2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

High-Degrees-of-Freedom Dynamic Neural Fields for Robot Self-Modeling and Motion Planning

Lennart Schulze[1] and Hod Lipson[2]

***Abstract*— A robot self-model is a task-agnostic representation of the robot's physical morphology that can be used for motion planning tasks in the absence of a classical geometric kinematic model. In particular, when the latter is hard to engineer or the robot's kinematics change unexpectedly, human-free self-modeling is a necessary feature of truly autonomous agents. In this work, we leverage neural fields to allow a robot to self-model its kinematics as a neural-implicit query model learned only from 2D images annotated with camera poses and configurations. This enables significantly greater applicability than existing approaches which have been dependent on depth images or geometry knowledge. To this end, alongside a curricular data sampling strategy, we propose a new encoder-based neural density field architecture for dynamic object-centric scenes conditioned on high numbers of degrees of freedom (DOFs). In a 7-DOF robot test setup, the learned self-model achieves a Chamfer-L2 distance of 2% of the robot's workspace dimension. We demonstrate the capabilities of this model on motion planning tasks as an exemplary downstream application.**

I. Introduction

Neural fields paired with differentiable rendering allow learning accurate 3D scene information from pose-annotated 2D images. This is achieved by overfitting a neural network to the scene observed from multiple camera views using a photometric reconstruction loss [1]. After training, the model can be used to render realistic images of the scene from novel camera views. Due to the importance of scene representations in robotics, neural field extensions have evolved focusing on use cases in this area. While most of these approaches [2, 3, 4, 5] use neural fields to capture and utilize information about the robot's environment, such as for reconstruction, navigation, or localization tasks, here we propose to learn neural fields to represent - and control - the robot.

We solve the task of robot self-modeling, the (robot's) ability to acquire a representation of the robot's kinematics from observing its behavior without human interference. Similar to a mental image of oneself, self-models can continually be updated to reflect the state of the robot. This renders them advantageous over classical geometric kinematic models, which are usually engineered once, may be mismatched to the current state of the robot, and are unavailable for unknown robots [6]. For these reasons, learning-based approaches to robot self-modeling emerged. Despite functional, a major drawback is their dependence on supervised samples or, in the self-supervised case, depth annotations in the training distribution. These requirements hinder the readiness of target applications of self-modeling in real-world scenarios where this information is not available, such as after damage to the robot's body during deployment. In particular, a recent approach [7] to learn a full-body kinematic forward model as a neural-implicit representation requires images annotated with depth values from RGB-D cameras. In this work, we propose to solve this obstacle by learning neural fields in a self-supervised manner directly from 2D images only annotated with camera parameters and the dynamic configuration. Consequently, we approach the task of learning a neural-implicit full-body kinematic model from unlabeled kinematic data through training a dynamic neural field that offers downstream compatibility (Fig. 1).

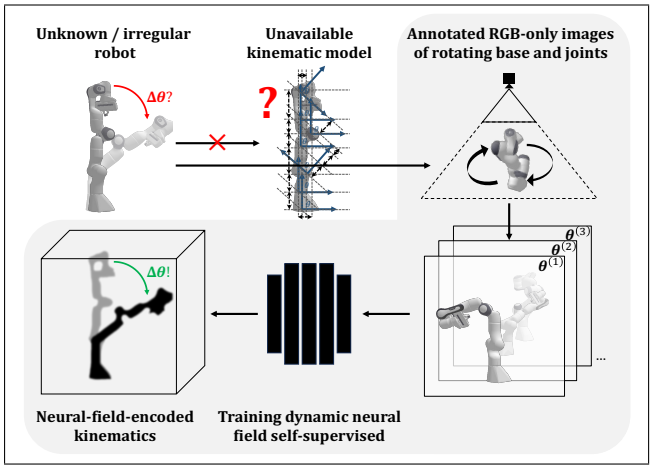

Fig. 1. **Overview of contributions (shaded):** When a kinematic model is unavailable for the robot, 1) our method to collect curricular annotated depth-free image data can be used instead to train 2) a high-DOFs dynamic neural density field which the robot uses as a self-model. 3) Its forward and inverse kinematics capabilities enable motion planning applications.

We achieve this by introducing a new type of *dynamic* neural field. Previous work [8, 9, 10] has extended the static-scene setup of neural radiance fields [1] by establishing time as an additional input dimension next to 3D coordinates, which together are mapped to density and color values. In contrast, in this work we introduce a high number of degrees of freedom (DOFs) that in complex interdependence change local parts of the scene as the conditioning variables for a coordinate-to-density map, which has not been done for robotic applications. Different from methods using deformation from a canonical representation [11], we propose a DOF-encoder-based dynamic neural density field, which is suitable for modeling complex changing scenes beyond robotics.

In summary, this work contributes the following:

[1]Department of Computer Science, Columbia University, New York, NY 10027, USA. lennart.schulze@columbia.edu
[2]Department of Mechanical Engineering, Columbia University, New York, NY 10027, USA. hod.lipson@columbia.edu

- We introduce a curricular data sampling method and neural network architecture to represent high-DOFs object-centric scenes as dynamic neural density fields.
- We use our method to visually learn the first robot self-model without depth information and from a single camera view, and quantify its quality experimentally.
- Extending [7], we discuss and demonstrate downstream applications of neural-field self-modeled kinematics in motion planning.

