PokeFlex: A Real-World Dataset of Deformable Objects for Robotics

Jan Obrist[*,1], **Miguel Zamora**[*,1] , **Hehui Zheng**[*,2,3], **Ronan Hinchet**[2], **Firat Ozdemir**[5], **Juan Zarate**[4], **Robert K. Katzschmann**[2,3], **Stelian Coros**[1]
[1] Computational Robotics Lab, ETH Zurich.
[2] Soft Robotics Lab, ETH Zurich.
[3] ETH AI Center. ETH Zurich.
[4] Advanced Interactive Technologies Lab, ETH Zurich.
[5] Swiss Data Science Center, ETH Zurich and EPFL.

Abstract: We propose PokeFlex, a dataset featuring real-world paired and annotated multimodal data that includes 3D textured meshes, point clouds, RGB images, and depth maps. To deal with the challenges posed by real-world 3D mesh reconstruction, we leverage a professional volumetric capture system that allows complete 360° reconstruction. PokeFlex consists of 18 deformable objects with varying stiffness and shapes. Deformations are generated by dropping objects onto a flat surface or by poking the objects with a robot arm. Interaction forces are also reported for the latter case. Using different data modalities, we demonstrated a use case for the PokeFlex dataset in online 3D mesh reconstruction. We refer the reader to our website[2] for demos and examples of our dataset.

Keywords: Deformable objects, Robotics, 3D mesh reconstruction

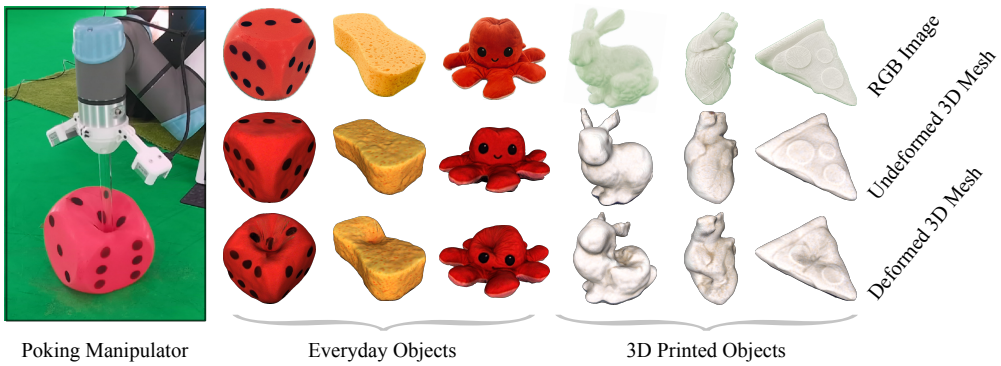

Figure 1: PokeFlex captures the deformability of various everyday and 3D-printed objects, as illustrated by the poking manipulator on the **Left**. On the **Right**, the **Top Row** contains segmented RGB images of selected objects. The **Middle Row** shows reconstructed objects in an undeformed state. The **Bottom Row** provides reconstructed 3D-textured meshes of deformed objects.

1 Introduction

The development of high-quality datasets is essential to further advance research in deformable object manipulation. Such datasets are crucial for training manipulation policies, estimating material parameters, and training 3D mesh reconstruction models. In light of these needs, this work aims to create a reproducible, diverse, and high-quality dataset for deformable volumetric objects that is grounded in real-world data and incorporates multiple sensor modalities.

[*]Equal contribution. Correspondence to miguel.zamora@inf.ethz.ch
[2] https://pokeflex-dataset.github.io/

Workshop on Mastering Robot Manipulation in a World of Abundant Data (CoRL 2024), Munich, Germany.

Table 1: Dataset overview (per object, per sequence).

Sequence Data	**Poking**	**Dropping**
3D textured deformed mesh model	✓	✓
RGB images from two Volucam cameras (cameras from the MVS)	✓	✓
RGB-D images from two Realsense D405 sensors (eye-in-hand mounted)	✓	
RGB-D images from two Azure Kinect sensors (eye-to-hand mounted)	✓	
Estimated 3D contact forces and torques	✓	
End-effector poses	✓	
Camera and Object Data		
Camera intrinsic and extrinsic parameters		✓
3D textured template mesh model		✓
Open-source print files to reproduce the 3D printed objects		✓

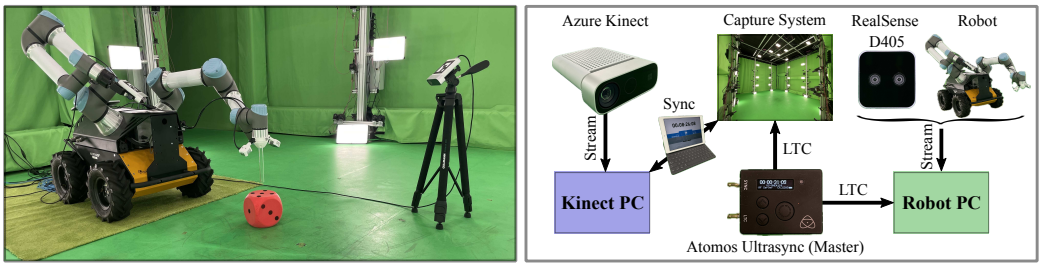

Figure 2: **Left**: Robotic manipulator positioned inside MVS with external lower-cost camera sensors during a poking sequence. **Right**: Overview of the system architecture to capture PokeFlex data.

Current state-of-the-art simulators are an attractive alternative to collect such datasets as they provide easy access to privileged information such as deformed mesh configurations and contact forces [1, 2, 3, 4, 5]. However, such simulators require careful system identification and fine-tuning to address the sim-to-real gap, which ultimately requires real-world data. Custom multi-camera systems [6] can be used to collect real-world 3D models, however, they require careful synchronization and data curation, especially when using noisy lower-cost sensors.

