DoughNet: A Visual Predictive Model for Topological Manipulation of Deformable Objects

Dominik Bauer[1], Zhenjia Xu[1,2], and Shuran Song[1,2]

[1] Columbia University
[2] Stanford University
Corresponding Author: dominik.bauer@columbia.edu

Abstract. Manipulation of elastoplastic objects like dough often involves topological changes such as splitting and merging. The ability to accurately predict these topological changes that a specific action might incur is critical for planning interactions with elastoplastic objects. We present DoughNet, a Transformer-based architecture for handling these challenges, consisting of two components. First, a denoising autoencoder represents deformable objects of varying topology as sets of latent codes. Second, a visual predictive model performs autoregressive set prediction to determine long-horizon geometrical deformation and topological changes purely in latent space. Given a partial initial state and desired manipulation trajectories, it infers all resulting object geometries and topologies at each step. Our experiments in simulated and real environments show that DoughNet is able to significantly outperform related approaches that consider deformation only as geometrical change. Our code, data and videos are available at https://dough-net.github.io.

Keywords: Geometric Deep Learning · Robotic Manipulation

1 Introduction

From kneading to cutting, our interactions with elastoplastic objects such as dough or clay constantly involve actions that, beyond their overall geometry, manipulate the topology of objects. For example, changing the number of components allows to portion a roll of dough. By increasing its genus, we may form a doughnut from it. Different end-effector (EE) geometries applying the same action to the same object, however, may yield drastically different topologies.

In this paper, we focus on this challenging task of topological manipulation with a learned visual predictive model, illustrated in Fig. 1. Specifically, we infer the sequence of geometrical *and* topological changes of an elastoplastic object – given a single RGB-D observation of its initial geometry, EE geometry, and planned actions. Such a visual predictive model equips robots with the ability to plan their actions and choice of EE tools for achieving diverse topological manipulation goals, while still forming the desired object geometries.

Most prior works on elastoplastic manipulation represent objects by a single global geometry [18, 32, 36, 37], or rely on additional post processing [12, 13, 24]

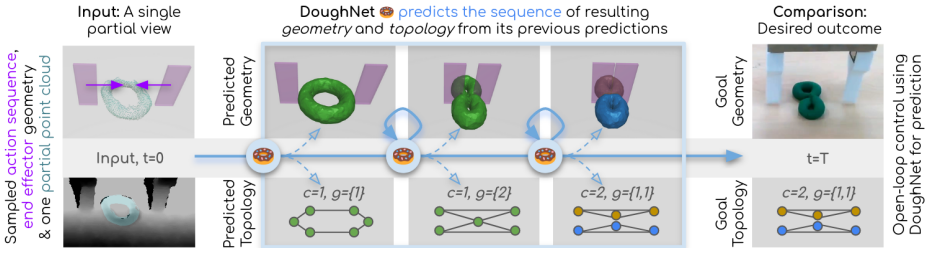

Fig. 1: Topological Manipulation. We predict the outcome of a series of actions from a single partial view. Beyond deformation, DoughNet considers topological changes (components c and genus g) conditioned on the end-effector geometry.

to handle a specific topological change (i.e., splitting into two components); they are unable to predict more general forms of topological change. We conjecture that the exclusive focus on object geometry prevents a deeper investigation of topological manipulation. In the example of doughnut forming, while the geometry may be very similar, a roll with *touching* ends insufficently solves the task. Rather, they must also stay *dynamically* connected when the object is moved. Determining these dynamic changes is, however, non-trivial. In the real world, conducting such a "topology check" destroys the geometrical state. In simulation, particle-based approaches [13,14,25] may implicitly represent topology but the information is not readily available; mesh-based approaches [2, 6, 12] have explicit access to the topology but no trivial way to change it.

Aiming at a general solution to these challenges, we present **DoughNet**, a visual predictive model that infers objects' geometrical deformation and topological changes. DoughNet consists of two main parts: First, an **autoencoder** that represents geometries of different topological connectivity as sets of latent codes. These codes may be decoded into occupancy maps for connected components, for each of which we predict its genus. Second, a **topology-aware dynamics model** learns the objects' geometrical deformation and topological changes in an autoregressive set prediction task. Multi-step prediction is performed completely in the learned latent space, with both parts trained exclusively on simulated data. To generate training data with ground-truth topological structure, we propose a set of topological-checking operations for particle-based simulation. They enable us to reliably determine *dynamic connectivity* between (merging, splitting) and within components (self-merging). Given a partial observation from a single RGB-D camera and a desired EE trajectory, DoughNet allows to determine all resulting geometries and topologies to guide topological manipulation.

In summary, our main contribution is DoughNet, a visual predictive model for the novel task of topological manipulation of elastoplastic objects. It is enabled by 1) a topology-aware dynamics model that jointly reasons about objects' geometrical deformation and topological changes under different physical interactions, and 2) a synthetic data generator that yields volumetric geometry and topological structure for training such models. Our experiments in simulated and real robotic environments show that DoughNet outperforms previous approaches for predicting manipulation of deformables, especially in long-horizon tasks.

2 Related Work

We compare related predictive models for deformable objects along the axes of time, as well as geometry and topology.

Temporal Abstraction. While simulation step sizes may critically affect predictions, learning-based methods may explore different temporal abstractions.

One end of the spectrum is to consider whether a skill will eventually achieve the desired goal [23,24,43]. While this prevents error propagation of incremental prediction, it requires single-step prediction of high-DoF deformation. Differentiable simulation may allow to improve upon these predictions [18,43].

On the other end of the temporal spectrum, Li et al. [20,21] train a GNN with data from particle-based simulation, predicting the corresponding graph's deformation in small increments. Similarly, Shi et al. [36,37] predict manipulation using different tool geometries. MPM simulation [38] is the template for a NeRF-based [28] dynamics model in [19]. Our approach yields more consistent shapes over longer manipulation horizons than incremental point-based predictions by learning a direct mapping between latent shapes at different time steps.