II. Background and Related Work

Robot self-modeling. A self-model is a task-agnostic, general-purpose representation of a robot's physical shape and structure that can be acquired at any time and continually updated without a human in the loop [12, 13]. The objective of enabling machines to produce a cognitive model of themselves to guide their behavior has been inspired by similar behavior in human beings [14]. In practice, whenever a geometric kinematic model, which captures the spatial relations and physical constraints of the robot's links and joints manually as a result of simulation and engineering, is unavailable, the ability to self-model is required. In particular, when the robot's kinematics are altered, for instance through damage or undocumented body manipulation, the robot can learn an updated self-model without the need to manually re-devise the kinematic model [15, 16].

Approaches to robot self-modeling have leveraged analytical, probabilistic, and evolutionary methods [15, 17, 16]. Learning-based approaches to implicitly represent self-models were first presented in [18], necessitating training samples that are labeled with the end effector position. Similarly, certain approaches [19, 20] pre-determine the set of parameters to learn for a system, identified based on prior knowledge about the shape or function. The most recent, partially self-supervised approach, which constructs an agnostic self-model without such information [7], still requires depth information, which is used to learn an SDF-based occupancy query model. In all approaches, data acquisition plays a crucial role, with strategies ranging from entirely random [7], to interactive [21], and targeted-exploratory [19, 22]. This work builds on the agnostic, neural-implicit class of representation proposed in [7] and removes the depth requirement using neural fields, while introducing a curricular-random training data acquisition strategy.

Neural (radiance) fields. A neural field is a continuous map from any spatial coordinate in 3D space $\mathbf{x} = \begin{bmatrix} x, y, z \end{bmatrix}^T$ to a scalar or vector. In neural *radiance* fields (NeRF) [1], each point is assigned a tuple of density and color. The map is parameterized via a neural network Φ, such as a multi-layer perceptron (MLP), overfit to the specific scene,

$$f_\Phi : (\mathbf{x}, \mathbf{d}) \rightarrow (\sigma, \mathbf{c}) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^3$ is the coordinate vector, $\mathbf{d} \in [0,1]^3, \|\mathbf{d}\| = 1$ is the unit viewing direction, $\sigma \in [0, \infty)$ is the predicted density, and $\mathbf{c} \in [0,1]^3$ is the predicted RGB color. Both to march a ray through the scene to obtain point coordinates $\mathbf{x}$ and to compute $\mathbf{d}$, the camera pose ${}^{w}\boldsymbol{T}_{\boldsymbol{c}}$ is used.

A NeRF is a neural-implicit representation of a scene since novel views can be rendered by querying the learned map, without the need to store 3D information explicitly, such as in point clouds or voxels. From the field over the 3D space, 2D projections to images from arbitrary camera views are rendered via volume rendering [23, 24]: The color of a pixel $\mathbf{C}$ is computed by integrating the product of color, density, and visibility of the points residing on the ray $\mathbf{r}$ that was marched through the scene from the projection plane within the depth view bounds. The visibility T_i of a point depends on the density values of the points between the projection plane and that point. Using quadrature, the integral is approximated on N points, which are sampled in a stratified manner from bins on the ray, as follows:

$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_{i=1}^{N} \hat{T}_i \; \alpha(\sigma_\Phi(\mathbf{x}^{(\mathbf{i})})\delta_i) \; \mathbf{c}_\Phi(\mathbf{x}^{(\mathbf{i})}, \mathbf{d}) \tag{2}$$

$$\hat{T}_i = \exp\left(- \sum_{j=1}^{i-1} \sigma_\Phi(\mathbf{x}^{(\mathbf{j})})\delta_j \right) \tag{3}$$

Here, $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ is the pixel-corresponding ray marched through the scene scaled by depth t; $\alpha(\sigma) = 1 - \exp(-\sigma)$ maps density values into the range $[0,1]$; and $\delta_i = t_{i+1} - t_i$ is the distance between adjacent points on the ray. The differentiable nature of volume rendering allows the MLP to be trained by minimizing a photometric reconstruction loss between training images and renders from the same poses.

Dynamic neural fields have emerged to represent scenes with changing components, for instance over a single time dimension [8, 9, 10, 25, 26, 27, 28, 29]. In extension, numerous works have aimed at modeling controllable human bodies, customarily using prior knowledge or annotations about their shape or multi-view video training data [30, 31, 11, 32, 33, 34]. Related to our work, we propose a new dynamic neural field architecture geared towards shape-unknown objects with many DOFs that are interdependent, trained on single-view images only.

III. Method

A. High-DOFs Dynamic Neural Density Field

We propose to extend neural fields to dynamic scenes in which changes are anchored in interdependent DOFs of the object in the scene. For this purpose, we condition the map f_Φ on the k-dimensional configuration of the object $\boldsymbol{\theta} = \begin{bmatrix} \theta_0, \ldots, \theta_{k-1} \end{bmatrix}^T \in \mathbb{R}^k$ that causes the observed changes:

$$f_\Phi : (\mathbf{x}, \boldsymbol{\theta}) \rightarrow (\sigma) \tag{4}$$

To model the shape, the field does not assign color and is thus independent of the viewing direction. Nonetheless, color can be included for the purpose of training the model via a photometric loss.

