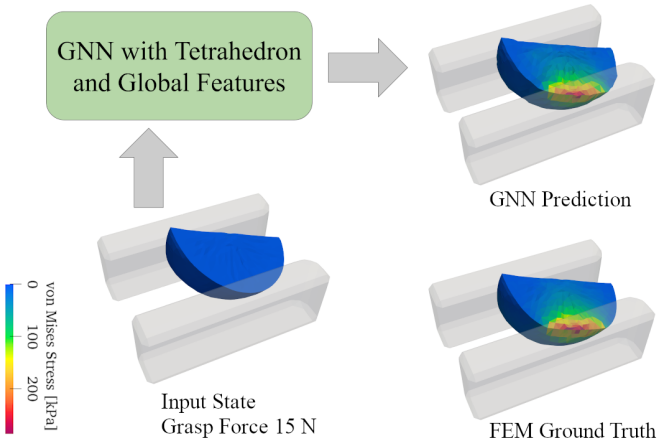

Fig. 1: Our implemented models are trained on FEM simulations of grasps on deformable objects, such as this lemon slice. Extended by tetrahedron features and a global feature as inductive biases, the GNN model accurately predicts node deformation and tetrahedron stress fields.

I. Introduction

Deformable objects are omnipresent to us humans, and we instinctively know how to grasp and handle them. Robotic systems today often still lack this intuitive judgement. While the dynamics of deformable objects can be simulated accurately using the Finite Element Method (FEM), it comes with a high computational burden and is too slow for robotic systems with real-time capability. Recent work, such as *DefGraspNets* by Huang et al. [1], attempts to close this gap by training a Graph Neural Network (GNN) architecture to predict deformation and stress fields in grasps on deformable objects, with data generated by an FEM simulator [2]. They showed that accurate predictions of deformation and stress fields are possible at a speedup of several orders of magnitude compared to FEM. For this, Huang et al. used the GNN architecture from the work *MeshGraphNets* [3], and express initial gripper and object state as input feature graph. Vertices of gripper and object mesh are the nodes in this graph, and the mesh edges are graph edges. Each node feature tells if the node belongs to gripper or object, and in which direction it moves, and mesh edge features give the 3D and scalar edge displacement and material parameters. Additionally, world edges are drawn between close object and gripper nodes to enable information flow between object and gripper, and their features give edge displacement and the force that the gripper closes with. Then, an *Encode-Process-Decode* GNN block performs the forward pass: After encoding all features to a common latent space using Multi-Layer Perceptrons (MLPs), multiple rounds of message passing propagate information through the graph. In each message passing round, nodes receive messages from connected edges, and an MLP processes these to obtain an updated node feature. Similarly, edge features are updated by an MLP receiving features of connected nodes as input. Finally, a decoder MLP decodes predictions of deformation and stress values resulting from the grasp at each node.

During our initial experimentation with the *DefGraspNets* code, training and evaluating on different grasped objects, we realized the following shortcomings or limitations:

1) It appears the released codebase [1] is an incomplete version, without e.g. the preprocessing of FEM data;
2) FEM computes stresses at tetrahedron elements, but the model predicts stress as values at nodes;
3) Due to limited message passing rounds, not all information is propagated through the entire graph.

[1]Technische Universität Darmstadt, Department of Computer Science, Intelligent Autonomous Systems Group. Correspondence: frederik.heller@stud.tu-darmstadt.de
[2]École Centrale de Lyon, LIRIS Group
[3]German Research Center for AI (DFKI)
[4]Hessian Center for Artificial Intelligence (Hessian.AI)
We thank Hessisches Ministerium für Wissenschaft & Kunst for the DFKI grant and "The Adaptive Mind" grant.