Spatial Abstraction and Topological Change. Whereas most analytical methods use particle, grid or tetrahedral representations, learned predictive models may be built upon a range of explicit and implicit shape representations.

Considering depth observations as input, for example, a common representation choice are point clouds and derived neighborhood graphs [17,20,21,26,36,37]. While points may diffuse over longer prediction horizons, the added regularization due to nearest-neighbor graphs or mesh-like constraints hinders (keeping track of) topological change. For the latter, Heiden et al. [12] demonstrate that splitting may still be achieved by carefully designed geometrical processing.

Prediction methods [1, 9, 18, 24, 27, 34] that build upon recent advances in neural implicit representations [28, 29, 31, 44], in contrast, may exploit that the used occupancy or signed-distance fields naturally handle varying topologies. Commonly though, single objects [1, 41, 42] or whole scenes [9, 11, 35] are represented by a global implicit shape. This may require the number of objects to remain unchanged, or does not allow to easily determine if such a change indeed happened. For example, objects may be extracted in an initial step and embedded separately to discern them [9, 24]. Inspired by Cheng et al. [5], our method dynamically deals with varying connectivity – and thereby topology – during manipulation and makes these changes explicitly observable in distinct occupancy maps. No additional postprocessing of the predicted shapes is required.

3 Method

Prior works commonly focus on the *geometrical* changes that result from manipulation of deformable objects. However, few works [12,24] consider the *topological* changes that occur when deformation leads to splitting or merging of objects. We argue that a joint consideration of both, geometry and topology, is important as it allows to minimize unwanted geometrical deformation when trying to achieve a topological goal (and vice-versa).

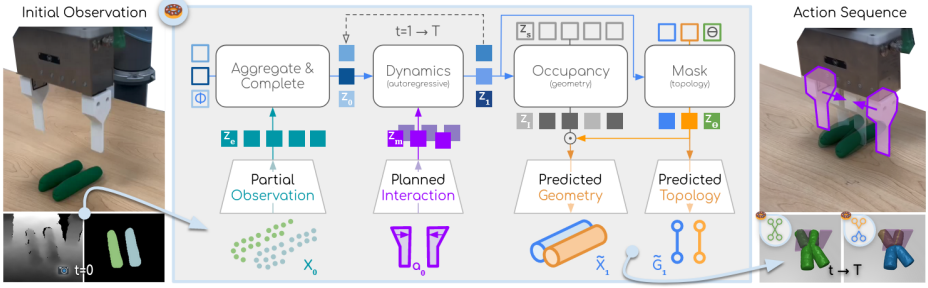

Fig. 2: DoughNet Pipeline. We encode the initial partial observation X_0 to a set of latent codes $[\boldsymbol{z}_0]$ using a learned geometry embedding Φ. The given interaction a_0 yields the next latent codes $[\boldsymbol{z}_1]$, which serve as input in subsequent time steps $t \to T$. The latent codes may be reconstructed into components using a learned topology embedding θ. This allows DoughNet to reconstruct the objects' geometry $\tilde{X}_t$ at sample locations $[\boldsymbol{z}_s]$. In addition, we may extract their topology $\tilde{G}_t$ (i.e., the number of components and their genera) from the per-component latents $[\boldsymbol{z}_\theta]$.

For this joint reasoning, DoughNet (presented in Fig. 2) is designed to perform two nested learning tasks, namely:

- **Learning a latent representation** of deformable shapes with the encoder and decoder network, Secs. 3.1 and 3.3.
- **Dynamics prediction** in the learned latent space, that infers objects' deformation and topological changes under different interactions, Sec. 3.2.

Tying these learning objectives together, we describe the losses and training procedure of our method in Sec. 3.4. While the autoencoder is supervised using a permutation-invariant reconstruction loss, the predictive model solely considers its prediction error in the latent shape space.

3.1 Shape Encoder

We hypothesize that repeated decoding, resampling and encoding would accumulate error and thus should be avoided. We therefore want to predict directly in latent space. As illustrated in Fig. 2, we achieve such a latent representation of the observed points $\boldsymbol{X} \in \mathbb{R}^{\mathrm{N}\times(3+1)}$ by a set of latent codes $[\boldsymbol{z}] \in \mathbb{R}^{257\times512}$ in two main steps, based on the architecture proposed by Zhang et al. [44].

Embed Partial Observation. To embed the observed point cloud, we apply the feature transformation from [44]. Each such feature is concatenated with its location $\boldsymbol{x}_{xyz}$, fed through a linear layer and appended with the one-hot encoded component label $l \in \mathbb{I}^P$ to create the set of per-point features $[\boldsymbol{z}_e]$.

Spatial Aggregation. A learned query set $[\boldsymbol{\phi}]$ of 256 latent codes aggregates over these per-point features via cross attention [40], $[\boldsymbol{z}_{agg}] = \mathrm{CrossAttn}([\boldsymbol{\phi}], [\boldsymbol{z}_e])$. To add further global information, inspired by [45], we add an average-pooled latent code by $[\boldsymbol{z}] = [[\boldsymbol{z}_{agg}], \overline{[\boldsymbol{z}_{agg}]}]$. The spatial aggregation to the latent set $[\boldsymbol{z}]$ allows us to work with point clouds of varying size and contains the quadratic

memory complexity of the following attention-based components by producing a latent shape set of small and fixed number. To further process the latent shape, i.e., learning to denoise and complete the partial observation, we leverage a stack of self-attention layers and their ability to consider global dependencies between the latents. While such a module is part of the decoder in [44], we observe significantly better performance when used in the encoder. Thereby, it already serves as input to the predictive model as compared to "postprocessing" its predictions.

3.2 Dynamics Model

Given a set of latent codes $[\boldsymbol{z}^t]$ in time step t, our model in Fig. 2 predicts a new set $[\boldsymbol{z}^{t+1}]$ in time step $t+1$. Its prediction is conditioned on the manipulator's interaction, i.e., its shape in both time steps. We assume this representation to be more general than an action space that depends on a specific kinematic chain.

