ManiGaussian: Dynamic Gaussian Splatting for Multi-task Robotic Manipulation

Supplementary Material

In this supplementary material, we provide additional details and experiments not included in the main paper due to limitations in space.

- Appendix A: Details of the RLBench dataset and the training pipeline used in our experiments.
- Appendix B: Additional implementation details of our ManiGaussian.
- Appendix C: Supplementary quantitative analysis.
- Appendix D: Supplementary qualitative analysis.

A Details of RLBench

RLBench Dataset. In this section, we provide a concise overview of the RLBench [4] dataset and our training pipeline. Table 1 is an overview of the 10 selected tasks we use in the experiments. Our task variations include randomly sampled colors, sizes, counts, placements, and categories of objects. We have a color palette of 20 shades, including red, maroon, lime, green, blue, navy, yellow, cyan, magenta, silver, gray, orange, olive, purple, teal, azure, violet, rose, black, and white. The size of the objects is categorized into two types: short and tall. The number of objects can be either 1, 2, or 3. Other properties vary depending on the specific task. Furthermore, objects are randomly arranged on the tabletop within a certain range, adding to the diversity of the tasks. In the ablation study, we adopt the task classification from [3] to group the RLBench tasks of Table 1 into 6 categories according to their key challenges. The task groups include:

- The `Planning` group contains tasks with multiple subtasks. The included tasks are: `meat off grill` and `push buttons`.
- The `Long` group includes long-term tasks that requires more than 10 keyframes. The included tasks are: `put in drawer` and `stack blocks`.
- The `Tools` group requires the agent to grasp an object to interact with the target object. The included tasks are: `slide block`, `drag stick` and `sweep to dustpan`.
- The `Motion` group requires precise control, which often causes failures due to the predefined motion planner. The included task is: `turn tap`.
- The `Screw` group requires gripper rotation to screw an object. The included task is: `close jar`.
- The `Occlusion` group involves tasks with severe occlusion problems from certain views. The included task is: `open drawer`.

Training Pipeline. To learn the policy, we uniformly sample a group of expert episodes from all the task variations, and then randomly choose an input-action pair for each of the tasks to form a batch. Other sampling strategies (*e.g.*, Auto-λ [6]) are also available. To simplify the tasks, the agent is assumed to access a

Table 1: Selected tasks.

Task	Type	Variations	Keyframes	Instruction Template
`close jar`	color	20	6.0	*"close the _ jar"*
`open drawer`	placement	3	3.0	*"open the _ drawer"*
`sweep to dustpan`	size	2	4.6	*"sweep dirt to the _ dustpan"*
`meat off grill`	category	2	5.0	*"take the _ off the grill"*
`turn tap`	placement	2	2.0	*"turn _ tap"*
`slide block`	color	4	4.7	*"slide the block to _ target"*
`put in drawer`	placement	3	12.0	*"put the item in the _ drawer"*
`drag stick`	color	20	6.0	*"use the stick to drag the cube onto the _ target"*
`push buttons`	color	50	3.8	*"push the _ button, [then the _ button]"*
`stack blocks`	color, count	60	14.6	*"stack _ _ blocks"*

predefined motion planner (*e.g.*, RRT-Connect), so that the input-action pairs are determined as the bottleneck end-effector poses (*i.e.*, keyframes) within each demonstration based on empirical rules: a pose is determined as a keyframe if the end-effector changes state (*e.g.* close the gripper) or its velocities approach zero [1, 2, 5, 7, 8]. This setting simplifies the sequential decision-making problem into predicting the next optimal keyframe action based on the current observation, which can also be interpreted as a classification task.

B Additional Implementation Details

In this section, we detail the architectural design of each submodule in our Gaussian world model. For more details, please refer to our code.

Representation model. The representation model q_ϕ is the same with [8], which is not the main contribution in this paper. The representation model utilize a shallow 3D UNet to encode the voxel $\in \mathbb{R}^{100^3 \times 10}$ (RGB features, coordinates, indices, and occupancy) into the high-level visual features $\mathbf{v}^{(t)} \in \mathbb{R}^{100^3 \times 128}$.

Gaussian regressor. Given the current features $\mathbf{v}^{(t)}$ encoded by the representation model q_ϕ, we pass it through a generalizable Gaussian regressor g_ϕ to infer the Gaussian distribution $\theta^{(t)}$ directly. The Gaussian regressor is designed as a lightweight multi-head neural network, where each head is responsible for predicting a specific feature. It consists of: (1) a position offset head that predicts the per-pixel 3D center offset $\in \mathbb{R}^3$, (2) a color head that predicts the coefficients of the spherical harmonic basis $\in \mathbb{R}^{12}$, (3) a rotation head with normalization that predicts the rotation quaternion $\in \mathbb{R}^4$, (4) a scaling head with exponential activation that outputs the scaling factor $\in \mathbb{R}^3$, (5) an opacity head with sigmoid activation that predicts the opacity $\in \mathbb{R}^1$. (6) a semantic head that predicts the semantic feature $\in \mathbb{R}^3$.

