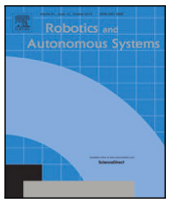
Robotics and Autonomous Systems 186 (2025) 104921

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



dGrasp: NeRF-Informed implicit grasp policies with supervised optimization slopes

Gergely Sóti [a,b,*], Xi Huang [b], Christian Wurll [a], Björn Hein [a,b]

[a] Karlsruhe University of Applied Sciences, Institute for Robotics and Autonomous Systems, Karlsruhe, 76133, Germany
[b] Karlsruhe Institute of Technology, Institute of Anthropomatics and Robotics, Karlsruhe, 76131, Germany

ARTICLE INFO

Keywords:
Implicit policy
Grasping
Implicit representation
Sim-to-real

ABSTRACT

We present dGrasp, an implicit grasp policy with an enhanced optimization landscape. This landscape is defined by a NeRF-informed grasp value function. The neural network representing this function is trained on simulated grasp demonstrations. During training, we use an auxiliary loss to guide not only the weight updates of this network but also the slope of the optimization landscape. This loss is computed on the demonstrated grasp trajectory and the gradients of the landscape. It requires second order optimization during training to incorporate valuable information from the trajectory and leads to facilitating the optimization process of the implicit policy. Experiments demonstrate that employing this auxiliary loss improves policies' performance in simulation as well as their zero-shot transfer to the real-world.

1. Introduction

Robotic grasping is a fundamental task in the automation of object manipulation. Despite extensive research and progress, dealing with unknown objects under real-world conditions is still a major challenge. In this context, learning from demonstration (LfD) has recently become an attractive alternative to reinforcement learning (RL) for policy learning due to its advantages in bypassing the need for a reward function and its higher sample efficiency. LfD does not require exploration or extensive data gathering but instead learns directly from high-quality demonstration data.

Within LfD, implicit behavior cloning (IBC, [1]) and diffusion policies [2] have emerged as effective methods. Both utilize optimization-based policies but with different underlying mechanisms. IBC learns an energy function over the joint action-observation distribution (Fig. 1(a)), which is minimized during inference to determine robot actions, effectively framing the policy as an optimization problem. This formulation allows the use of convenient action spaces like the 6-DoF task space of a robot, facilitating the incorporation of advanced scene representations such as Neural Radiance Fields (NeRFs, [3,4]), as demonstrated by Sóti et al. [5]. These integrations lead to improved generalization and enable zero-shot sim-to-real transfer of grasp policies. However, training IBC policies requires negative sampling for normalization, which can lead to instability, as noted by Florence et al. [1], Du et al. [6].

Diffusion policies address this instability by learning a gradient field and using a denoising process to generate robot actions (Fig. 1(b)). The gradient field is approximated directly by a noise prediction network, avoiding the need for normalization through negative sampling, offering a more stable alternative to IBC.

Our approach seeks to maintain the benefits of implicit policies while reducing the instability from negative sampling by also supervising the gradient field of the implicit policy during training (Fig. 1(c)). In action spaces such as TCP poses, the gradient of the energy function in implicit policies should naturally align with the robot's movement. By extending the IBC framework to align its gradients with demonstrated robot movements, we ensure that the gradients reflect the pose changes seen in successful demonstrations. The primary goal of this work is to show that this sort of alignment of the model to the physical world in combination with NeRF based representations leads to significantly improved convergence behavior and sim-to-real transfer of implicit policies.