Embed Manipulator Geometry. The manipulator's geometry is represented by a set of points uniformly sampled on its surface, combined for two time steps to describe its action $\boldsymbol{a}^t = [\boldsymbol{M}^t, \boldsymbol{M}^{t+1}]$. We conjecture that embedding the manipulator analogous to the observation facilitates finding interacting regions, e.g., for determining collisions that are implicitly required for dynamics prediction. The latent representation of the interaction is thus given by $[\boldsymbol{z}_m^t, \boldsymbol{z}_m^{t+1}]$.

Dynamics Prediction. The embedded points are concatenated, forming the context $[\boldsymbol{z}_m^{t,t+1}] \in \mathbb{R}^{257\times1024}$ for prediction based on the current object shape $[\boldsymbol{z}^t]$. We use four stacks that alternate two self- and with one cross-attention blocks to yield the next object shape $[\boldsymbol{z}^{t+1}]$. This module is autoregressively applied to predict the outcome of long-horizon manipulations, including topological change.

3.3 Geometry Decoder and Components

To complete the autoencoder, we require a shape decoder that extracts both, geometry and topology, from the latent shape $[\boldsymbol{z}]$. We jointly represent this information in the form of a set of $K+1$ occupancy masks $P(p_k|\boldsymbol{z}, \boldsymbol{s}_i)$ for samples $[\boldsymbol{s}_i] \in \mathbb{R}^{Sx3}$ and components $[p_k]_{k\in K}$. An additional mask represents samples that lie outside all shapes in the scene (*outliers*).

Interpolate Latents. To decode the latent shape information at each sample location, samples are first transformed by $\boldsymbol{z}_f(\boldsymbol{s}_i)$, concatenated with $\boldsymbol{s}_i$ itself and fed through a linear layer to yield $[\boldsymbol{z}_s]$. Note that the linear layer is different from the one used by the encoder since we have no label information for samples. Next, analogous to the spatial aggregation, we employ a cross attention layer to interpolate the latent codes $[\boldsymbol{z}]$ at the sample locations by $[\boldsymbol{z}_I] = \text{CrossAttn}([\boldsymbol{z}_s], [\boldsymbol{z}])$.

Segment Samples. As shown in Fig. 2, we combine this local decoder with a component prediction branch that is inspired by 2D segmentation [5]; instead of binary 2D masks, however, we predict a set of 3D occupancy masks. Correspondingly, we use a learned query set $[\boldsymbol{\theta}]$ of $K+1$ latent codes to compute a global component representation $[\boldsymbol{z}_{\boldsymbol{\theta}}] = \text{CrossAttn}([\boldsymbol{\theta}], [\boldsymbol{z}])$. The dot product of the globally and locally decoded shape information is interpreted as the logits of

the component occupancy probability $P(p_k|\boldsymbol{z}, \boldsymbol{s}_i)$ for component p_k at sample location $\boldsymbol{s}_i$. We prescribe each sample to belong to exactly one component (or the outliers) and extract the label that maximizes P.

Predict Topology. We integrate topology prediction as a classification task into our pipeline. In contrast to previous work [4], we simplify this task by leveraging the "pre-segmented" components. Classifying each component latent $\boldsymbol{z}^i_{\boldsymbol{\theta}}$ to either a class corresponding to its genus or an *empty* class allows us to additionally obtain the number of components using the same network head. As a practical benefit, moreover, one does not need to query the occupancy for the latent geometry of the corresponding components.

3.4 Training Procedure and Losses

Since the required ground-truth components are infeasible to obtain for real observations (i.e., determining a topological change without intervention and thus without destroying the state of the deformable object), we base our training on synthetic data described in Sec. 4. We assume that component meshes are available for each time step of the manipulation trajectories in the training set.

Generate Synthetic Observations. To generate realistic training data, we render depth images of these meshes from jittered camera poses. Sensor noise is added as a normally distributed per-pixel offset, by random pixel shifts and by removing pixels with large depth difference [8, 39]. Rendering at a lower resolution and upscaling via bilinear interpolation introduces smoothing of the depth image, creating "flying pixels" that further reduce the domain gap.

Reconstruct and Complete the Observations. The order of the K occupancy masks is ambiguous; the component label has no semantic meaning since it only encodes distinct connectivity. Therefore, similar to Cheng et al. [5], we compute the reconstruction loss using the matching $\mathcal{L}_{rec!} = \min_{K \in \mathcal{K}!} \mathcal{L}_{rec}(K, \mathcal{K}_{true})$, where $\mathcal{K}!$ is the set of all permutations of the K predicted components and $\mathcal{K}_{true}$ their true permutation. An additional component for outliers at a fixed index facilitates its rejection. We use the Focal Loss [22] to score these permutations. This considers the imbalance of occupied to empty space. The best-component permutation is reused in the cross-entropy loss for the topology classifier.

While the encoder is only given a partial and noisy observation, the targets are computed from the true shapes and labels by computing the signed distance with respect to the component meshes at the sample locations $[\boldsymbol{s}_i]$. Assuming non-intersecting meshes, a sample is assigned to the outliers if all signed distances are positive (outside all meshes) and to the component for which it is ≤ 0 else.

Predict the Next Shapes. We want our prediction model to iteratively find the shapes that follow from the given interactions in the next time step, i.e., the latent codes that would decode to the complete ground-truth shape at the respective time step. Therefore, we directly supervise the predictive model in latent space by the Huber loss [15] between the latent set of the prediction $\boldsymbol{z}_{pred}$ and the corresponding encoder output $\boldsymbol{z}$. We combine this loss for two time steps, namely $t, t+1$. In the first time step, the prediction $[\boldsymbol{z}^t_{pred}]$ is computed from the encoded shape $[\boldsymbol{z}^{t-1}]$. To reduce the exposure bias commonly observed in

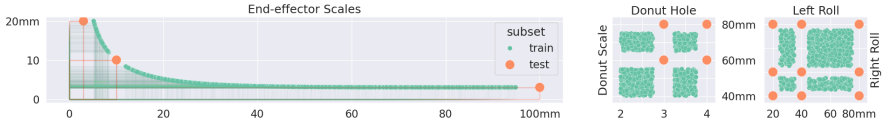

Fig. 3: Data Distribution. The scales of the objects and EE geometry, where test samples are held-out from regions inside and outside the training boundaries.

autoregressive models [3,33], we use $[\boldsymbol{z}_{pred}^{t+1}]$ computed from $[\boldsymbol{z}_{pred}^{t}]$ in the second time step. Thereby, the predictive model must learn to map latents from the encoder as well as itself to the same shape. The topology classifier is also trained on latents from both sources. We find training to be more stable if we pretrain the autoencoder using the reconstruction loss and, in a second stage, freeze it and only train the predictive model using the latent loss.