Specifically, to learn a self-model of a robot, the map is conditioned on the joint configuration of the robot composed of k joint values and thus learned as a $3 + k$-dimensional neural field. We can subsequently compute density fields in novel configurations and render these from novel views.

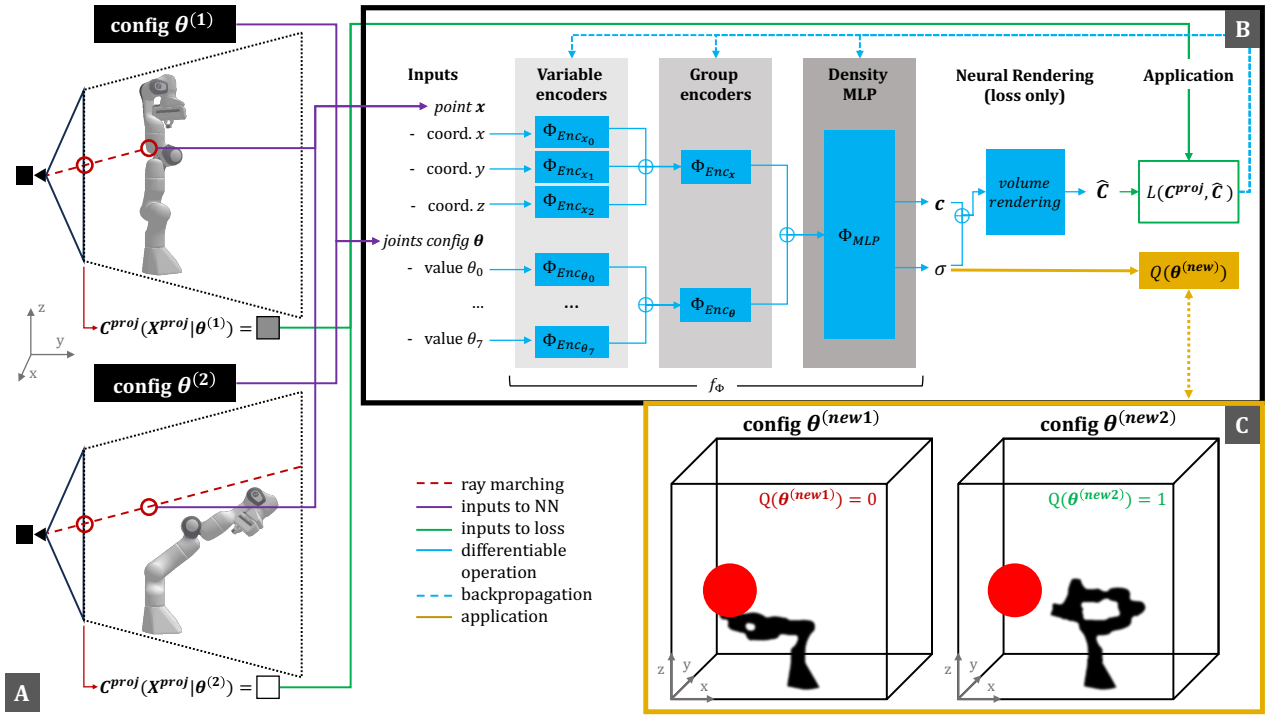

Fig. 2. **Overview of proposed method:** A) Training images of robot in different configurations: While the point coordinates and annotated configuration are the inputs to the neural network, the true color of the pixel is used for the reconstruction loss. B) Neural network architecture: The DOF values and spatial coordinates are individually encoded, concatenated and group-wise encoded, and concatenated and processed to an output density. C) The trained neural density field is used to evaluate the validity of a configuration relative to an obstacle by predicting the densities of points queried from its volume.

Encoder-based architecture. We parameterize the map via a neural network based on [1], that is an MLP with ReLU activations. Extending this architecture by adding $\boldsymbol{\theta}$ as input parameters produces unsatisfactory results. Consequently, we extend [7]'s approach and introduce separate encoders for the spatial and conditioning input variables. This leverages the independence of the coordinates and the DOFs configuration and promises to learn useful representations resulting from the combinations of the constituents of each group.

Applying the shuffled curriculum learning approach introduced below, however, results in continual forgetting of previously learned relationships of DOFs as the training progresses, reducing performance on previously well reconstructed samples. For this reason, we introduce DOF-individual encoders, MLPs that encode each input variable. Since gradients flowing back to the weights of these MLPs will be zero when the joint values are zero in batches in which exclusively other DOFs are sampled, we argue this improves the memorability of useful features for the behavior introduced by each DOF individually. The outputs of the individual encoders are concatenated and chained to the group-based encoders. The complete model is described by:

$$f_\Phi(\mathbf{x}, \boldsymbol{\theta}) = \Phi_{\text{MLP}}\left(\Phi_{\text{Enc}_\mathbf{x}}(\bigoplus_{l=0}^{2}\Phi_{\text{Enc}_{x_l}}(x_l)), \Phi_{\text{Enc}_{\boldsymbol{\theta}}}(\bigoplus_{l=0}^{k-1}\Phi_{\text{Enc}_{\theta_l}}(\theta_l))\right) \quad (5)$$

where $\oplus$ is the concatenation operator and $\Phi(x) = h_n \circ \ldots \circ h_1(x)$ is an n-layer MLP with ReLU activations. The architecture is shown in Fig. 2.