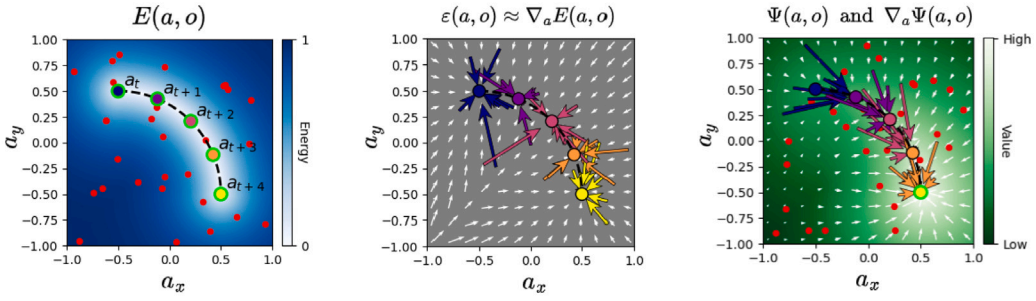
NeRFs offer foundational properties for this method as they represent the environment as a continuous volumetric field and map poses to specific properties, aligning seamlessly with the IBC framework. By incorporating principles from computer graphics, rendering, and camera modeling, NeRFs offer a strong inductive bias that captures rich geometric details. This bridges perception and the robot's tasks space, providing a differentiable framework for learning and optimizing manipulation-related properties directly, making NeRFs a powerful tool for robotic grasping and other manipulation tasks.

We summarize our main contributions as follows:

* Corresponding author at: Karlsruhe University of Applied Sciences, Institute for Robotics and Autonomous Systems, Karlsruhe, 76133, Germany.
E-mail address: gergely.soti@h-ka.de (G. Sóti).

https://doi.org/10.1016/j.robot.2025.104921

Available online 18 January 2025



(a) Implicit Behavior Cloning (b) Diffusion Policy (c) Our Approach

Fig. 1. Policy Representations - Comparison of policy representations for observation o and a demonstration trajectory $\{a_t, a_{t+1}, \dots, a_{t+4}\}$ in a two-dimensional action space. (a) *Implicit Behavior Cloning (IBC)* learns an energy function $E(a, o)$ using negative sampling; (b) *Diffusion Policy* learns a noise function $\varepsilon(a, o)$ that approximates the gradient field of the energy function $\nabla_a E(a, o)$; (c) In *our approach* we learn a value function using negative sampling $\Psi(a, o)$ additionally supervising its gradients $\nabla_a \Psi(a, o)$ using the demonstration trajectories during training. This formulation combines the convenient representation of IBC and the stability and robustness of diffusion policy.

- We propose a natural way to incorporate demonstration trajectories into the training of implicit policies.
- We introduce a simple augmentation and training mechanism to supervise the gradients of implicit policies using the trajectories.
- We apply this method in the NeRF-based implicit policy by Sóti et al. [5] and demonstrate its effectiveness on simulated and real grasping tasks.

The primary focus of this work is to evaluate the impact of the proposed auxiliary loss on NeRF-based implicit grasp policies. While comparisons with additional baseline models are valuable, they fall outside the scope of this study. State-of-the-art approaches such as diffusion policies represent a fundamentally different paradigm (e.g., closed-loop control) and require a separate, rigorous comparison. This work aims to establish the foundational benefits of the proposed method, paving the way for more comprehensive comparisons in future research.

The remainder of this paper is structured as follows: Related work is discussed in Section 2, and a summary of the background information required for a complete description of our approach is provided in Section 3. This is followed by a detailed description of our method in Section 4, and the conducted experiments and results in Section 5. The paper concludes with a brief discussion of the findings, the limitations of our approach, and possible ideas for future research in Section 6.

2. Related work

Data-driven policies in the context of robotic grasping are widely researched topic with a variety of approaches as detailed in the surveys by Bohg et al. [7], Kleeberger et al. [8], Newbury et al. [9]. Broadly, these methods fall into four categories: (i) object-detection based [10–12], (ii) reinforcement learning [13–15], (iii) supervised learning from a large-scale, labeled dataset [16–18], and (iv) learning from demonstration [1,2,19] .

In this work, we focus on behavior cloning, a popular end-to-end framework to learn policies from demonstrations, and even in behavior cloning there are two emerging branches, i.e. explicit and implicit models. Explicit models, like the works of Avigal et al. [20], Florence et al. [21], Rahmatizadeh et al. [22], Zeng et al. [19], propose actions directly from observations. Implicit models on the other hand learn to evaluate actions and are used in conjunction with sampling based or gradient-based optimization to find optimal actions [1,5]. In this paper, we aim to extend the training process of such implicit models in a way that facilitates the optimization process, and thus improves the policy. In the following we review related work in the context of implicit policies and finally we discuss the core idea of our proposed approach.