To address the first limitation, we reimplemented the *DefGraspNets* baseline, moving from TensorFlow to PyTorch. The second limitation implies that, in preprocessing, the FEM stress field given at tetrahedrons must be sampled at nodes by averaging the stresses of adjacent tetrahedrons. This would blur peaks in the stress field and hurt the accuracy of obtained predictions, especially since peak stress is an important measure. For a physically more accurate stress prediction, we propose to inform the GNN of mesh tetrahedrons, allowing prediction of values directly in tetrahedrons. The third limitation is especially evident in grasps on corners of the object, from where no information can reach the opposite corner. To address it, we propose to add a global feature [4] to the GNN, which accumulates and distributes globally relevant information through the entire graph.

II. Methodology

The efforts in implementing our methods started with the reimplementation of *DefGraspNets*. For the details of its node and edge feature constrution, we refer to [1], for node and edge updates in message passing to [3].

A. Tetrahedral Features and Stress Prediction

Our first proposed architectural extension informs the GNN about tetrahedrons in the object mesh. Similar to edges in [1], [3], a tetrahedron set is added to the input feature graph representation, consisting of four-tuples $\mathbb{T}_i$ that give corresponding node indices for each mesh tetrahedron i. Additionally, for each tetrahedron, an input feature vector $\mathbf{t}_i$ is given. First, the tetrahedral features are encoded into the common 128D latent space by an MLP f^t_{enc},

$$\tilde{\mathbf{t}}_i = f^t_{\text{enc}}(\mathbf{t}_i).$$

For all tetrahedrons, the input $\mathbf{t}_i$ is scalar zero, hoping that all relevant information can be accumulated in message passing. This means the encoding step could be skipped and the latent feature initialized with zeros instead, but we keep it in our implementation to permit for future extensions.

In each message passing round, the update for the latent feature of each tetrahedron is computed as following: An update MLP f^t_{upd} is given the current latent tetrahedron feature $\tilde{\mathbf{t}}_i$, and the latent feature $\mathbf{v}_j$ of each node j that is part of the tetrahedron. Then, the updated feature is given by the MLP output with a residual connection:

$$\tilde{\mathbf{t}}'_i = f^t_{\text{upd}}\left(\tilde{\mathbf{t}}_i, \{\tilde{\mathbf{v}}_j\}_{j\in\mathbb{T}_i}\right) + \tilde{\mathbf{t}}_i$$

The information flow to the tetrahedron feature is shown in Fig. 2. This update rule lets information about the behavior of nodes flow into each tetrahedron feature. A decoder MLP f^t_{dec} finally decodes a prediction of the scalar von Mises stress $\hat{\sigma}_i$ within each tetrahedron,

$$[\hat{\sigma}_i] = f^t_{\text{dec}}(\tilde{\mathbf{t}}'_i).$$

The design choice that node information flows into tetrahedrons, but not back, is motivated by how FEM derives the stress field after computing node deformations.

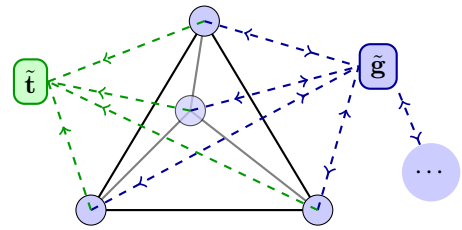

Fig. 2: The proposed inductive biases function during the message passing phase, illustrated for a single tetrahedron. In green: A tetrahedron feature receives messages from each of the tetrahedrons nodes. In blue: A global feature receives and sends messages from and to all nodes not only in this tetrahedron, but the entire graph.

B. Global Feature and Rigid Transformation

Addressing the problem of limited propagation through the graph, we modify the GNN architecture with a global feature [4]. It can be imagined as a node that is connected to all other graph nodes, and thus can accumulate and broadcast globally relevant information to all nodes. An input global feature vector $\mathbf{g}$ is transformed to its latent representation by an encoder MLP f^g_{enc}:

$$\tilde{\mathbf{g}} = f^g_{\text{enc}}(\mathbf{g}).$$

Again, we decide on a zero input vector $\mathbf{g} = 0$. In each message passing round, the update for the latent global feature is computed by an update MLP for the global node f^g_{upd}, given the mean of all node features as input, together with a residual connection:

$$\tilde{\mathbf{g}}' = f^g_{\text{upd}}\left(\frac{1}{\#\text{nodes}}\sum_{i=1}^{\#\text{nodes}}\tilde{\mathbf{v}}_i\right) + \tilde{\mathbf{g}}$$