Following [1], we train two models of this type to separately model spatially coarse and fine predictions. The second model evaluates points on the ray sampled from regions of t where the first model has resulted in higher density predictions. All points are used to render the projection of a ray. We find the sinusoidal positional encoding used in [1] to hinder the learning of the physical movement associated with traversing the DOF value ranges. Consequently, we substitute it with the $[-1, 1]$-normalized original joint values, resulting in $3 + k$-dimensional inputs to the model.

Learning from a one-camera setup. To circumvent the need for multiple cameras observing the robot to produce the neural field, which limits real-world applications, we harness the mobility of the robot's base. Given the robot's configuration $\boldsymbol{\theta}$ and the camera pose as camera-to-world transform ${}^{w}\boldsymbol{T}_{\boldsymbol{c}}$, we enforce the first DOF to be the base rotation. For a front-facing camera pointing at the center of the robot with zero pitch and roll, rotating the object at its base is equivalent to rotating the camera about the upward axis. Hence, multi-view consistency for the density predictions can be enabled by assigning $({}^{w}\boldsymbol{T}_{\boldsymbol{c}})' \leftarrow \boldsymbol{R}_{\boldsymbol{z}}(\theta_0)\,{}^{w}\boldsymbol{T}_{\boldsymbol{c}}$ and $\theta_0' \leftarrow 0$.

Curricular training data. Due to the large space of configurations and the serial dependence among the k DOFs, the most distal joint's position depending on all $k-1$ previous DOFs, learning a high-DOFs neural field is difficult. Thus, the training data generation approach is crucial to the success

of the model inferring the correct marginal influence of each DOF. We propose a curriculum-learning-inspired sampling approach. For the set of all DOF indices $\Theta = \{l\}_{l=1}^{k-1}$, we compute the powerset containing all subsets of Θ and sort it in ascending order by magnitude, excluding the empty set: $S_\Theta = \{s\}_{s \subseteq \Theta}$. For each set of DOF indices $s \in S_\Theta$, samples are generated by uniformly randomly sampling values from the permissible ranges of the DOFs in s. The values of the remaining DOFs are fixed to zero. Thus, we encourage learning the contribution of each DOF first by itself and then in combination with other DOFs in order of increasing complexity, until all DOFs are interacting. For example,

$$\boldsymbol{\theta}^{(1,4)} = \begin{bmatrix} 0 & \theta_1 \sim D_1 & 0 & 0 & \theta_4 \sim D_4 & \dots & 0 \end{bmatrix}, \quad (6)$$

where $D_i = \mathcal{U}(\theta_i^{(min)}, \theta_i^{(max)})$. We find shuffling the images such that images with different numbers of active DOFs lie in the same batch to improve the training performance.

Training. We optimize the model via a photometric mean squared error (MSE) loss between ground-truth $\mathbf{C}^{\mathbf{proj}}$ and rendered pixels $\hat{\mathbf{C}}$. Our experiments suggest that keeping the RGB output improves training performance. Nonetheless, the density prediction is the only output kept to be used in the self-model after training. To train density-output-only high-DOFs neural fields, the MSE loss may be used between binarized images and renderings with $\mathbf{c}$ set to black.

B. Neural-Field Self-Model and Applications

Self-model. The trained map f_Φ is an implicit full-body kinematic model of the robot since it enables the reconstruction of its shape conditioned on its joint configuration. Learning this model only from annotated 2D images replaces the need to know the robot geometry altogether.

Motion planning: Reaching a target via inverse kinematics. Extending [7], we demonstrate motion planning as an inherent downstream application of the model. Due to the differentiable forward prediction of the density of a point given the configuration, we can compute the inverse kinematics, that is the configuration such that a point is occupied. For this purpose, the MLP parameters are fixed, and the input joint values are optimized via projected gradient descent (PGD) to minimize the delta to the desired density.

