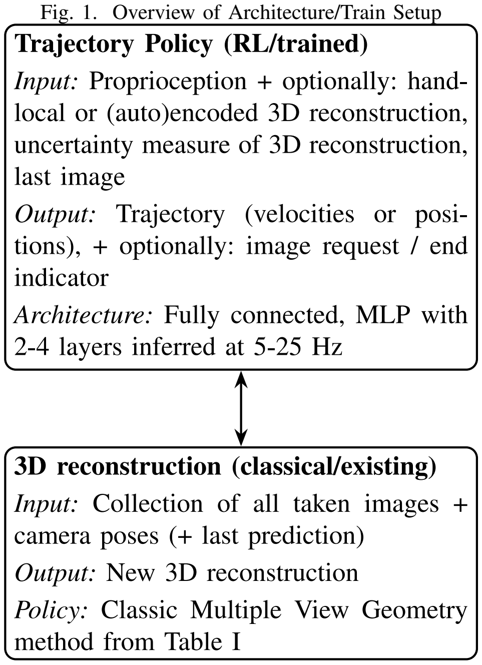
ADLR-Project 18: Efficient Environment Exploration and 3D Reconstruction with Reinforcement Learning and Multiple View Geometry

Frank Zillmann
Department of Computer Engineering
Technical University of Munich (TUM)
Munich, Germany
frank.zillmann@tum.de

Fig. 1. Overview of Architecture/Train Setup



Trajectory Policy (RL/trained)

Input: Proprioception + optionally: hand-local or (auto)encoded 3D reconstruction, uncertainty measure of 3D reconstruction, last image

Output: Trajectory (velocities or positions), + optionally: image request / end indicator

Architecture: Fully connected, MLP with 2-4 layers inferred at 5-25 Hz

3D reconstruction (classical/existing)

Input: Collection of all taken images + camera poses (+ last prediction)

Output: New 3D reconstruction

Policy: Classic Multiple View Geometry method from Table I

I. Objective

The goal is to develop a policy capable of autonomously exploring the workspace within a robot arm's reach and producing a 3D reconstruction, such as a *(Truncated) Signed Distance Field (TSDF)* that can be used for further applications such as collision checking or trajectory optimization. The robot arm is mounted vertically on a table with several obstacles placed within its reach. With a camera at the hand, the policy should explore the workspace. As input, the policy can get proprioceptive data - such as the positions/angles and velocities of the joints or the camera -, the rendered images of the camera. As output, it should provide an accurate 3D reconstruction with respect to the robot's base. The policy should learn to avoid collisions, remain computationally efficient, and generalize robustly to unseen environments. Therefore, I will combine a lightweight Trajectory Policy trained with Reinforcement Learning with a classical approach from Multiple View Geometry (not trained on shape datasets).

II. Related Work

Sharing Meli's and Dehio's interest in getting a precise model of the fixed obstacles in a robot arm's reach i.e. a cell model with low effort [1], I want to automate the exploration process.

The literature research for the selection of a suitable 3D reconstruction method has shown a variety of suitable methods.

TABLE I
Comparison of Multi View 3D Reconstruction Methods

Method	**Uses Poses**	**Uses Depth**	**Output**	**Online**	**Uncertainty**	**Time**
Liao [2]	Yes	No	Neural SDF	Partial	Latent variance	minutes
SparseCraft [3]	Yes	No	Neural SDF+color	–	Photometric residuals	minutes
COLMAP [4]	Yes/No	Optional	Point cloud	Yes	Reprojection error	seconds
TSDF Fusion [5]	Yes	Yes	Voxel TSDF	Yes	Voxel observation count	0.5–2 s
Open3D TSDF [6]	Yes	Yes	TSDF + mesh	Yes	Weight map	<1 s

Many modern approaches use *Neural Fields* for *SDF* or other *occupancy maps* [2], [3], [7]. While these methods yield high quality outputs and can provide uncertainty measures [2], they are relatively slow (often 5–30 minutes compute per scene), and therefore unsuitable for most RL methods. They could, however, serve as a potential post-training enhancement to replace a simpler reconstruction method.