To address these challenges, we leverage a professional multi-view volumetric capture system (MVS) that allows capturing detailed 360° mesh reconstructions of deformable objects over time [7]. We integrate a robotic manipulator with joint-torque sensing capabilities into the MVS, enabling contact force estimation and facilitating automated data collection. Moreover, to enhance reproducibility and to expand the diversity of data modalities, we also integrate and synchronize lower-cost Azure Kinect and Intel Realsense D405 RGB-D sensors into the MVS.

An overview of the paired, synchronized, and annotated data is presented in Table 1. We demonstrated a use case of the PokeFlex dataset, proposing baseline models capable of ingesting PokeFlex multimodal data. We present evaluation criteria for benchmarking the results. The proposed architectures are suitable for online applications, reconstructing 3D meshes at a range from 106 Hz to 215 Hz depending on the input data modality, on a desktop PC with an NVIDIA RTX 4090 GPU. A summary of related work on deformable object datasets can be found in Appendix A.

2 Methodology

Data Acquisition. The PokeFlex dataset involves the acquisition of deformations under two different protocols (i) poking and (ii) dropping. For the poking protocol, we use a robotic manipulator that pokes objects with a transparent acrylic stick multiple times along a randomly oriented horizontal vector (Figure 2). For the dropping protocol, objects are attached to a light nylon cord at approximately 2 m height and captured in a free-fall drop onto a flat surface. We record data at 30 fps and 60 fps for the poking and dropping protocols, respectively. The capture system – MVS – consists of 53 RGB and 53 infrared cameras with 12 MP resolution. For the poking protocol, we integrated

and synchronized additional hardware to the MVS to ensure temporally aligned data capture across all modalities. The additional hardware includes the robot manipulator and four additional RGB-D cameras: two eye-to-hand Azure Kinect cameras, and two eye-in-hand Intel RealSense D405 cameras. The robot logs end-effector poses, interaction forces and torques at 120 Hz, while these four cameras record RGB-D data at 30 Hz. Further details about device synchronization, using a leader/follower architecture based on a Linear Timecode (LTC) signal, can be found in Appendix B.

We utilize a system similar to that described by [7] to reconstruct the meshes and textures of the entire scene. When recording at 30 fps, the MVS generates raw data at approximately 27 GB/s, which is curated by the authors to ensure that only the deformable objects are retained in the scene.

Learning-based Mesh Reconstruction. We train models for template-based mesh reconstruction, predicting the deformation of the rest-state mesh of an object using various combinations of input data modalities: sequences of images, point clouds (PCDs), and/or robot data. Figure 3 illustrates the building blocks that we used to generate different architectures depending on the input modalities.

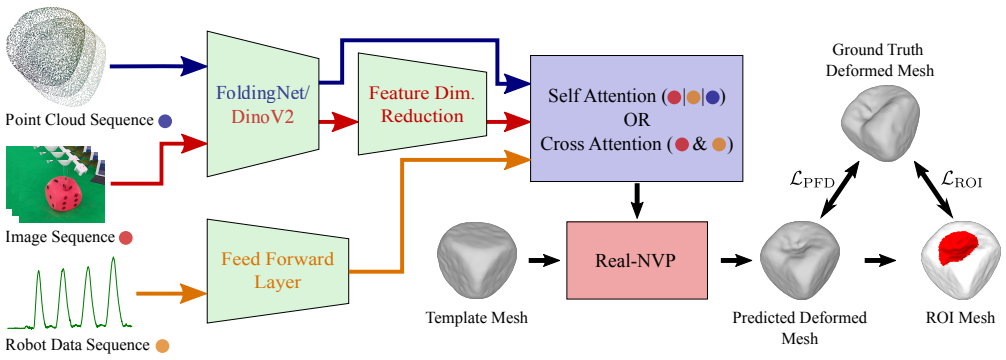

Figure 3: Superimposed representation of the proposed network architectures for ingesting the multi-modal PokeFlex data to predict deformed mesh reconstruction.

At a high level, we use three main common components: an encoder for extracting features from an input modality, an attention mechanism for exploiting temporal information, and a conditional-NVP [8] for predicting the offsets of template vertices, yielding the predicted deformed mesh.

Image input: For pipelines using images as input, we use a DinoV2 vision transformer to extract embeddings of each image frame. In particular, we use a DinoV2-small model, pretrained via distillation from the largest DinoV2 transformer presented in Oquab et al. [9] (LVD-142M dataset). The embedding dimension is later reduced using a 1D convolutional layer and a subsequent fully connected layer (Feature Dim. Reduction block in Figure 3).

Point cloud input: When using PCDs, we leverage a FoldingNet encoder [10] for representation learning, trained end-to-end together with the attention mechanism and the conditional-NVP.

Robot data input: To fuse the robot data, we concatenate the measured end-effector forces and the position of the interaction point. The concatenated data is later fed into a single fully connected layer, to match the dimensionality of the embeddings used for the attention mechanisms.

For set-ups having single data modality as input, we use self-attention mechanism. In contrast, we employ cross-attention across modalities when multiple modalities are available as observation (see Figure 3). In the experiments presented in the results section, we use cross-attention to handle a mixture of image sequences and robot data sequences as input. However, other combinations of input data are also possible. Loss function and metrics are detailed in Appendix C.

3 Results

Dataset: The PokeFlex dataset comprises 18 deformable objects, including 13 everyday items as well as 5 objects that are 3D printed. In order to ensure the reproducibility of the 3D printed ob-

jects, we provide detailed specifications in Appendix D. For the poking protocol, we recorded 4-8 sequences of 5-6 seconds at 30 fps for each object. Similarly, for the dropping protocol, we recorded 3 sequences of 1 second at 60 fps for each object. In the case of the poking sequences, each frame includes synchronized and paired data from all modalities.