By choosing appropriate points, the robot can, for example, compute how to reach an object. Furthermore, by selecting the initial configuration of the optimization to be the current configuration of the robot and acting in an obstacle-free environment, the inverse kinematics optimization steps can be cast as a path to reach the target. Precisely, given information about the target, we uniformly sample N points from its surface $O_s = \{\mathbf{x}^{(\mathbf{i})}\}_{i=1}^{N}$ and query the robot's density on them, leveraging that density on the surface above a threshold τ indicates touch. In the fine model, starting from a no-touch configuration $\boldsymbol{\theta}^{(\mathbf{0})}$, we minimize the following loss, which will be ≤ 0 when the target is reached:

$$L(\boldsymbol{\theta}, O_s) = \min_{\mathbf{x} \in O_s} \left[-\alpha(f_\Phi(\mathbf{x}, \boldsymbol{\theta}))\right] + \tau \quad (7)$$

The final ReLU activation at Φ_{MLP}'s output unit for σ is removed to produce non-zero gradients. To enforce that the final joint configuration and every step of the optimization are within the joint limits, after each step of size η the joint values are projected back into the k-dimensional ball representing the permissible ranges:

$$\boldsymbol{\theta}^{(j+1)} = \Pi_{\boldsymbol{\theta}^{(min)}}^{\boldsymbol{\theta}^{(max)}} \left[\boldsymbol{\theta}^{(j)} - \eta \frac{\partial L(\boldsymbol{\theta}^{(j)}, O_s)}{\partial \boldsymbol{\theta}^{(j)}}\right] \quad (8)$$

If only the inverse kinematics task is required, random initializations of the configuration can accelerate the optimization.

Motion planning: Configuration space. For more complex constraints and in the presence of obstacles, customary motion planning algorithms can be used with the self-model. Any planning algorithm using the configuration space, that is the binary map over the k-dimensional space of possible configurations indicating which configuration is collision-free, is compatible with the implicit kinematic model. Given the neural density field and information about obstacle(s) in the scene, a configuration is valid if the maximum density of the robot among N uniformly sampled points from the volume of the obstacle O_v is below a threshold. Thus, sampling-based motion planning methods that search the configuration space, such as Probabilistic Roadmap [36] or Rapidly-exploring Random Trees (RRT) [37], can be used. The membership in the configuration space is queried as:

$$Q(\boldsymbol{\theta}) = \begin{cases} True & \text{if: } \max_{\mathbf{x} \in O_v} \left[\alpha(f_\Phi(\mathbf{x}, \boldsymbol{\theta}))\right] < \tau \\ False & \text{else.} \end{cases} \quad (9)$$

IV. Experimental Setup

We demonstrate our method on a simulated 7-DOF robot.

Training distribution. We apply the curriculum data generation described above with 16 different configurations per set s and 6 random base rotations sampled anew for each configuration, totalling $5,588$ annotated 400×400 images. We generate the images in simulation of the Panda robot [38] with 7 joints and a rotatable base ($k = 8$), using the Pybullet simulator [39]. We group batches of 15 images and, to avoid overfitting to one batch, only process $10,240$ rays per image.

Training. We use the described architecture with 3-layer DOF-individual encoder MLPs, 1-layer coordinate encoder MLPs, 2-layer group encoder MLPs, and the final 7-layer density MLP. We train using the Adam optimizer for 1,320,000 steps with a learning rate of $4e-5$ and optimize the parameters of all MLPs together.

Visualizations. To visualize the *predicted* self-model, we produce point clouds by querying the field from two camera poses at the front and side of the scene on the y- and x-axes. Points with alpha values above 0.015 are kept, determining the isolevel. On the fused point cloud, marching cubes [40] reconstruction is applied to generate a triangle mesh, followed by hole repair and Taubin smoothing [41] algorithms. To visualize the *ground truth*, we simulate the true robot model in Pybullet. The ground-truth point cloud for a joint configuration is the fusion of six point clouds from RGB-D images obtained from two camera views per axis, one at either end, with the view centered at the object. The mesh is then produced identically to the predicted mesh.

TABLE I

SPATIAL DISTANCES BETWEEN PREDICTED AND GROUND-TRUTH SELF-MODEL MESHES IN RANDOM TEST CONFIGURATIONS AND ACROSS TEST SET.

Distance metric	config a	config b	config c	config d	config e	test set (n=30)
Chamfer-L2 (m) ↓	.017	.019	.053	.013	.013	.024
Chamfer-L2 (% of workspace-z) ↓	1.35	1.51	4.22	1.06	1.07	1.94
Surface area IoU ↑	.501	.479	.408	.571	.572	.496
Hull volume IoU ↑	.685	.607	.336	.714	.690	.573