To allow for information to flow back from the global to node features, the node update rule from [3] is modified to take the global feature as additional input,

$$\tilde{\mathbf{v}}'_i = f^v_{\text{upd}}\left(\tilde{\mathbf{v}}_i, \sum_j \tilde{\mathbf{e}}_{ij}, \tilde{\mathbf{g}}\right) + \tilde{\mathbf{v}}_i.$$

The flow of information to and from the global feature is illustrated in Fig. 2. Finally, enabling prediction of values relevant to the entire graph, after message passing the latent global feature is decoded into an output feature

$$[\hat{\mathbf{t}}_{\text{rigid}}, \hat{\mathbf{r}}_{\text{rigid}}] = \hat{\mathbf{g}} = f^g_{\text{dec}}(\tilde{\mathbf{g}}').$$

As demonstration, we use this decoded global output feature as a prediction for the best-fit rigid body transformation of the object during the grasp, given by a 3D translation vector $\hat{\mathbf{t}}_{\text{rigid}}$ and a 6D rotation representation $\hat{\mathbf{r}}_{\text{rigid}}$. We choose this continuous representation, as proposed by Zhou et al. [5], for improved training performance. The ground truth best-fit rigid body transformation is obtained with singular value decomposition of the covariance matrix between original and deformed object points [6], [7].

TABLE I: Achieved MAPE score (lower is better) on predicted deformation $\mathbf{u}$, stress $\boldsymbol{\sigma}$, and 9D rigid transformation $\mathbf{p}$ fields for each model variant, and each object trained and validated on.

Object	8polygon06			cylinder07			lemon01			potato2			sphere03			strawberry01		
MAPE in %	$\mathbf{u}$	$\boldsymbol{\sigma}$	$\mathbf{p}$	$\mathbf{u}$	$\boldsymbol{\sigma}$	$\mathbf{p}$	$\mathbf{u}$	$\boldsymbol{\sigma}$	$\mathbf{p}$	$\mathbf{u}$	$\boldsymbol{\sigma}$	$\mathbf{p}$	$\mathbf{u}$	$\boldsymbol{\sigma}$	$\mathbf{p}$	$\mathbf{u}$	$\boldsymbol{\sigma}$	$\mathbf{p}$
Baseline [1]	3.24	1.52	–	3.00	3.23	–	4.66	1.39	–	**3.05**	2.46	–	**2.37**	1.37	–	3.34	**1.12**	–
A: Tet. Features	3.86	1.91	–	2.93	2.37	–	4.51	1.92	–	3.34	1.75	–	2.71	**1.18**	–	4.05	4.57	–
B: Tet. + Global Feat.	**2.04**	**0.93**	**35.4**	**1.84**	**1.04**	**37.4**	**3.48**	**0.77**	**4.43**	3.12	**0.53**	**4.70**	3.24	1.21	**25.9**	**3.16**	3.40	**10.8**

C. Dataset and Training Procedure

Using *DefGraspSim* [2], we generated a ground truth dataset of deformations and stresses resulting in grasps on deformable objects. For each of six used different objects (shown in the header of Table I), this includes 100 different grasps, each including 50 states with increasing grasp force. 80 grasps were used as training data, and 20 for testing, sampled randomly once and then kept fixed for all variants.

To evaluate our methods, we trained three different model variants on each object: Variant A is our reimplemented *DefGraspNets* baseline, B adds tetrahedral features. Model variant C combines tetrahedral features with the global feature. The models were trained on normalized targets, and as loss function the sum of Mean Squared Error on each predicted field was used. On the checkpoint with lowest test loss, the Mean Absolute Percentage Error (MAPE) on each predicted field was recorded. The choice of MAPE is motivated by the different scale of observed deformation and stress per object, and we expect it to yield better comparable metrics than e.g. Mean Absolute Error.