The total number of reconstructed frames used to generate ground-truth data was 20k, which comprises 16.8k frames for the poking sequences and 3.2k frames for the dropping sequences. Considering the different modalities, the total of PokeFlex amounts to more than 240k samples. It is worth noting that after curating the frames of the poking sequences, i.e., discarding the frames where the robot arm is not in contact with the objects, the total number of active paired poking frames sum up to 8.4k. A summary of the physical properties of the objects, as well as a per-object list of the recorded frames under deformation for the poking sequences, is presented in Appendix E. For the dropping protocol, we recorded 180 frames per object.

Overview of training data. In the following experiments, we exclusively used poking sequences because of the higher diversity of input data modalities available. The input sequence length was set to 5 steps, heuristically, for better performance. The train-validation split was generated by randomly choosing one recording sequence per object as the validation set.

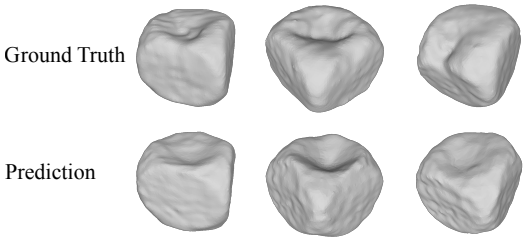

Figure 4: Three example predictions of deformations of a foam dice and their corresponding losses and the CD_{UL1} metric.

Learning from different data modalities. In this experiment, we train various deformed mesh prediction models using sequences from different input modalities. For each combination of the tested input modalities, a single model is trained on a subset of PokeFlex containing five objects simultaneously. Detailed performance results for the evaluated data modalities are in Table 2. Inference rates across different data modalities, detailed in Appendix G, range from 106 Hz to 215 Hz for dense PCDs and forces, respectively. Figure 4 shows examples of predicted meshes with varying reconstruction quality obtained using a multi-object model trained from image-sequences. Additional experiments and a comprehensive discussion of the results are presented in Appendix H and Appendix I, respectively.

Table 2: Prediction performance metrics for proposed model configurations.

Input	$\mathcal{L}_{\textbf{PFD}} \cdot 10^3 \downarrow$	$\mathcal{L}_{\textbf{ROI}} \cdot 10^3 \downarrow$	**RPFD** $\downarrow$	$\textbf{CD}_{\textbf{UL1}}$**[mm]** $\downarrow$	$J(\boldsymbol{M}_{\textbf{P}}, \boldsymbol{M}_{\textbf{GT}}) \uparrow$
Images	3.90	4.27	0.649	5.656	0.847
Robot Data	4.82	2.81	0.747	6.266	0.830
Images + Robot data	3.08	3.34	0.548	5.291	0.860
Kinect PCDs	2.83	3.73	0.589	**5.050**	**0.870**
Dense synthetic PCDs (5k)	**2.58**	**2.81**	**0.450**	5.150	0.869
Sparse synthetic PCDs (100)	3.72	5.30	0.640	5.513	0.851

4 Conclusion

This paper introduced PokeFlex, a new dataset that captures the behavior of 18 deformable volumetric objects throughout poking and dropping. The focus is on volumetric objects, while thin clothing items or thin cables are not considered in the dataset. In an effort to enhance reproducibility, the objects included in our dataset can be either purchased from global providers or 3D printed with our open-source models. Using different combinations of the data modalities provided in PokeFlex, we train and benchmark a list of baseline models for the task of multi-object template-based mesh reconstruction. In doing so, we present a list of suitable criteria for evaluating PokeFlex.

With its rich and multimodal data, as well as its focus on reproducibility, we believe that PokeFlex will drive innovation in both simulation-based and real-world applications of deformable object manipulation. This includes better material parameter identification to fine-tune simulators, viewpoint-agnostic online 3D mesh reconstruction methods, and policy learning for manipulation tasks. We look forward to sharing this dataset with the community and fostering new collaborations.

Acknowledgments

This article is supported by the SDSC Grant entitled 'C22-08: Data-Driven Inference of Mesh-based Representations for Deformable Objects from Unstructured Point Clouds'

References

[1] 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*, 7(3):6274–6281, 2022. doi:10.1109/LRA.2022.3158725.

[2] M. Macklin. Warp: A high-performance python framework for gpu simulation and graphics. https://github.com/nvidia/warp, March 2022. NVIDIA GPU Technology Conference (GTC).

[3] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin. Differentiable simulation of soft multi-body systems. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[4] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi:10.1109/IROS.2012.6386109.

[5] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, and S. Cotin. SOFA: A Multi-Model Framework for Interactive Physical Simulation. In Y. Payan, editor, *Soft Tissue Biomechanical Modeling for Computer Assisted Surgery*, volume 11 of *Studies in Mechanobiology, Tissue Engineering and Biomaterials*, pages 283–321. Springer, June 2012. doi:10.1007/8415_2012_125.

[6] H.-y. Chen, E. Tretschk, T. Stuyck, P. Kadlecek, L. Kavan, E. Vouga, and C. Lassner. Virtual elastic objects, 2022.

[7] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan. High-quality streamable free-viewpoint video. *ACM Trans. Graph.*, 34(4), July 2015. ISSN 0730-0301. doi:10.1145/2766945. URL https://doi.org/10.1145/2766945.

[8] E. A. Mansour, H. Zheng, and R. K. Katzschmann. Fast point cloud to mesh reconstruction for deformable object tracking, 2024.

[9] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[10] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215, 2018.

[11] W. Xie, Z. Yu, Z. Zhao, B. Zuo, and Y. Wang. Hmdo : Markerless multi-view hand manipulation capture with deformable objects. *Graphical Models*, 127:101178, 2023. ISSN 1524-0703. doi:https://doi.org/10.1016/j.gmod.2023.101178.