4 Topological Manipulation Environment

Synthetic data generated with an MPM-based simulator is used for training. The simulation setup is presented in Sec. 4.1. Ground-truth information is extracted as training supervision, including both geometry and topology structure via a set of topological-checking operations (Sec. 4.2). Finally, the model trained with synthetic data is directly evaluated on a real-world platform (Sec. 5.2).

4.1 Simulation Setup

The simulation environment consists of one or two objects made of soft material and an end-effector (i.e., parallel-jaw gripper) with varying finger geometry. The initially opened gripper closes gradually until reaching the minimal opening width at a varying in-plane pose. Within the process, full object particle and manipulator mesh states are recorded at constant intervals ("keyframes").

The soft material is represented using an elastoplastic continuum model simulated with MLS-MPM [13] and von Mises yield criterion. MPM-based methods naturally support arbitrary deformation and topology change. The rigid gripper fingers are expressed as time-varying signed distance fields (SDFs), derived from external meshes. Contact between soft and rigid materials is modeled through the computation of surface normals of the SDFs.

We create simulation environments that are visually simple scenes, yet undergo complex topological changes in terms of the number of components (c) and genera (g). Scenes with two "rolls of dough" side-by-side (initially $c = 2$) may merge (to $c = 1$) and split (to $c = 2, 3$ or 4). Scenes with a "doughnut" (initially $g = \{1\}$) may self-merge (to $g = \{0\}$ or $\{2\}$) and split (to any combination of $g = \{0, 1\}$), where the resulting topology is highly sensitive on the EE's geometry and pose. For each type of scene, we randomly sample 1000 training and 100 testing instances for a total of 2000 and 200 scenes. Five faulty test scenes are discarded. Each scene is subsampled to 13 frames for larger displacements.

We vary the end effector geometry from thin "scissors" to a wide "vise", starting from a random in-plane pose and closing its fingers to a minimal distance. We hold out a range of scales around the test geometries, as shown in Fig. 3. For the EE and scenes, the sampled parameter regions for training and testing are disjoint and include cases requiring interpolation and extrapolation.

4.2 Topological Check

Topological connectivity is implicit in MPM-based simulation; it operates on a set of particles and a corresponding grid, not on well-separated objects. Therefore, (self-)merging and splitting are ill-defined from particle locations alone. For example, distance-based connectivity cannot distinguish two cylinders lying next to each other (in contact) from ones that have been squeezed (merged). Instead, we define their connectivity by the counterfactual argument that "they would not stay in contact *when moved apart*, if they had not merged or had split".

To this end, we propose a physically plausible topology annotation by focusing on connectivity changes induced by the EE. We derive the object regions between the EE fingers where a merge or split may occur from the kNN graph of particles at rest and their current locations. As shown in Fig. 4, our simulation-based topology check involves adjusting particle velocities to move these object regions opposite to the end effector's closing direction (merge) or orthogonal to it (split). If the distance between these regions after perturbation by the checking action falls below (merge) or above (split) a threshold, this topological change is recorded in both the particle-connectivity graph and the scene-topology graph. The latter is the objects' initial topology in terms of a simple planar graph. Each recorded change alters this graph using merge, split, and remove graph operations. To support, e.g., complex self-merge events, our operations also take the particles surrounding the check regions into account. This scene-topology graph is easily evaluated in terms of the number of connected components and cycles, where the number of cycles is equivalent to the genus of the corresponding object.

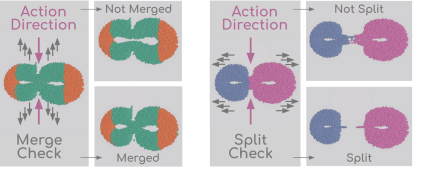

Fig. 4: Topology Check. Left: Two components are considered merged *iff* they remain connected after opposite velocities are applied. Right: A component is considered split into two *iff* they have no connection after checking.

While the particle representation lends itself to simulation, for further processing such as resampling the object geometry's surface or interior, a surface mesh representation is more flexible. Equipped with the dense labeling in the particle-connectivity graph, we can select all particles that belong to an individual component and compute its corresponding surface mesh. To this end, we compute the signed distance with respect to the each component's particles in a regular grid and apply Dual Contouring [10, 16], which is designed to maintain sharp features – such as cuts in deformable objects. Our final processed dataset minimally consists only of color-coded component meshes and manipulator meshes with poses, as well as the scene-topology graph.

5 Experiments

Going beyond predicting deformation of geometry, in this section, we show that DoughNet is able to also accurately predict the resulting changes in connectivity as components merge and split during manipulation.

Procedure and Metrics. A *single step* prediction observes the scene in every step and predicts one step into the future. For *full sequence* prediction, only a single initial observation is given and methods need to generate subsequent future states from their own output. We measure performance using the following four metrics, computed per frame and reported as mean over all frames:

- **Voxel IoU (VIoU)** compares the predicted and true scene occupancy (i.e., over all components). It thus only measures the geometrical quality. Occupancy is evaluated on a grid with the equivalent of a 1mm spacing, ignoring points below the plane or within the EE since they are assumed to be known.
- **Component IoU (CIoU)** compares the occupancy per true component. It shows both geometrical and topological quality. We report the mean over the best matching components' IoU.
- **Accuracy of the Number of Components (AccC)** is the percentage for which methods are able to predict the correct number of components.
- **Accuracy of the Genus (AccG)** is the equivalent metric for the genera of these components. For consistency, the permutation found for the CIoU metric is reused here to match the components.