Florence et al. [1] investigate the effects of using implicit models for behavior cloning (IBC) across a variety of robot policy learning tasks. They define an implicit policy as the argmin of a continuous energy function, which is learned from demonstrations. This energy function is expected to assign lower energies to optimal actions like the demonstrations, and higher energies otherwise. To find minimum locations and thus optimal actions, they propose two sampling-based algorithms and a gradient-based algorithm. Their research shows that implicit models provide competitive results or outperform explicit models and reinforcement learning algorithms on complex, discontinuous and multi-modal simulated and even real robotic tasks.

With a similar framework, Sóti et al. [5] learns 3 degree-of-freedom (DoF) grasps in simulation and applies zero-shot transfer to the real world. Unlike IBC, which minimizes an energy function, these methods maximize a value function and use a gradient-based optimization with the Adam optimizer [23] instead of the gradient-based Langevin sampling described in IBC. An additional key characteristic of the approach is the usage of a pre-trained Neural Radiance Field (NeRF) [3,4] to inform the implicit model and thus requiring only RGB observations during inference and enabling a large degree of generalization.

NeRFs themselves learn an implicit representation of the environment, and have been used in various works involving robotic grasping [24–27]. However, these applications primarily leverage NeRFs for augmenting observations or as feature extractors for explicit policies, differing from the implicit optimization-based framework we use.

Although Neural Motion Fields, by Chen et al. [18], do not employ learning from demonstration, they also use an implicit model to learn a grasp value function and to generate grasp trajectories in an object-centric way. In this approach, training the grasp value function requires a curated set of ground truth grasp poses and their model requires a segmented point cloud as input. Trajectories are generated by optimizing the learned implicit value function via sampling-based model predictive control.

Related to implicit models, Weng et al. [28] approach grasp synthesis by predicting the distance of an action candidate to the nearest successful grasp and minimize this distance to achieve successful grasps. By integrating this distance metric into CHOMP motion planning as an additional cost, the model can generate grasp trajectories. For training, this method also requires a large grasp dataset [17] and the model processes a segmented point cloud as input.

Training the energy based model for implicit polices often involves a negative sampling process, which is known to cause training instability [1,6]. To avoid this, diffusion policies by Chi et al. [2], use a denoising neural network that captures the gradient of the reversed diffusion process. They train their model by minimizing the difference of a diffused action from ground truth and a synthetic denoised action. Given a sequence of observations, the policy uses the gradient field iteratively to denoise a sequence of randomly sampled actions and finally execute them. While the diffusion policy learns the gradient field to update action sequences from noisy ones, the landscape of implicit models captures the slope from an arbitrary state towards an optimal

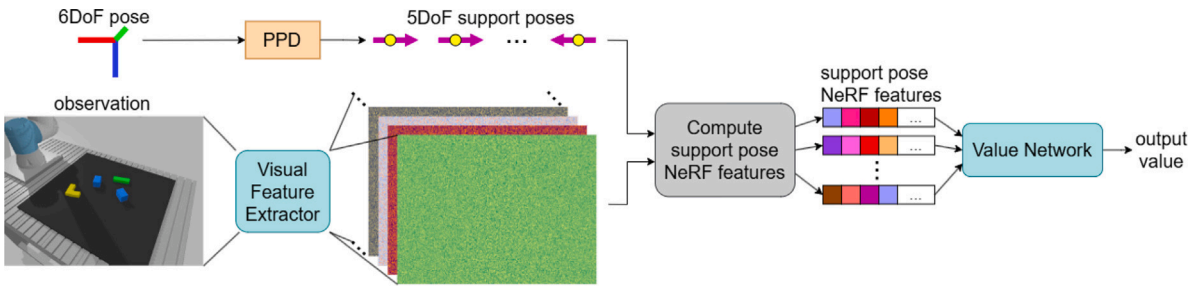


Fig. 2. Computational model for the implicit policy's value function Ψ for a 6-DoF grasp candidate and an observation - First, partial pose decomposition (PPD) is applied to the 6-DoF grasp candidate to obtain a set of 5-DoF support poses, and a feature map is computed from the input observation. Then, for each 5-DoF support pose, a NeRF feature vector is computed using the extracted feature map and a pre-trained NeRF. These are finally processed by the value network to obtain the grasp value for the input 6-DoF grasp candidate.

action. This means, that the optimization process of an implicit policy itself results in an action sequence.