[12] X. Li, Y. Guo, Y. Tu, Y. Ji, Y. Liu, J. Ye, and C. Zheng. Textureless deformable object tracking with invisible markers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[13] I. Garcia-Camacho, J. Borràs, B. Calli, A. Norton, and G. Alenyà. Household cloth object set: Fostering benchmarking in deformable object manipulation. *IEEE Robotics and Automation Letters*, 7(3):5866–5873, 2022. doi:10.1109/LRA.2022.3158428.

[14] J. Lei and K. Daniilidis. Cadex: Learning canonical deformation coordinate space for dynamic surface representation via neural homeomorphism. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6624–6634, June 2022.

[15] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[16] N. Wang, Y. Zhang, Z. Li, Y. Fu, H. Yu, W. Liu, X. Xue, and Y.-G. Jiang. Pixel2mesh: 3d mesh model generation via image guided deformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(10):3600–3613, 2021. doi:10.1109/TPAMI.2020.2984232.

[17] D. Jack, J. K. Pontes, S. Sridharan, C. Fookes, S. Shirazi, F. Maire, and A. Eriksson. Learning free-form deformations for 3d object reconstruction, 2018.

[18] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections, 2018.

[19] J. Xu, W. Cheng, Y. Gao, X. Wang, S. Gao, and Y. Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models, 2024.

[20] G. Turk. The stanford bunny. Technical report, The Stanford Graphics Laboratory, 1994.

[21] N. Noor, A. Shapira, R. Edri, I. Gal, L. Wertheim, and T. Dvir. 3d printing of personalized thick and perfusable cardiac patches and hearts. *Advanced science*, 6(11):1900344, 2019.

[22] D. Ispolatov. Stylized pizza slice 3d model. `https://stock.adobe.com/fr/3d-assets/stylized-pizza-slice/373507126`, 2024. Accessed: 2024-09-25.

[23] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*, pages 303–312, 1996.

[24] P. Sundaresan, R. Antonova, and J. Bohgl. Diffcloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10828–10835, 2022. doi:10.1109/IROS47612.2022.9981101.

[25] E. Heiden, M. Macklin, Y. S. Narang, D. Fox, A. Garg, and F. Ramos. DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi:10.15607/RSS.2021.XVII.067.

A Related Work

Table 3: Feature comparison of the PokeFlex dataset with other deformable object datasets.

	Real-world	Meshes	Point clouds	RGB images	Force torque	# of objects	# of time frames	Type of deformation
PokeFlex (ours)	✓	✓	✓	✓	✓	18	20k	Poke, drop
HMDO [11]	✓	✓		✓		12	2,166	Manual†
PLUSH [6]	✓		✓	✓	Force	12	22.84k	Airstream
DOT [12]	✓		✓	✓		4	117k	Manual
Household Cloth Object Set [13]	✓	✓‡		✓		27	67	/
Defgraspsim [1]		✓				34	1.1M	Grasp

† by hand ‡ for ten static scenes of the cloth objects folded

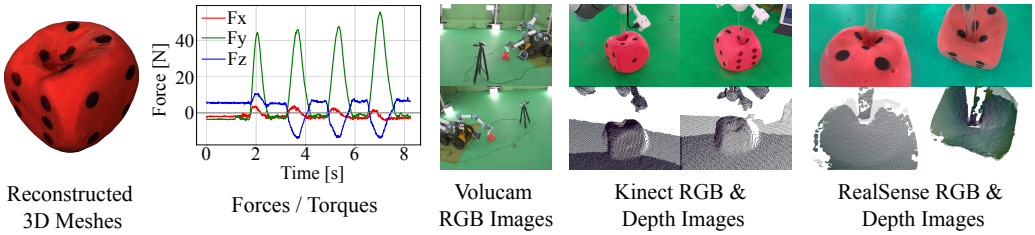

Figure 5: Samples of different data modalities provided by the PokeFlex dataset.

Deformable object datasets. Depending on the use of synthetic or real-world data, deformable object datasets can be roughly categorized into two major groups. Huang et al. [1], for instance, evaluates multiple grasping poses for deformable objects on a large-scale synthetic dataset. Qualitative sim-to-real experiments for such dataset, show that their simulator captures the general deformation behavior of objects during grasping. However, careful system identification and parameter tuning are necessary to achieve higher sim-to-real fidelity.

On the other hand, real-world data collection opens up the door to better capture the complex behavior of deformable objects. Current real-world datasets focus mostly on RGB images. HMDO [11] also provides real-world 3D meshes for objects undergoing deformation due to hand manipulation. However, they fell short of providing point cloud or force contact information. Chen et al. [6] provides points clouds and force contact information but it does not perform 3D mesh reconstruction and the deformations are only globally produced using an airstream. Li et al. [12] offer a large number of frames, however, the object diversity in their dataset is limited.

In a departure from other datasets, PokeFlex offers a more comprehensive list of features including; 3D meshes, point clouds, contact forces, higher diversity of objects, and multiple types of deformations as detailed in Table 3 and illustrated in Figure 5. For simplicity, we report only the effective number of paired time frames in our table, in contrast to what is reported by Xie et al. [11] and Li et al. [12], where the total number of samples is computed as the number of time frames times the number of cameras.

Data-driven mesh reconstruction methods vary widely in terms of the input data modalities they employ. Previous approaches that rely on point clouds to predict deformations are typically trained on synthetic data [8, 14, 15]. While synthetic training data offers controlled and dense point cloud representations, it often leads to a sim-to-real gap as real-world point cloud measurements tend to be noisy and sparse, especially in dynamic and unstructured environments. In contrast, methods using single images as input have gained attention for their real-world reconstruction capability without the need for depth information [16, 17, 18]. However, many of these image-based approaches are not optimized for online inference, making them unsuitable for downstream applications in robotics, where online feedback is essential. For instance, Xu et al. [19] proposes an instant image-to-3D framework to generate high-quality 3D assets, but requires up to 10 seconds per frame, limiting its practicality for scenarios demanding real-time processing.