Baselines. We compare DoughNet with different approaches to (1) geometry representation (particle point cloud or occupancy), (2) topology (post-hoc static or dynamic check), and (3) prediction (simulation- or learning-based):

- **VPM** (Visual Predictive Model) is a variant of our approach that discards the predicted components and instead computes the topology from scratch. These results motivate the inclusion of topology into our joint prediction.
- **MLS-MPM** [13] is a predictive model using the MPM simulator and our initial geometry reconstruction. This algorithm would provide a performance oracle for the dynamics prediction. However, the simulation could be sensitive to initialization and less robust to partial and noisy observations.
- **RoboCook** [36] is a GNN-based predictive model of object deformation. Building upon a line of previous works [20, 21, 37], it is representative of learned models that closely follow the design of particle-based simulators. Our comparison therefore shows the benefit of reasoning on an object level.

We furthermore augment these baselines by post-hoc topology checks:

- **Static topology check** extracts the connected components from the nearest-neighbor graph [30]. A mesh is computed for each, using the Euler characteristic to obtain its genus. This baseline illustrates the necessity of considering *dynamic* connectivity; beyond using only static Euclidean distances.
- **Dynamic topology check** uses the same approach used for our synthetic data generation. As this requires repeated simulation, it is combined with the MLS-MPM baseline and highlights the computational burden of both.

Table 1: Performance on Simulated Data. Results for all 194 test scenes (13 time steps each). + with a static topological check, ‡ with our proposed dynamic topological check and ours (🍩) with predicted topology. Frame rates (fps) are for geometry *and* topology prediction, without exploiting parallel simulation or batched prediction, on a system equipped with an Intel Xeon Gold 6248 and an NVIDIA Geforce RTX 2080 Ti.

	full sequence				*at final time step*				
	VIoU↑	CIoU↑	AccC↑	AccG↑	VIoU↑	CIoU↑	AccC↑	AccG↑	fps↑
DoughNet 🍩	**92.0**	**90.5**	**97.9**	98.6	**84.1**	75.1	**92.3**	90.3	**22.1**
VPM^{+}	**92.0**	88.5	92.8	94.8	83.3	70.7	68.7	79.7	0.4
MLS-MPM [13]‡	87.2	86.2	97.2	**98.9**	82.0	**79.4**	91.8	**95.6**	0.2
RoboCook [36]$^{+}$	60.8	59.2	91.5	95.6	44.1	37.4	64.1	80.9	4.5

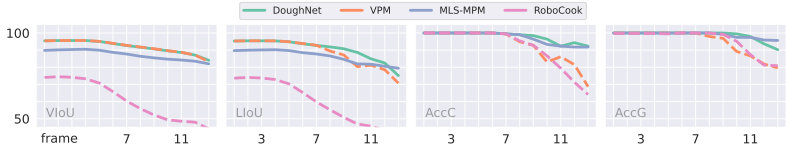

Fig. 5: Performance on Simulated Sequences. Mean performance per frame; dashed for the static topology check ($^{+}$), and solid for dynamic (‡) and predicted (🍩).

5.1 Results

As indicated in Tab. 1, DoughNet significantly outperforms the baselines in full sequence prediction, retraining high accuracy over a long prediction horizon. In Tab. 2, we compare the effect of key design decisions on prediction quality.

Comparison to Post-hoc Topology Prediction. The significant performance drop of using a static topology check for VPM illustrates that the effects of topological manipulation go beyond geometrical changes. Our approach, instead, learns to exploit the observations generated by the proposed dynamic check in simulation to jointly reason about such topological change, rather than relying on post-hoc clustering of predicted points.

Comparison to Simulation-based Prediction. As shown in Tab. 1, only in the final time step, MLS-MPM – the simulation initialized with our reconstruction and using our proposed check as topology oracle – performs better on the CIoU and AccG metrics, while still being outperformed by ours in terms of VIoU and AccC. This drop may be attributed to DoughNet at times requiring additional steps to cleanly resolve a topological change in the per-component occupancy. Still, we achieve better or comparable results but magnitudes faster.

Comparison to GNN-based Prediction. While DoughNet maintains high performance over subsequent predictions, RoboCook's performance deteriorates over time in Fig. 5. It is further diminished by the downsampling to a sparse point cloud as input for the GNN, necessitating costly meshing for upsampling to a dense (yet still lossy) prediction again in each frame. DoughNet achieves a 31.3% higher CIoU than RoboCook. This comparison supports our choice of an implicit representation that may be evaluated at any resolution, jointly handling geometrical and topological change without loss in quality.

Table 2: Ablation Study on Design Decisions. In each row, we replace one design decision with an alternative described in Sec. 5.1. We note the significant challenge of long-horizon prediction of topological manipulation, as compared to single steps.

	single step				*full sequence*			
	VIoU$^\uparrow$	CIoU$^\uparrow$	AccC$^\uparrow$	AccG$^\uparrow$	VIoU$^\uparrow$	CIoU$^\uparrow$	AccC$^\uparrow$	AccG$^\uparrow$
DoughNet	94.3	93.0	98.2	98.8	**92.0**	**90.5**	**97.9**	**98.6**
one-step prediction	94.1	92.7	98.1	98.8	90.8	89.0	97.4	98.4
explicit supervision	**94.8**	**93.7**	98.0	98.7	76.3	75.4	91.7	93.1
relative prediction	94.4	93.2	**98.4**	**99.0**	73.1	74.2	80.4	81.6
complete observation	77.0	75.3	95.5	98.1	75.8	73.4	95.7	96.9

Effect of Training Multi-step Prediction. Prior works [3,33] observe that autoregressive models, as ours, suffer from an *exposure bias*; training on ground-truth inputs but using own predictions as input in subsequent steps at test time. Shi et al. [36] are able to reduce this effect by predicting two time steps during training. Following this observation, we predict one step from reconstruction of the true state and one step from the prediction of the next step. As compared to only training the first time step, there is a slight improved performance in the *single step* setting that increases to a bit over 1% on the IoU metrics when moving to *full sequences*, as shown in Tab. 2.