In this current approach, our goal is to improve the optimization landscape of the implicit grasp policy described in Sóti et al. [5]. In addition to learning from the grasp pose itself, we use second-order optimization to supervise the gradients of the optimization landscape using the demonstrated grasp trajectory. Although, using a second-order optimization algorithm like BFGS algorithm [29] might prove beneficial for training, such algorithms usually involve the computation of the Hessian and often its inverse, too, which can be expensive. Instead, we use the Adam optimizer for both training and to lead the pose along the landscape during optimization, consistent with [5,30,31].

3. Background

We build on a previous work that introduces the concept of using transfer learning from Neural Radiance Fields (NeRFs, [3]) to train a grasp value function that can be used in an implicit policy [5]. In this section we briefly introduce NeRFs and describe the architecture used by Sóti et al. [5] to compute the grasp value, the way it is trained and used in an implicit policy framework to infer grasp poses.

3.1. Neural radiance field - NeRF

NeRFs, Mildenhall et al. [3] have revolutionized scene representation by learning implicit 3D structures from 2D images. Originally developed for novel view synthesis, NeRFs combine traditional rendering methods with deep learning to achieve impressive performance. They map positions and direction vectors, representing the 5-DoF space, to color and density values, which are then used in a volume rendering pipeline to render pixels for a camera image via raycasting. The key innovation of NeRFs lies in their ability to accurately synthesize new views of a scene by learning a continuous volumetric scene function from a sparse set of 2D images. This scene function acts as a powerful scene representation, capturing the detailed geometry and appearance of the environment.

By embedding the geometry of the scene into a continuous volumetric representation, NeRFs allow for efficient and natural learning of physical structure, which is directly relevant to grasping and other manipulation tasks. Raycasting introduces inductive biases that reflect how cameras perceive scenes, thereby simplifying learning by eliminating the need to train for features that are inherently understood through volume rendering. Nevertheless, we recognize that NeRFs come with notable drawbacks, including high computational costs and potential limitations in scaling to larger or more complex task spaces. This work aims to highlight the potential of implicit representations, such as NeRFs, rather than position it as the definitive solution for robotic manipulation tasks.

3.2. Transfer learning with NeRFs for grasp value function

Leveraging the scene representation provided by NeRFs, Sóti et al. [5] introduce a method that applies transfer learning to an image-conditioned NeRF variant [4], using it to inform a grasp value function within an implicit policy framework. The grasp value function, denoted as Ψ, maps 6-DoF Tool Center Point (TCP) poses p and observations o to scalar values, with higher values indicating a greater likelihood of a successful grasp. The grasp policy, π, is formulated as follows:

$$\pi(o) = \text{argmax}_p \Psi(p, o) \tag{1}$$

Here, $\pi(o)$ represents an estimated optimal TCP pose for grasping. To find these maximum locations of Ψ, gradient-based optimization is employed.

3.2.1. Architecture

Ψ itself consists of four modules: partial pose decomposition (PPD), a visual feature extractor, a module to compute NeRF features using a pre-trained image-conditioned NeRF, and a value network. Fig. 2 illustrates their interaction to compute the value of a 6-DoF grasp candidate given an observation.

Partial Pose Decomposition (PPD) - NeRFs process 5-DoF poses to obtain color and density values for novel view synthesis. To evaluate 6-DoF grasp candidates, PPD is applied. This computes a set of predefined 5-DoF support poses from a 6-DoF pose, which can be processed independently by the NeRF and aggregated later to characterize the initial 6-DoF pose. Fig. 3 shows a possible PPD for grasping that corresponds to the geometry of the gripper.

Image-Conditioned NeRF - An image-conditioned NeRF (VisionNeRF by Lin et al. [4]) is used to compute feature vectors for each support pose. In this context, image conditioning means that during training and inference, a visual feature extractor processes input observations and informs the NeRF of the current scene. This results in a NeRF that can be used in multiple environments without retraining based on a set of observations, ensuring consistent representation across different scenes.