III. Results

For each combination of training object and model variant, Table I presents the achieved MAPE metrics at the checkpoint with lowest test loss. Table II provides the error metrics for each model variant averaged over all objects.

The baseline and model variant A have similar error metrics on all objects, suggesting that the addition of tetrahedral features does not clearly outperform the baseline. This could be attributed to both variants suffering from the same limitation—information not reaching nodes far away—that may dominate the error term. Only for strawberry01, we see significantly smaller MAPE on the stress field with the baseline model, which we believe to be an outlier.

Model variant B, however, shows significantly smaller errors on the majority of objects, both for deformation and stress prediction. We deduce that, indeed, information not reaching parts of the graph posed a serious limitation that the proposed global feature overcomes. Only on sphere03, variant B does not improve on the baseline in deformation or stress. We hypothesize that this is due to the highly regular shape, which results in low variance in sampled grasps. This way, nodes far from the gripper show little movement, and accurate prediction is possible without global feature. Variant B is also the only architecture able to predict the best-fit rigid transformation, though with relatively high MAPE.

TABLE II: Average MAPE (lower is better) on deformation $\mathbf{u}$, stress $\boldsymbol{\sigma}$, and 9D rigid transformation $\mathbf{p}$ across all objects, for each model variant.

MAPE in %	$\mathbf{u}$	$\boldsymbol{\sigma}$	$\mathbf{p}$
Baseline [1]	3.28	1.85	–
A: Tet. Features	3.57	2.28	–
B: Tet. + Global Feat.	**2.81**	**1.31**	**19.77**

IV. Conclusion

We reimplemented the *DefGraspNets* [1] model to a working state to support reproducibility and future research on learning dynamics of grasps on deformable objects. We examined the model architecture to reveal key limitations, and addressed these with two proposed inductive biases as architectural extensions: tetrahedron features for stress prediction in line with FEM, and a global feature acting as shortcut for information through the entire graph. Our model variant combining tetrahedral stress prediction and a global feature clearly improves on the baseline, as shown by our evaluation using grasps on different object geometries. While the model variant using tetrahedron features alone did not improve metrics over the baseline, we believe they are an important novel extension to the GNN architecture. The tetrahedron features let predictions be decoded directly at tetrahedron elements, which allows predicting stress accurately in line with the physical model of the FEM. To further prove the advantages, we suggest that future research should experiment with encoding material properties in input tetrahedron features, again in line with FEM. With message passing adjusted to propagate information from tetrahedron features also back to nodes, such a model could generalize to objects with non-isotropic material. For this, a simulated larger dataset of grasps on non-isotropic objects with different geometries would have to be generated.

Acknowledgment

We thank Dr. Isabella Huang for providing her *DefGraspNets* [1] dataset for the evaluation of our methods.

This work was supported by the French Research Agency, l'Agence Nationale de Recherche (ANR), and the German Federal Ministry of Education and Research (BMBF) through the project Aristotle (ANR-21-FAI1-0009-01).

REFERENCES

[1] I. Huang, Y. Narang, R. Bajcsy, F. Ramos, T. Hermans, and D. Fox, "DefGraspNets: Grasp planning on 3D fields with graph neural nets," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5894–5901.

[2] I. Huang, Y. Narang, C. Eppner, B. Sundaralingam, M. Macklin, R. Bajcsy, T. Hermans, and D. Fox, "DefGraspSim: Physics-based simulation of grasp outcomes for 3D deformable objects," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6274–6281, 2022.

[3] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. Battaglia, "Learning mesh-based simulation with graph networks," in *International conference on learning representations*, 2020.

[4] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[5] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5745–5753.

[6] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, 1987.

[7] O. Sorkine-Hornung and M. Rabinovich, "Least-squares rigid motion using SVD," Department of Computer Science, ETH Zurich, Tech. Rep., Jan. 2017. [Online]. Available: https://igl.ethz.ch/projects/ARAP/svd_rotation.pdf