B Device Synchronization

To synchronize devices, we rely on a Linear Timecode (LTC) signal provided by an Atomos Ultrasync device. The cameras of the MVS have a leader/follower architecture, where the internal clocks of the follower cameras are synchronized to one single leader camera, which reads the LTC signal. In addition to the MVS control system, we use two desktop PCs to read the additional data streams: a Robot PC that reads the robot data and the streams of the two RealSense D405 cameras and a dedicated Kinect PC that reads the streams of the two Azure Kinect devices. The robot PC is synchronized with the capture system by reading the same LTC signal provided by the Atomos Ultrasync device. The Kinect cameras are hardware-synchronized with each other. Their synchronization with the capture system is achieved retrospectively by comparing the current timecode displayed on a screen in the camera frames of the Kinect and the camera frames of the capture system. An overview of the architecture is illustrated on Figure 2 (Right).

C Training Loss and Evaluation Metrics

All architectures are end-to-end trained using the same loss. We include the weights of the DinoV2 transformer during backpropagation for finetuning. The main point face distance (PFD) criterion $\mathcal{L}_{\text{PFD}}$ accounts for the global deformation of the objects, which computes the average squared distance $d(\boldsymbol{p}, \boldsymbol{f})$ from the set of sampled points $\boldsymbol{p}_i \in \mathcal{P}$ on the predicted mesh to the nearest faces in the set of triangular faces $\boldsymbol{f}_i \in \mathcal{F}$ of the ground truth mesh and vice versa (eq. (1)). Moreover, to deal with the local deformations generated in the poking region, we add a region-of-interest (ROI) loss $\mathcal{L}_{\text{ROI}}$ (eq. (2)) that computes the unidirectional chamfer distance from the points $\boldsymbol{p}_i$ in the ROI to the set of sampled points $\boldsymbol{q}_i \in \mathcal{Q}$ of the ground truth mesh. The ROI is defined using the indicator function $\mathbb{I}(\mathcal{C}(\boldsymbol{p}_i))$, which evaluates to 1 if point $\boldsymbol{p}_i$ is close enough to the contact point $\boldsymbol{t}$ according to a threshold ϵ, and if the minimum vertical component of the contact point $\boldsymbol{p}_{i,y}$ is bigger than the minimum vertical coordinate across all the vertices $y_{\min}$ scaled by a factor (eq. (3)).

$$\mathcal{L}_{\text{PFD}} = \frac{1}{|\mathcal{P}|} \sum_{\boldsymbol{p}_i \in \mathcal{P}} \min_{\boldsymbol{f}_j \in \mathcal{F}} d(\boldsymbol{p}_i, \boldsymbol{f}_j) + \frac{1}{|\mathcal{F}|} \sum_{\boldsymbol{f}_j \in \mathcal{F}} \min_{\boldsymbol{p}_i \in \mathcal{P}} d(\boldsymbol{f}_j, \boldsymbol{p}_i) \,, \tag{1}$$

$$\mathcal{L}_{\text{ROI}} = \frac{1}{|\mathcal{P}|} \sum_{\boldsymbol{p}_i \in \mathcal{P}} \mathbb{I}(\mathcal{C}(\boldsymbol{p}_i)) \cdot \min_{\boldsymbol{q}_j \in \mathcal{Q}} \|\boldsymbol{p}_i - \boldsymbol{q}_j\|^2 \,, \tag{2}$$

$$\mathcal{C}(\boldsymbol{p}_i) = (\|\boldsymbol{p}_i - \boldsymbol{t}\| \leq \epsilon) \wedge (\boldsymbol{p}_{i,y} > 0.2 \cdot y_{\min}) \,. \tag{3}$$

The total loss is then set as $\mathcal{L} = \mathcal{L}_{\text{PFD}} + 0.5\, \mathcal{L}_{\text{ROI}}$.

During training, we reposition and re-scale all meshes into a cube of unit size ($[-0.5, 0.5]^3$) to maintain a consistent scale across all objects. The losses $\mathcal{L}_{\text{PFD}}$ and $\mathcal{L}_{\text{ROI}}$ are computed in this normalized scale. Additionally, we calculate the relative point-to-face distance (RPFD) by dividing $\mathcal{L}_{\text{PFD}}$ by the average point-to-face distance between the template mesh $\boldsymbol{M}_{\text{T}}$ and the ground truth mesh $\boldsymbol{M}_{\text{GT}}$. An RPFD value below 1 indicates that the predicted deformed mesh $\boldsymbol{M}_{\text{P}}$ is closer to the ground truth than the undeformed template, with smaller values indicating better accuracy.

To further assess the prediction accuracy, we evaluate two additional metrics between the predicted mesh and the ground truth mesh in their original scale: the unidirectional L1 Norm Chamfer Distance CD_{UL1} (eq. (4)) and the volumetric Jaccard Index J (eq. (5)), which we defined in terms of the volume operator V. The two metrics provide insights into the L1 Norm surface distance and the volume overlap ratio, respectively.

$$\text{CD}_{\text{UL1}} = \frac{1}{|\mathcal{P}|} \sum_{\boldsymbol{p}_i \in \mathcal{P}} \min_{\boldsymbol{q}_j \in \mathcal{Q}} \|\boldsymbol{p}_i - \boldsymbol{q}_j\|_1 \,, \tag{4}$$

$$J(\boldsymbol{M}_{\text{A}}, \boldsymbol{M}_{\text{B}}) = \frac{V\left(\boldsymbol{M}_{\text{A}} \cap \boldsymbol{M}_{\text{B}}\right)}{V\left(\boldsymbol{M}_{\text{A}} \cup \boldsymbol{M}_{\text{B}}\right)} \,. \tag{5}$$

D 3D Printing Details

All 3D printed objects were printed using thermoplastic polyurethane (TPU) Filaflex Shore 60A Pro White filament on Prusa MK3S+ and Prusa XL 3D printers equipped with 0.4mm nozzles. The mechanical properties of the filament are presented in Table 4.