Therefore, classical multiple view geometry approaches will be used. Specifically, *Open3D*'s [6] *TSDF* method will be the first choice due its quick inference (C++), good documentation, and suitable output formats as *TSDF* (*SDF* but clipped at positive and negative threshold) and optionally mesh (via marching cubes). *COLMAP* [4], might be an alternative as it does not necessarily need depth information and the reprojection error provides a well interpretable uncertainty estimate.

III. Technical Outline

First, I will familiarize myself with the software components used — *robosuite*, *MuJoCo*, and *Stable Baselines 3* — to ensure proper integration.

As a first coding task, the described environment will be added to *robosuite* and needs to be verified via some camera renderings and a dummy policy. A ground truth can be constructed with existing Mesh to *SDF* methods [8].

Next, the proposed policy, its architecture, and the training algorithm will be implemented according to Figure 1. Specifically, the trajectory policy will be designed using *pytorch* and both it and the 3D reconstruction method need to be integrated in the training pipeline and *robosuite*. In the beginning, the Trajectory Policy policy will be kept simple, using little data (proprioception + few extra information) and only a few fully connected layers. Depending on the outcome of the few experiments, this leaves room for later iterations to provide richer input information e.g. the last image, uncertainty estimates, etc. and adapt the architecture of the neural network to better fit the problem e.g. convolutions, residual connections, etc.

Furthermore, a loss function for the predicted 3D reconstruction needs to be implemented and a reward function will be designed using it. The reward function will be based on the difference between the predicted 3D reconstruction and the ground truth of the environment, combined with large penalties for collisions and smaller penalties for factors such as joint torques or number of image renderings.

Finally, initial training runs will employ *Proximal Policy Optimization* [9] as baseline, with the option to switch in case of need for more sample efficiency due to performance issues.

Further ideas / extensions:

- In addition to the image input, one could render the currently predicted SDF (in some way) for the current camera position and orientation and provide that as an extra channel to the policy so that it gets the possibility to better compare between reality and the currently predicted SDF.
- Instead of ground truth, one could use the uncertainty esitimates of the Multiple View Geometry Method in the reward function and study if and to what degree performance degrades.

References

[1] G. Meli and N. Dehio, “Robot cell modeling via exploratory robot motions,” 2025. [Online]. Available: https://arxiv.org/abs/2502.01484

[2] Z. Liao and S. L. Waslander, “Multi-view 3d object reconstruction and uncertainty modelling with neural shape prior,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2024, pp. 3098–3107, arXiv:2306.11739.

[3] M. Younes, A. Ouasfi, and A. Boukhayma, “Sparsecraft: Few-shot neural reconstruction through stereopsis guided geometric linearization,” *arXiv preprint*, vol. arXiv:2407.14257, 2024, eCCV 2024 poster, project page: https://sparsecraft.github.io/.

[4] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, “Pixelwise view selection for unstructured multi-view stereo,” in *European Conference on Computer Vision (ECCV)*, 2016.

[5] B. Curless and M. Levoy, “A volumetric method for building complex models from range images,” in *Proceedings of SIGGRAPH*, 1996.

[6] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3d: A modern library for 3d data processing,” *arXiv preprint arXiv:1801.09847*, 2018.

[7] M. Guo, C. Li, H. Chen, and G. H. Lee, “Unikd: Uncertainty-filtered incremental knowledge distillation for neural implicit representation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024, arXiv:2212.10950, code: https://github.com/dreamguo/UNIKD.

[8] M. Kleineberg, “mesh_to_sdf: Calculate signed distance fields for arbitrary meshes,” https://github.com/marian42/mesh_to_sdf, 2020, mIT License, GitHub repository.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017. [Online]. Available: https://arxiv.org/abs/1707.06347