Effect of Latent Space Supervision. In RoboCook [36], prediction is supervised by the reconstruction's distances to the ground truth – a small error in latent space is not penalized during training as long as the reconstruction is similar. We hypothesize that this is an additional source for error accumulation in sequential prediction. Indeed, we find that directly supervising the prediction of the latents is more robust as this forces the predictive model to map to the space learned by the autoencoder, further avoiding degenerate reconstructions. Quantiatively, we observe a significantly improved long-horizon performance using our approach, achieving about 15% higher IoU scores in Tab. 2.

Effect of Set-to-set Prediction An approach in previous visual predictive models, perhaps inspired by a Lagragian view on dynamics, is to learn the relative change between states. Like any iterative procedure without error correction, we conjecture that this approach is prone to error accumulation. Rather, we directly predict the next state – from one set of latents to the next. The results in Tab. 2 quantify the benefit of this, specifically a 18.9% VIoU increase and 17.5% AccC increase compared to the *relative* variant. Further supporting our decision, we observe that our predictive model is able to correct, e.g., initial reconstructions in terms of topological errors and geometrical inaccuracies.

Effect of Training on Partial Observations. Rather than assuming a complete initial state (as for MLS-MPM [13]) or a multi-view reconstruction (as in [36]), we conjecture that a single partial observation is sufficient. To bridge the domain gap between synthetic and real point-cloud observations, we introduce common artefacts of depth sensing, namely incompleteness, oversmoothing, and fringing at transitions to the background. Experiments on synthetic partial data

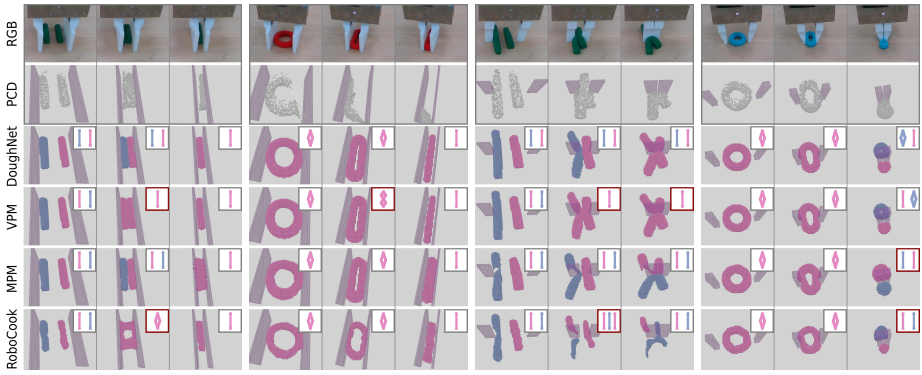

Fig. 6: Qualitative Comparison on Real Sequences. Full-sequence predictions from an initial partial observation. Results are shown at time steps $t = \{5, 10, 15\}$. Erroneous topology predictions (white boxes, top right) feature a dark red border.

in Tab. 2 show an expected yet significant improvement using our data preparation pipeline as compared to training on complete observations. Importantly, as shown in Fig. 6, this robustness to partial input carries over to experiments on the robot and enables successful sim-to-real transfer. We want to highlight that, predicting from a single partial observation, we even achieve plausible reconstructions of components that are eventually hidden from direct observation.

5.2 Real-world Evaluation

We further evaluate our model on a real-world platform as shown in Fig. 7. The scene setup and placement strategy follows the synthetic dataset. The setup is observed using a single camera, which provides the initial partial and noise-afflicted observation for our model. A total of 60 scenes (*interactions*) of 15 frames each are recorded. In a preprocessing step, we segment the objects of interest by color and crop the resulting point cloud by the known table plane. Statistical outlier removal and voxel downsampling reduce the gap between synthetic and real data. Furthermore, for two-roll scenes, PCA-based reorientation and centering along the median axis between the objects bring the observation to a (noise-afflicted) canonical frame, in which we can easily determine the initial components. The qualitative results are shown in Fig. 6, demonstrating that our model trained on purely synthetic data is able to generate plausible deformation and topology changes from single RGB-D images in real robotics experiments.

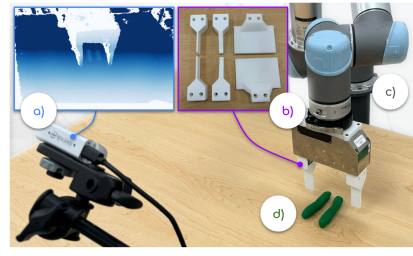

Fig. 7: Real-world Setup. a) Intel RealSense D415 (and depth image), b) Weiss Robotics WSG50 (with three held-out 3D-printed tools), attached to c) a Universal Robots UR5 to manipulate d) molded Play-Doh objects.

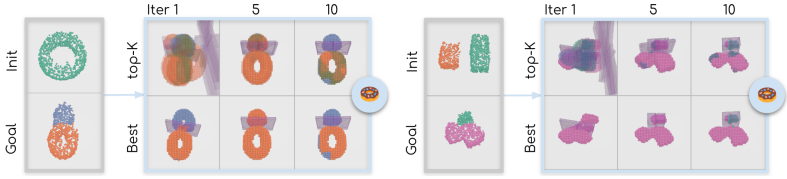

Fig. 8: Planning Topological Manipulation. Two examples, each defined by an initial partial observation and a partial goal observation (color-coded topology). We visualize DoughNet's predictions for the top-K sampled actions (top) and the best one selected based on our latent representation (bottom) after 1, 5, and 10 CEM iterations.