Table 4: Mechanical Properties of Filaflex shore 60A Pro TPU provided by the manufacturer.

Mechanical properties	**Value**	**Unit**	**Test method according to**
Tensile strength	26	MPa	DIN 53504-S2
Stress at 20% elongation	1	MPa	DIN 53504-S2

The printing parameters of the 3D printed objects are summarized in Table 5, where the infill used for all objects is the isotropic gyroid pattern with uniform properties and behavior in all directions. Example of the gyroid pattern can be seen in Figure 6.

Table 5: Printing parameters of 3D printed objects featured in the PokeFlex dataset.

Object	**Infill density [%]**	**Layer thickness [mm]**	**Perimeters**	**Bottom layers**	**Top layers**
Bunny [20]	10	0.2	3	3	3
Cylinder	10	0.15	2	3	3
Heart [21]	10	0.2	3	3	3
Pizza [22]	10	0.2	3	3	3
Pyramid	8	0.2	3	3	3

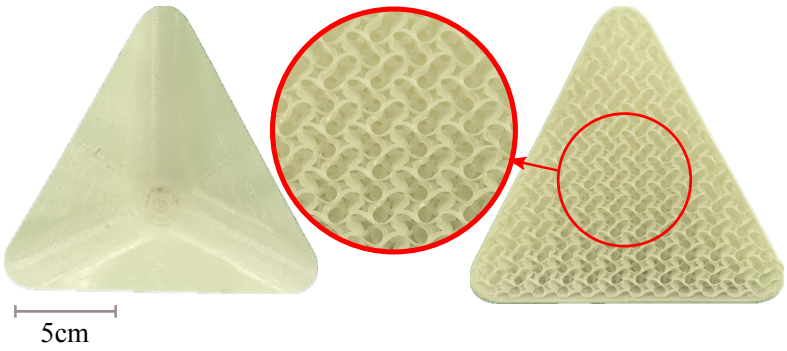

Figure 6: Top (**Left**) and bottom (**Right**) view of 3D printed pyramid, with a close-up view of the interior gyroid infill pattern.

E Properties of Featured Objects

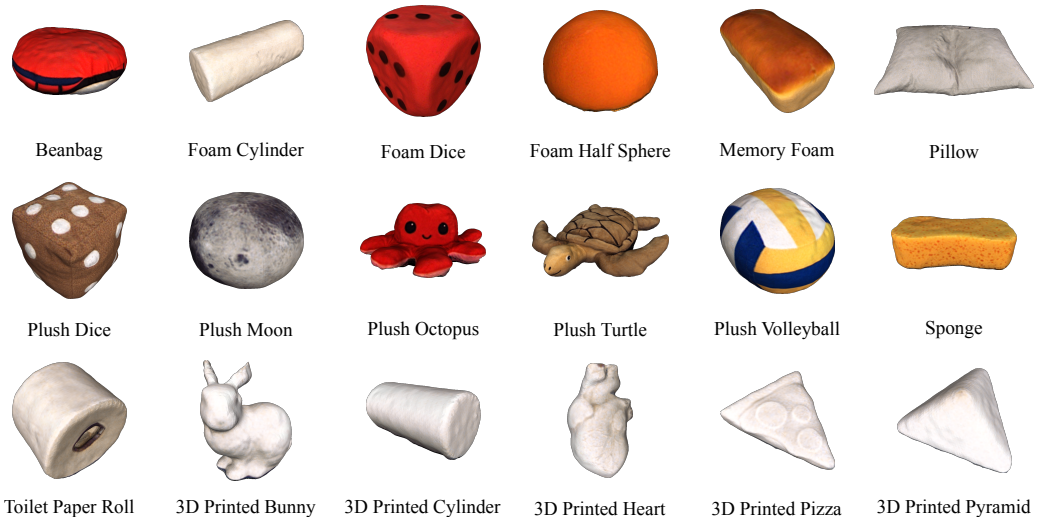

Figure 7: Rest-state reconstructed 3D meshes of all 18 objects featured in the PokeFlex dataset.

Table 6: Physical properties of objects featured in the PokeFlex dataset. Dimensions of sphere-like objects are described by their diameter (D). Cylinder-like objects are characterized by their diameter (D) and height (H). For objects with irregular or complex shapes, dimensions are provided using a bounding box defined by length (L), width (W), and height (H). Stiffness of the objects is estimated according to the method described in Section 3.

Object	Weight [g]	Dimensions [cm]	Est. stiffness [N/m]	Frames	Deformations
Beanbag	184	DxH: 26x9	523	1084	363
Foam cylinder	153	DxH: 12x29	250	990	407
Foam dice	140	L: 15.5	748	1220	738
Foam half sphere	41	D: 15	1252	619	384
Memory foam	213	LxWxH: 17.5x8.5x7	395	420	141
Pillow	975	LxWxH: 58x50x10	474	1085	565
Plush dice	340	L: 22	149	1259	567
Plush moon	151	D: 17	366	959	517
Plush octopus	130	LxWxH: 22x22x11	325	1085	525
Plush volleyball	303	D: 22	323	1099	604
Plush turtle	194	LxWxH: 35x30x10	1035	930	427
Sponge	28	LxWxH: 22x12x6.1	1045	1237	772
Toilet paper roll	134	DxH: 10.5x9.5	2156	600	295
3D printed bunny	105	LxWxH:13x9x15	950	1127	593
3D printed cylinder	223	DxH: 10x20	585	1020	574
3D printed heart	100	LxWxH: 16x9x10	1198	940	444
3D printed pizza	68	LxWxH:18x15x3	3879	680	288
3D printed pyramid	48	LxWxH: 14.5x14.5x7	861	420	193

The PokeFlex dataset consists of 18 objects, whose rest-state reconstructed meshes are shown in Figure 7. For each of the object, we summarize the physical properties and the number of frames in Table 6. The Frames column of the table presents the total captured frames of the poking sequences for each object, and the Deformations column gives the number of active poking frames after the data curation, i.e., discarding the frames where the robot arm is not in contact with the objects. It is worth noting that we report only the effective number of paired time frames in our table, in contrast to the total number of samples, which is computed as the number of time frames multiplied by the number of cameras.