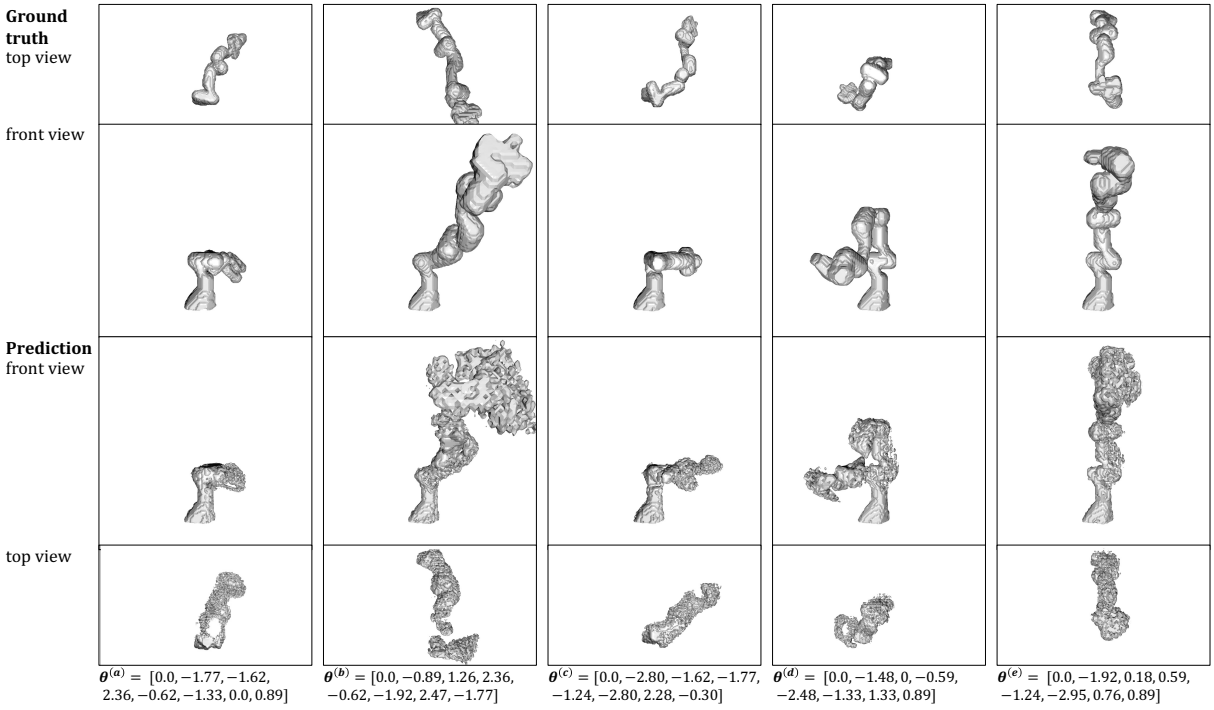

Fig. 3. **Self-model results:** Predicted vs. ground-truth meshes, smoothed and reconstructed via marching cubes from point clouds generated by querying the high-DOFs neural density field in the given random test configuration. The configurations are shown in radians. Please also see suppl. video [35].

Metrics. To assess the quality of our model, we compare the ground-truth against the predicted meshes. First, we use the customary Chamfer-L2 distance, the shortest Euclidean distance of a point in a set to any point in the other set, applied symmetrically and averaged over all points. This returns an average spatial offset per point. We generate the point sets by sampling uniformly from the mesh surfaces. Second, as measures for the spatial similarity of the shapes, we compute two intersection over union (IoU) metrics. Both are based on a union point cloud, constructed by fusion, and an intersection point cloud, constructed by keeping points with a negative signed distance to the mesh defined by the other point cloud. We compute a 2D metric, relating the surface areas of the meshes reconstructed from the point clouds, and a 3D metric, relating the volumes of their convex hulls, produced from uniformly sampled surface points.

V. RESULTS

Neural-field self-model. We show the predicted meshes from the 7-DOF robot self-model for five random test configurations from a fixed view in Fig. 3. It can be observed that in each configuration, the prediction follows the shape of the ground truth, subject to small deviations in the rotations of smaller parts of the body. For the shown samples, this indicates that the model learned to correctly approximate the shape from the configuration, despite the large space of possible configurations. We find that density scales with the certainty in the prediction and that despite the solid material of the robot, most of the non-zero density values are in the lower regime as opposed to close to one. Consequently, the threshold selection, that is the marching cube isolevel, is a significant hyperparameter since it controls the sensitivity with which sampled points are included. A too high value may exclude parts of the body in whose prediction the model is less certain. Due to the serial dependence among the DOFs - the first link's density only depends on the first joint value, whereas the seventh link's density depends on all previous joint values - those excluded parts are the upper parts of the body. The highest density values belong to points in the base of the robot, which remains static. In addition, the querying resolution determines the trade-off between computational cost and approximation of the true predicted model.