Deformation predictor. After obtaining the current visual features $\mathbf{v}^{(t)}$, Gaussian embedding $\theta^{(t)}$ and action $a^{(t)}$, we parameterize the transition process as a deformation predictor p_ϕ to predict the deformation $\Delta\mu_i^{(t)} \in \mathbb{R}^3$ and $\Delta r_i^{(t)} \in \mathbb{R}^4$ of each Gaussian, resulting in the future Gaussian embedding $\theta^{(t+1)}$. The deformation predictor is a fully-connected network with residual connections.

Hyperparameters. The hyperparameters used in ManiGaussian are shown in Table 2. To train the robotic manipulation agent, we use $\lambda_{\text{Geo}} = 0.01$, $\lambda_{\text{Sem}} = 0.0001$ and $\lambda_{\text{Dyna}} = 0.001$ to focus on the action prediction. Other hyperparameters are in line with previous works [7, 8] for fair comparison.

Table 2: Hyperparameters.

Hyperparameter	Value
training iteration	100k
image resolution	128×128
voxel resolution	$100 \times 100 \times 100$
batch size	2
optimizer	LAMB
learning rate	0.0005
weight decay	0.000001
Number of Gaussian points	16384
λ_{Geo}	0.01
λ_{Sem}	0.0001
λ_{Dyna}	0.001

C Additional Quantitative Analysis

We provide further ablation study on different implementation choices in our ManiGaussian. Table 3 presents the impact of different balance hyperparameters on the overall performance, from which we can conclude that the balance of each loss item is important to learn an optimal manipulation policy.

Table 3: Impact of Balance Hyperparameters.

λ_{Geo}	λ_{Sem}	λ_{Dyna}	Planning	Long	Tools	Motion	Screw	Occlusion	**Average**
0.01	0	0.00001	42.0	24.0	48.0	48.0	28.0	72.0	42.4
0.01	0	0.0001	54.0	12.0	44.0	52.0	28.0	80.0	42.4
0.01	0	0.001	54.0	10.0	49.3	64.0	24.0	72.0	**43.6**
0.01	0.00001	0	48.0	8.0	34.7	48.0	24.0	64.0	35.2
0.01	0.0001	0	46.0	8.0	53.3	64.0	28.0	56.0	**41.6**
0.01	0.001	0	46.0	2.0	37.3	60.0	40.0	68.0	37.6
0.01	0.0001	0.001	40.0	14.0	60.0	56.0	28.0	76.0	**44.8**

D Additional Qualitative Analysis

We provide 9 additional comprehensive episodes generated by our ManiGaussian and the state-of-the-art generative method GNFactor [8] in the attached video file (*demo.mp4*). In the long-term *"stack 2 rose blocks"*, *"put the item in the*

bottom drawer" and *"take the steak off the grill"* tasks, ManiGaussian completes the human instructions in the correct order with the understanding of high-level scene dynamics mined by the Gaussian world model. In the *"sweep dirt to the short dustpan"* and *"use the stick to drag the cube onto the azure target"* tasks that involve tool usage, our ManiGaussian succeeds in solving the tasks by correctly understanding the low-level scene dynamics of objects in contact. In the *"slide the block to green target"*, *"turn left tap"*, *"close the azure jar"* and *"open the bottom drawer"* tasks that require semantic understanding and precise control, our ManiGaussian can successfully comprehend the semantic information to interact with the correct object instance, while the baseline method often confuses different instances.

References

1. Chen, S., Garcia, R., Schmid, C., Laptev, I.: Polarnet: 3d point clouds for language-guided robotic manipulation. arXiv preprint arXiv:2309.15596 (2023)
2. Gervet, T., Xian, Z., Gkanatsios, N., Fragkiadaki, K.: Act3d: 3d feature field transformers for multi-task robotic manipulation. In: CoRL. pp. 3949–3965 (2023)
3. Guhur, P.L., Chen, S., Pinel, R.G., Tapaswi, M., Laptev, I., Schmid, C.: Instruction-driven history-aware policies for robotic manipulations. In: CoRL. pp. 175–187. PMLR (2023)
4. James, S., Ma, Z., Arrojo, D.R., Davison, A.J.: Rlbench: The robot learning benchmark & learning environment. RA-L (2020)
5. James, S., Wada, K., Laidlow, T., Davison, A.J.: Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In: CVPR (2022)
6. Liu, S., James, S., Davison, A.J., Johns, E.: Auto-lambda: Disentangling dynamic task relationships. arXiv preprint arXiv:2202.03091 (2022)
7. Shridhar, M., Manuelli, L., Fox, D.: Perceiver-actor: A multi-task transformer for robotic manipulation. In: CoRL (2023)
8. Ze, Y., Yan, G., Wu, Y.H., Macaluso, A., Ge, Y., Ye, J., Hansen, N., Li, L.E., Wang, X.: Gnfactor: Multi-task real robot learning with generalizable neural feature fields. In: CoRL. pp. 284–301. PMLR (2023)