For the dropping protocol, we recorded 3 sequences of 1 second at 60 fps for each object, summing up to 180 time frames per object. Figure 8 shows two reconstructed deformed mesh sequences for poking a foam dice and dropping the plush octopus, respectively.

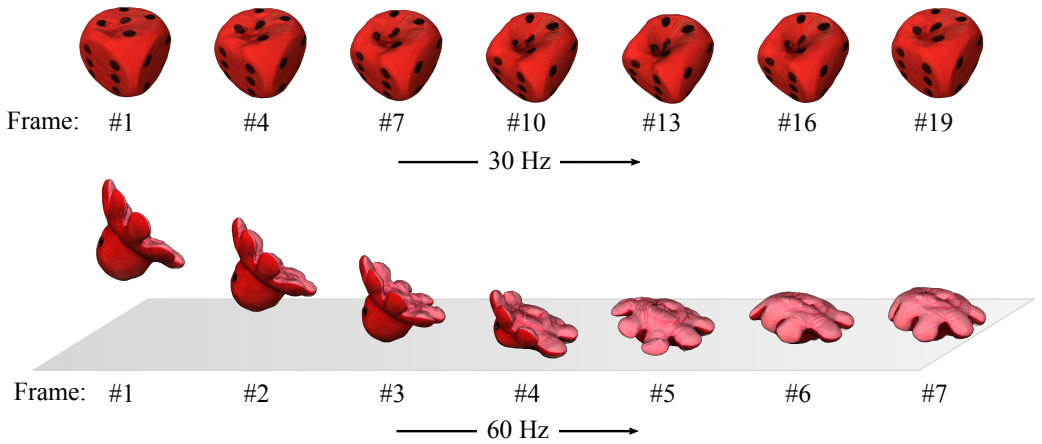

Figure 8: **Top:** Mesh reconstructions of foam dice for a poking sequence shown in every third frame. **Bottom:** Mesh reconstructions of plush octopus for a dropping sequence.

F Training Details

The hyperparameters used to train the models in Section 3 are listed in Table 7.

Table 7: Training hyperparameters.

Hyperparameters	**Value**
Learning rate	1e-4
Batch size	16 (5 objects) / 8 (1 object)
Optimizer	Adam
Weight decay	5e-6
Learning rate scheduler	Cosine
Minimum learning rate	1e-7
Epochs	200

G Inference Speed for Different Input Data Modalities

Table 8 shows the measured inference rates for our five proposed models with different input data modalities. The rate is tested with an AMD Ryzen 7900 x 12 Core Processor CPU and NVIDIA GeForce RTX 4090 GPU with 24GB memory.

Table 8: Inference rate for proposed model configurations.

Input	**Inference Speed**
Images	115 Hz
Forces	215 Hz
Images + forces	110 Hz
Point clouds (5000 points)	106 Hz
Point clouds (100 points)	195 Hz

H Additional Experiment

Table 9: Prediction performance for four models trained on a single viewpoint from different cameras for a single object. Arrows indicate that a better performance is either higher ↑ or lower ↓.

Input	$\mathcal{L}_{\mathbf{PFD}} \cdot 10^3 \downarrow$	$\mathcal{L}_{\mathbf{ROI}} \cdot 10^3 \downarrow$	**RPFD** ↓	$\mathbf{CD}_{\mathbf{UL1}}$**[mm]** ↓	$J(M_{\mathbf{P}}, M_{\mathbf{GT}}) \uparrow$
Volucam	2.98	3.92	0.646	5.392	0.892
RealSense	3.04	4.98	0.494	5.221	0.899
Kinect	4.21	6.32	0.752	6.456	0.845
Rendered	**2.15**	**3.47**	**0.350**	**4.700**	**0.914**

Learning from RGB images of different cameras. In this experiment, we train different models to predict meshes using sequences of RGB images only. Each model was trained for a specific camera, namely Volucam (capture system), Intel RealSense D405, Azure Kinect, and a Virtual camera (Images rendered from ground truth mesh). The viewpoint of each camera is different. For training, we use only deformation sequences of the foam dice. The performance of the different models is reported in Table 9. The training hyperparameters used for this and the following experiments are reported in Appendix F.

I Discussion

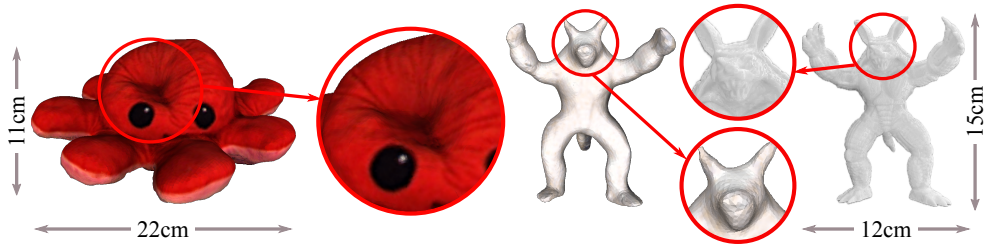

Figure 9: Examples of reconstructed ground truth meshes for medium (**Left**) and small (**Right**) size objects in deformed states. Reconstruction of fine-grained details is a limitation of our current setup (close-up views on the Right).