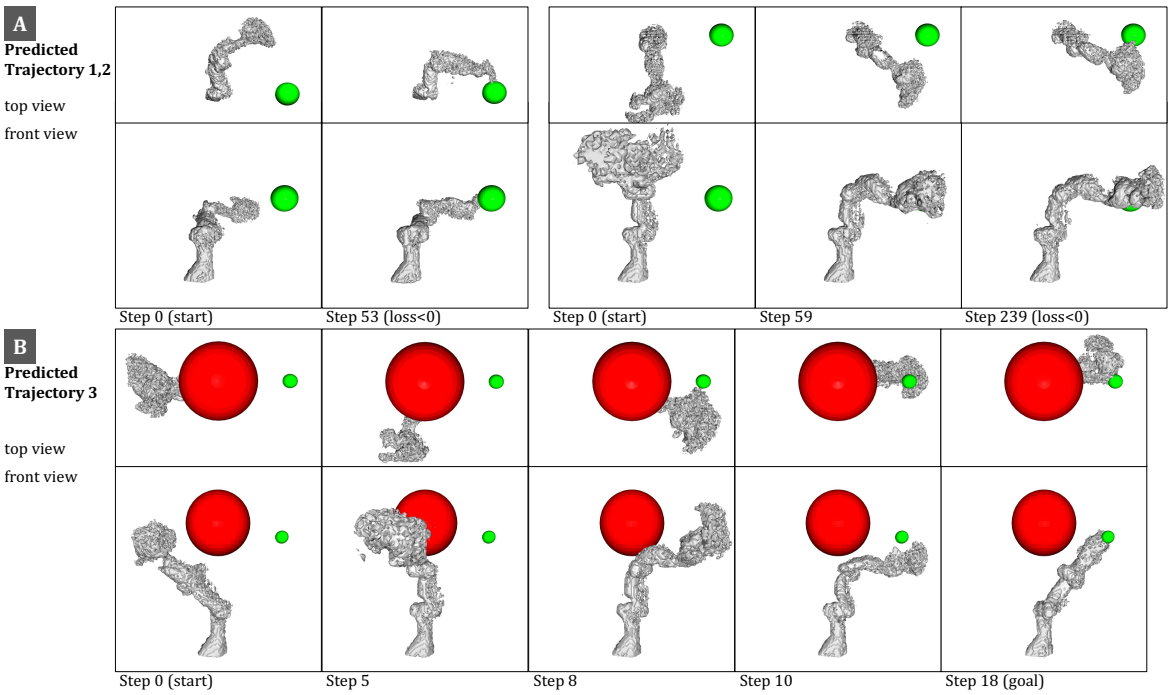

Fig. 4. **Motion planning results:** A) Joint value optimization via input PGD. A density loss is minimized when the sphere (green) is touched. B) RRT planning in config. space to intersect a point (green). The query model rejects samples with density on the obstacle (red). Please also see suppl. video [35].

In addition to the qualitative evaluation, numerical results on the spatial distance metrics are provided in Table I. As the most important metric, the mean of the Chamfer-L2 distance for the test set is 1.94% relative to the length of the shortest dimension of the workspace of the robot, 1.254m along the vertical axis. This indicates that, on average, each point on the mesh that was reconstructed from the points in the volume predicted to have sufficiently high density is close to a point on the robot's true surface given the queried configuration. Greater variance can be observed for the two IoU metrics. For the volume-based IoU, this is due to the constraint that the hull must convexly contain all points of the surface point cloud so that outliers have a large effect on its shape and, thus, volume. In addition, while marginally off-positioned predicted robot parts can still produce moderate Chamfer-L2 distances, these parts may not or only partially intersect with the ground-truth parts, leading to a lower value. The surface area is similarly outlier-sensitive and depends on the smoothness of the surface, which is not reliably given.

Motion planning. In Fig. 4, PGD- and RRT-generated trajectories are shown. The task for the former was to touch an object in an obstacle-free environment, while the task for the latter was to circumvent an obstacle to move from a start to a goal position. For PGD, the joint-limit-projected optimization results in valid trajectory steps. The robot moves itself into a configuration in which the sphere is touched such that the density on a surface point is above the threshold ($\tau = 0.6$). Unlike in classical kinematics, the part touching the target can be different from the end effector. However, distant start configurations may stop in local optima before reaching the target, rendering the learning rate a crucial hyperparameter. In addition, τ controls the closeness to the target in the final configuration. In those challenging cases, neural-field-based RRT reliably finds valid trajectories. In Fig. 4, the robot is able to move around the obstacle to reach its goal position. This approach is computationally more expensive since the neural field is queried extensively to construct the tree. Similarly, the strategy for sampling from the obstacle's volume impacts the performance and runtime.

VI. Conclusion and Outlook

We propose dynamic neural density fields conditioned on high DOFs. To this end, we introduce a hierarchical MLP architecture and curricular data sampling strategy. We use this method to learn the first neural-implicit self-model of a robot without depth or geometry information and from one camera, which can be used in lieu of a classical kinematic model. Future work may explore limiting the training data to more sparsely observed DOFs configurations and removing the need for camera parameter annotation via automatic estimation. In navigation and manipulation tasks, the integration of our approach, which models the robot, with previous work, which models the robot's environment, would be beneficial, as well as an extension to multi-robot setups. We highlight the usability of our method for dynamic object-centric scenes outside robotics in general DOFs-controlled environments.

VII. Acknowledgment

This work was supported in part by the US National Science Foundation (NSF) AI Institute for Dynamical Systems (DynamicsAI.org), grant 2112085.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision*, Springer, 2020, pp. 405–421.

[2] S. Lee, L. Chen, J. Wang, A. Liniger, S. Kumar, and F. Yu, "Uncertainty guided policy for active robotic 3d reconstruction using neural radiance fields," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 070–12 077, 2022.

[3] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606–4613, 2022.

[4] A. Moreau, N. Piasco, D. Tsishkou, B. Stanciulescu, and A. d. L. Fortelle, "Lens: Localization enhanced by nerf synthesis," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, 2022, pp. 1347–1356.

[5] D. Maggio, M. Abate, J. Shi, C. Mario, and L. Carlone, "Loc-nerf: Monte carlo localization using neural radiance fields," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 4018–4025.

[6] H. W. Stone, *Kinematic modeling, identification, and control of robotic manipulators*. Springer Science & Business Media, 1987, vol. 29.