5.3 Planning for Topological Manipulation

We leverage DoughNet to plan topological manipulations. Goals may be provided as a (partial) point cloud with color-coded components (Fig. 8), or a desired topology (see the supplementary). Specifically, DoughNet is used to roll-out sampled actions in a CEM planner [7]. We use the cosine similarity between the goal and our predictions in the learned latent space, naturally capturing geometrical and topological fitness for resampling and selection. We initialize $K = 64$ actions; in-plane translation and rotation from Gaussian distributions $t_x, t_y \sim \mathcal{N}_t(\mathbf{0}, \boldsymbol{I} \cdot 8) \in [-40, 40]^2$mm and $\theta_z \sim \mathcal{N}_\theta(0, 10) \in [-10, 10]$deg, as well as the gripper from a multinomial distribution $x \sim \mathcal{M}_x(\text{narrow} = \frac{1}{3}, \text{regular} = \frac{1}{3}, \text{wide} = \frac{1}{3})$. After each rollout, the sampled actions are scored by similarity, the best $K_{\text{top}} = 8$ are kept and used to re-fit the distributions. The best action is selected after ten CEM iterations. The results in Fig. 8 visualize the distribution of the top-K and the best samples over multiple CEM iterations, as predicted and reconstructed by DoughNet. The large initialization variance of in-plane actions and EE geometries quickly reduces and we are able to fulfill both goal aspects, also finding a well-suited EE. Quantitatively, the action parameters generating the goal observations in Fig. 8 are accurately estimated by our best plans, achieving translation and rotation errors of 1.6 / 1.1mm and 0.4 / 0.5deg, respectively; close to the grid resolution of 1mm used for the reconstructions during evaluation. A comparison to baselines is provided in the supplementary.

5.4 Limitation and Failure Case

As shown in Fig. 9, we observe two common failure cases in DoughNet's predictions. First, since this is not enforced, the predicted component occupancy and topology may be inconsistent. For example, in Fig. 9 (left), two geometrical components but three topological ones are predicted. This is related to an observed tendency to delayed, or even missed, splitting events.

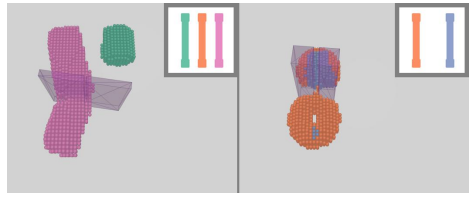

Fig. 9: Failure Cases. Inconsistency between occupancy and topology (left) and within components (right).

Second, topological changes may result in inconsistency within the occupancy prediction. In the example in Fig. 9 (right), the new components are still partially assigned to one another. However, we observe that this mistake is typically resolved in subsequent steps, presumably as such erroneous shapes (i.e., their latents) are mapped back to the learned manifold of shapes.

Stemming from our dataset design, targeted at providing a minimal test bed for topological manipulation, we only consider individual squeezing actions and a single set of material properties. While these added complexities are beyond the scope of this work, we hope that the release of our data generator will facilitate future work on topological manipulation. Future tasks may involve repeated topological and geometrical changes; such as splitting material into multiple components, deforming each with a sequence of actions, and merging them into an intricate shape that would not be attainable by deformation alone.

Finally, we do not anticipate any negative societal impact of this work, if responsibly incorporated in robotic systems such that potentially unreliable or erroneous predictions may be considered via verification before open-loop execution, or via reinitialization of short prediction horizons in closed-loop control.

6 Conclusion

We propose **DoughNet**, a visual predictive model that jointly reasons about geometrical deformation and resulting topological changes. In our experiments using simulated and real robotics interactions, our approach accurately predicts splitting and merging of deformable objects, manipulated by varying end-effector shapes. In addition, we propose a synthetic data generator to facilitate further research into this task of *topological manipulation*, where a geometrical or topological goal may be constraint by the respective other.

The successful sim-to-real transfer of our experimental setup should motivate future work extending it towards more complex, sequential topological manipulation tasks. As a predictive model, DoughNet lends itself to be employed in robotic planning pipelines; its general notion of interactions via a tuple of "manipulator" geometries moreover extends its application possibilities beyond robotics, while its speed and differentiability opens up integration into existing end-to-end trained or optimization-based approaches.

Acknowledgements

This work was supported in part by the Toyota Research Institute, NSF Award #2143601, and Microsoft Fellowship. We would like to thank Google for the UR5 robot hardware. Dominik Bauer is partially supported by the Austrian Science Fund (FWF) under project No. J 4683. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