Quality of ground-truth meshes. The overall geometry of the objects in the dataset, in static configurations, is well captured by the meshes reconstructed with the MVS as shown in Figure 7, even though the system's intended use is the reconstruction of human-size objects. Furthermore, the proposed poking protocol, using a transparent acrylic stick, helps prevent occlusions at the contact point, leading to detailed reconstruction of objects even when they undergo deformations, as can be seen in Figure 9 (Left). However, reconstruction of fine-grained details for smaller objects such as the 3D-printed Stanford armadillo [23] remains challenging with the current setup of the professional capture system, as seen in Figure 9 (Right). Better fine-grained reconstruction results can be expected by rearranging the cameras in a smaller workspace.

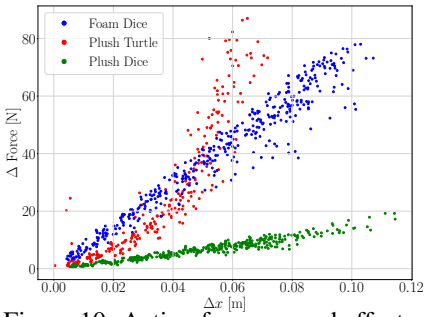

Figure 10: Acting force vs. end effector displacement, shown across all frames for three objects from PokeFlex.

Estimated stiffness. The estimated stiffness that we provide for the featured objects is only intended to offer insights into the range of material properties included in PokeFlex. The simple linear interpolation method using RANSAC can successfully characterize the linear Hookean behavior of objects such as the foam or plush dice shown in Figure 10. More sophisticated approaches, like the

ones presented by Sundaresan et al. [24] and [25] leveraging differentiable simulation, are needed to better characterize the nonlinear behavior exhibited by thinner objects such as the plush turtle.

Learning from RGB images of different cameras.
The results in Table 9 show that the best performance is obtained using synthetic RGB images rendered from ground-truth meshes. Regarding real-world data streams, the model trained from the Volucam cameras performs the best in terms of validation losses. However, the model trained using the Realsense cameras performs the best in terms of additional metrics (RPFD, CD_{UL1}, $J(\boldsymbol{M}_{\text{P}}, \boldsymbol{M}_{\text{GT}})$). The small variations in performance show that good-performing models can be trained using camera sensors that are external to the professional capture system, even if they have different viewpoints.

Multi-object mesh reconstruction from different modalities. Table 2 shows that the point clouds yield the best performance among all data modalities. Specifically, the dense synthetic point clouds perform best in terms of validation losses and RPFD, while Kinect point clouds perform best in terms of CD_{UL1} and $J(\boldsymbol{M}_{\text{P}}, \boldsymbol{M}_{\text{GT}})$. Furthermore, Table 2 also reveals that $\mathcal{L}_{\text{ROI}}$ is bigger than $\mathcal{L}_{\text{PFD}}$ for most models, which can be expected since the majority of deformation occurs in the ROI, with the exception of the model trained with the robot data. The good performance of such a model in $\mathcal{L}_{\text{ROI}}$ could be attributed to the fact that the contact position is explicitly provided as part of the robot data. The model trained with the combination of images and robot data outperforms those using images or robot data alone, hinting at the effectiveness of our cross-attention mechanism.

To analyze the levels of accuracy across multiple objects, we focus on the image-based mesh reconstruction model. Figure 11 shows $J(\boldsymbol{M}_{\text{P}}, \boldsymbol{M}_{\text{GT}})$ and RPFD for only 3 objects separately, for clarity of visualization. The horizontal axis is the Jaccard distance, which indicates the level of deformation of the ground truth mesh with respect to the rest-state template mesh, defined as $d_J(\boldsymbol{M}_{\text{T}}, \boldsymbol{M}_{\text{GT}}) = 1 - J(\boldsymbol{M}_{\text{T}}, \boldsymbol{M}_{\text{GT}})$. Figure 11 shows that the best prediction performance is obtained for the foam dice, having the highest Jaccard Index and the lowest RPFD.

In contrast, for low deformation regimes (small values of $d_J(\boldsymbol{M}_{\text{T}}, \boldsymbol{M}_{\text{GT}})$), the sponge exhibits a lower accuracy, reaching values higher than 1 for the RPDF metric. Such high values correspond to a performance worse than that of predicting the rest-state mesh. Both performance metrics reported in Figure 11 show a negative correlation with the Jaccard distance, indicating that the prediction accuracy of our models decreases for larger deformations.

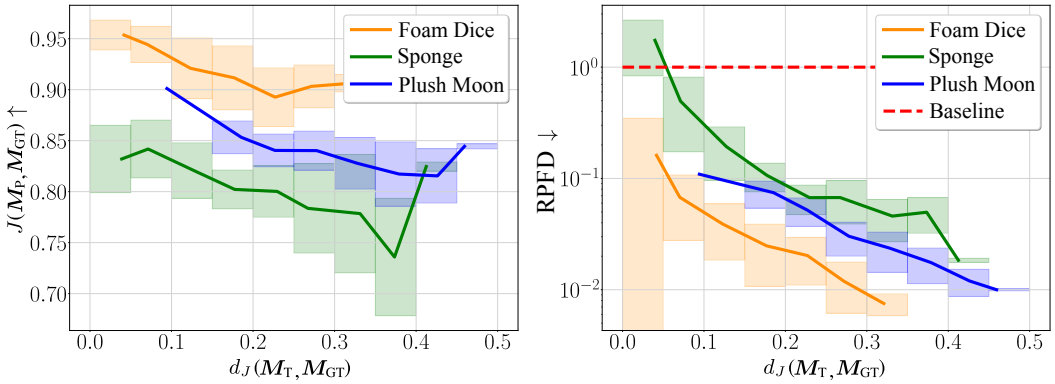

Figure 11: Validation accuracy for image-based mesh reconstruction, evaluated by Jaccard Index J (**Left**) and RPFD (**Right**), plotted against the deformation level quantified by Jaccard distance d_J.