[7] B. Chen, R. Kwiatkowski, C. Vondrick, and H. Lipson, "Fully body visual self-modeling of robot morphologies," *Science Robotics*, vol. 7, no. 68, eabn1944, 2022.

[8] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-nerf: Neural radiance fields for dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 318–10 327.

[9] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Nerfies: Deformable neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5865–5874.

[10] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt, "Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 959–12 970.

[11] S. Peng, J. Dong, Q. Wang, S. Zhang, Q. Shuai, X. Zhou, and H. Bao, "Animatable neural radiance fields for modeling dynamic human bodies," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 314–14 323.

[12] R. Kwiatkowski, *Deep Self-Modeling for Robotic Systems*. Columbia University, 2022.

[13] A. Dearden, "Developmental learning of internal models for robotics," Ph.D. dissertation, Imperial College London, 2008.

[14] P. Rochat, "Five levels of self-awareness as they unfold early in life," *Consciousness and cognition*, vol. 12, no. 4, pp. 717–731, 2003.

[15] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[16] J. C. Bongard and H. Lipson, "Automated damage diagnosis and recovery for remote robotics," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, IEEE, vol. 4, 2004, pp. 3545–3550.

[17] K. Gold and B. Scassellati, "Using probabilistic reasoning over time to self-recognize," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 384–392, 2009.

[18] R. Kwiatkowski and H. Lipson, "Task-agnostic self-modeling machines," *Science Robotics*, vol. 4, no. 26, eaau9354, 2019.

[19] K. Hang, W. G. Bircher, A. S. Morgan, and A. M. Dollar, "Manipulation for self-identification, and self-identification for better manipulation," *Science Robotics*, vol. 6, no. 54, eabe1321, 2021.

[20] Z. Jiang, W. Zhou, H. Li, Y. Mo, W. Ni, and Q. Huang, "A new kind of accurate calibration method for robotic kinematic parameters based on the extended kalman and particle filter algorithm," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 4, pp. 3337–3345, 2017.

[21] J. Bohg, K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme, "Interactive perception: Leveraging action in perception and perception in action," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1273–1291, 2017.

[22] B. Amos *et al.*, "Learning awareness models," in *International Conference on Learning Representations*, 2018.

[23] J. T. Kajiya and B. P. Von Herzen, "Ray tracing volume densities," *ACM SIGGRAPH computer graphics*, vol. 18, no. 3, pp. 165–174, 1984.

[24] N. Max, "Optical models for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, 1995.

[25] Z. Li, S. Niklaus, N. Snavely, and O. Wang, "Neural scene flow fields for space-time view synthesis of dynamic scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6498–6508.

[26] W. Xian, J.-B. Huang, J. Kopf, and C. Kim, "Space-time neural irradiance fields for free-viewpoint video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9421–9431.

[27] C. Gao, A. Saraf, J. Kopf, and J.-B. Huang, "Dynamic view synthesis from dynamic monocular video," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5712–5721.

[28] A. Noguchi, X. Sun, S. Lin, and T. Harada, "Neural articulated radiance field," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5762–5772.

[29] T. Li *et al.*, "Neural 3d video synthesis from multi-view video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5521–5531.

[30] H. A. Correia and J. H. Brito, "3d reconstruction of human bodies from single-view and multi-view images: A systematic review," *Computer Methods and Programs in Biomedicine*, p. 107 620, 2023.

[31] M. Sun, D. Yang, D. Kou, Y. Jiang, W. Shan, Z. Yan, and L. Zhang, "Human 3d avatar modeling with implicit neural representation: A brief survey," in *2022 14th International Conference on Signal Processing Systems (ICSPS)*, IEEE, 2022, pp. 818–827.

[32] L. Liu, M. Habermann, V. Rudnev, K. Sarkar, J. Gu, and C. Theobalt, "Neural actor: Neural free-view synthesis of human actors with pose control," *ACM transactions on graphics (TOG)*, vol. 40, no. 6, pp. 1–16, 2021.

[33] C.-Y. Weng, B. Curless, P. P. Srinivasan, J. T. Barron, and I. Kemelmacher-Shlizerman, "Humannerf: Free-viewpoint rendering of moving people from monocular video," in *Proceedings of the IEEE/CVF conference on computer vision and pattern Recognition*, 2022, pp. 16 210–16 220.

[34] R. Li, J. Tanke, M. Vo, M. Zollhöfer, J. Gall, A. Kanazawa, and C. Lassner, "Tava: Template-free animatable volumetric actors," in *European Conference on Computer Vision*, Springer, 2022, pp. 419–436.

[35] https://youtu.be/7kawCtVSpmA.

[36] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[37] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[38] E. Coumans, *Pybullet robots*, 2020. [Online]. Available: https://github.com/erwincoumans/pybullet_robots.

[39] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[40] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares, "Efficient implementation of marching cubes' cases with topological guarantees," *Journal of graphics tools*, vol. 8, no. 2, pp. 1–15, 2003.

[41] G. Taubin, "Curve and surface smoothing without shrinkage," in *Proceedings of IEEE international conference on computer vision*, IEEE, 1995, pp. 852–857.