1. Bartsch, A., Avra, C., Farimani, A.B.: Sculptbot: Pre-trained models for 3d deformable object manipulation. arXiv preprint arXiv:2309.08728 (2023)
2. Bathe, K.J.: Finite element procedures. Klaus-Jurgen Bathe (2006)
3. Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N.: Scheduled sampling for sequence prediction with recurrent neural networks. NIPS **28** (2015)
4. Chen, Q., Nguyen, V., Han, F., Kiveris, R., Tu, Z.: Topology-aware single-image 3d shape reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 270–271 (2020)
5. Cheng, B., Schwing, A., Kirillov, A.: Per-pixel classification is not all you need for semantic segmentation. NeurIPS **34**, 17864–17875 (2021)
6. Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning (2016)
7. De Boer, P.T., Kroese, D.P., Mannor, S., Rubinstein, R.Y.: A tutorial on the cross-entropy method. Annals of operations research **134**, 19–67 (2005)
8. Denninger, M., Winkelbauer, D., Sundermeyer, M., Boerdijk, W., Knauer, M., Strobl, K.H., Humt, M., Triebel, R.: Blenderproc2: A procedural pipeline for photorealistic rendering. Journal of Open Source Software **8**(82), 4901 (2023). https://doi.org/10.21105/joss.04901
9. Driess, D., Huang, Z., Li, Y., Tedrake, R., Toussaint, M.: Learning multi-object dynamics with compositional neural radiance fields. pp. 1755–1768 (2023)
10. Gibson, S.F.F.: Constrained elastic surfacenets: Generating smooth models from binary segmented data. TR99 **24** (1999)
11. Hafner, D., Lillicrap, T., Ba, J., Norouzi, M.: Dream to control: Learning behaviors by latent imagination. In: ICLR (2019)
12. Heiden, E., Macklin, M., Narang, Y.S., Fox, D., Garg, A., Ramos, F.: DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting. In: RSS. Virtual (July 2021). https://doi.org/10.15607/RSS.2021.XVII.067
13. Hu, Y., Fang, Y., Ge, Z., Qu, Z., Zhu, Y., Pradhana, A., Jiang, C.: A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. ACM Transactions on Graphics (TOG) **37**(4), 150 (2018)
14. Hu, Y., Li, T.M., Anderson, L., Ragan-Kelley, J., Durand, F.: Taichi: a language for high-performance computation on spatially sparse data structures. ACM TOG **38**(6), 201 (2019)
15. Huber, P.J.: Robust estimation of a location parameter. The Annals of Mathematical Statistics **35**(1), 73–101 (1964)
16. Ju, T., Losasso, F., Schaefer, S., Warren, J.: Dual contouring of hermite data. In: Proceedings of the 29th annual conference on Computer graphics and interactive techniques. pp. 339–346 (2002)
17. Le Cleac'h, S., Yu, H.X., Guo, M., Howell, T., Gao, R., Wu, J., Manchester, Z., Schwager, M.: Differentiable physics simulation of dynamics-augmented neural objects. RAL **8**(5), 2780–2787 (2023)
18. Li, S., Huang, Z., Chen, T., Du, T., Su, H., Tenenbaum, J.B., Gan, C.: Dexdeform: Dexterous deformable object manipulation with human demonstrations and differentiable physics. arXiv preprint arXiv:2304.03223 (2023)
19. Li, X., Qiao, Y.L., Chen, P.Y., Jatavallabhula, K.M., Lin, M., Jiang, C., Gan, C.: Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. In: ICLR (2022)

20. Li, Y., Lin, T., Yi, K., Bear, D., Yamins, D.L., Wu, J., Tenenbaum, J.B., Torralba, A.: Visual grounding of learned physical models. In: ICML (2020)
21. Li, Y., Wu, J., Tedrake, R., Tenenbaum, J.B., Torralba, A.: Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In: ICLR (2019)
22. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV. pp. 2980–2988 (2017)
23. Lin, X., Huang, Z., Li, Y., Tenenbaum, J.B., Held, D., Gan, C.: Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. In: ICLR (2022)
24. Lin, X., Qi, C., Zhang, Y., Huang, Z., Fragkiadaki, K., Li, Y., Gan, C., Held, D.: Planning with spatial-temporal abstraction from point clouds for deformable object manipulation. In: CoRL (2022)
25. Macklin, M., Müller, M., Chentanez, N., Kim, T.Y.: Unified particle physics for real-time applications. ACM Transactions on Graphics (TOG) **33**(4), 1–12 (2014)
26. Matl, C., Bajcsy, R.: Deformable elasto-plastic object shaping using an elastic hand and model-based reinforcement learning. In: IROS. pp. 3955–3962 (2021)
27. Van der Merwe, M., Berenson, D., Fazeli, N.: Learning the dynamics of compliant tool-environment interaction for visuo-tactile contact servoing. In: CoRL. pp. 2052–2061 (2023)
28. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
29. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR. pp. 165–174 (2019)
30. Pearce, D.J.: An improved algorithm for finding the strongly connected components of a directed graph. Victoria University, Wellington, NZ, Tech. Rep (2005)
31. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: ECCV. pp. 523–540. Springer (2020)
32. Qi, C., Lin, X., Held, D.: Learning closed-loop dough manipulation using a differentiable reset module. RAL **7**(4), 9857–9864 (2022)
33. Schmidt, F.: Generalization in generation: A closer look at exposure bias. arXiv preprint arXiv:1910.00292 (2019)
34. Seo, Y., Hafner, D., Liu, H., Liu, F., James, S., Lee, K., Abbeel, P.: Masked world models for visual control. In: CoRL. pp. 1332–1344 (2023)
35. Shen, B., Jiang, Z., Choy, C., Savarese, S., Guibas, L.J., Anandkumar, A., Zhu, Y.: Acid: Action-conditional implicit visual dynamics for deformable object manipulation. In: RSS (2022)
36. Shi, H., Xu, H., Clarke, S., Li, Y., Wu, J.: Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. arXiv preprint arXiv:2306.14447 (2023)
37. Shi, H., Xu, H., Huang, Z., Li, Y., Wu, J.: Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks. In: RSS (2022)
38. Sulsky, D., Chen, Z., Schreyer, H.L.: A particle method for history-dependent materials. Computer methods in applied mechanics and engineering **118**(1-2), 179–196 (1994)
39. Tölgyessy, M., Dekan, M., Chovanec, L., Hubinskỳ, P.: Evaluation of the azure kinect and its comparison to kinect v1 and kinect v2. Sensors **21**(2), 413 (2021)
40. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS **30** (2017)

41. Wi, Y., Florence, P., Zeng, A., Fazeli, N.: Virdo: Visio-tactile implicit representations of deformable objects. In: ICRA. pp. 3583–3590 (2022)
42. Wi, Y., Zeng, A., Florence, P., Fazeli, N.: Virdo++: Real-world, visuo-tactile dynamics and perception of deformable objects. In: CoRL (2022)
43. You, Y., Shen, B., Deng, C., Geng, H., Wang, H., Guibas, L.: Make a donut: Language-guided hierarchical emd-space planning for zero-shot deformable object manipulation. arXiv preprint arXiv:2311.02787 (2023)
44. Zhang, B., Tang, J., Nießner, M., Wonka, P.: 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. ACM Trans. Graph. **42**(4) (jul 2023). https://doi.org/10.1145/3592442
45. Zhao, Z., Liu, W., Chen, X., Zeng, X., Wang, R., Cheng, P., Fu, B., Chen, T., Yu, G., Gao, S.: Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. arXiv preprint arXiv:2306.17115 (2023)