Visual Feature Extractor and Support Pose NeRF Features - The visual feature extractor combines a pre-trained vision transformer with fully convolutional neural networks as described by Lin et al. [4] to compute features from input observations to inform the NeRF. The extracted features correspond to the same perspective that input was provided from. Using the camera's calibration information, the 3D point of a support pose is projected onto the extracted feature maps to obtain visual feature vectors, as shown in Fig. 4 . The support poses, along with their corresponding visual feature vectors, are processed by the NeRF. A positional encoding typical for NeRFs is applied to both the 3D point and the direction vectors of the support pose:

$$\gamma(v) = (sin(2^0\pi v), cos(2^0\pi v), \dots, sin(2^{M-1}\pi v), cos(2^{M-1}\pi v)) \tag{2}$$

The function γ is applied to each vector dimension separately, and the results are concatenated. The NeRF itself contains six ResNet blocks (see

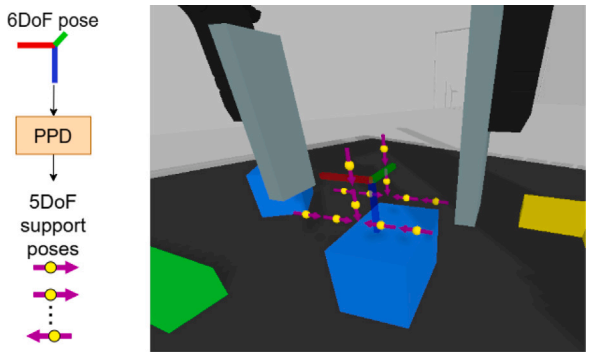


Fig. 3. Partial pose decomposition - A set of 5-DoF support poses are computed from an initial 6-DoF pose using predefined transformations. The image shows the TCP as a 6-DoF pose and a possible set of its support poses that correspond to the gripper's geometry: the yellow points with purple direction vectors pointing inwards to characterize possible object boundaries that the gripper could grasp.

Fig. 5(a) for ResNet architecture) and is pre-trained with randomized scenes for novel view synthesis. To inform the value network, the NeRF's activations after the last four ResNet blocks are used. These are aggregated into the NeRF feature vector characterizing the support pose. Fig. 4 shows the architecture for computing the NeRF features for a support pose given the extracted visual features from the observation.

Value Function - Finally, the support pose NeRF features are processed by the value function to obtain the value for the implicit policy. Fig. 5(b) shows the network architecture representing the value function.

3.2.2. Training

The implicit policy's value function Ψ is trained using demonstrations by transforming the learning process into a binary classification problem. Each demonstrated grasp is labeled with 1, and randomly sampled poses within the workspace serve as negative examples labeled with 0. The input observations o are camera images of the scene with known calibration before executing the grasp. With this setup, the categorical cross-entropy loss function is used during training:

$$\mathcal{L}_{value} = -\log \frac{e^{\Psi(p^0,o)}}{\sum_{i=0}^{N} e^{\Psi(p^i,o)}} \tag{3}$$

Here, p^0 is the successful demonstration and p^i (with $i \in [1, N]$) are the sampled negative examples. This way, the model is trained to assign higher scores to demonstrated poses compared to other poses within the workspace.

3.2.3. Optimization process

The optimization process adapts a set of randomly sampled initial input poses to maximize the output value of Ψ, as outlined in Algorithm 1. A pose consists of a 3D position vector and the quaternion representation of its orientation. First, the translations are optimized for 16 iterations, then the quaternions for 16 iterations, both adapted via the Adam optimizer [23]. After each optimization step, the quaternions are normalized.

When comparing NeRFs to other potential models for robotic manipulation, the key requirement is the ability to map poses to certain values relevant for task execution. In previous work [5], pretraining NeRFs for novel view synthesis was shown to provide significant advantages for manipulation tasks. Testing the same NeRF-based architecture with and without this pretraining revealed substantially better performance in the pretrained model. This demonstrates the value of leveraging learned geometric representations, reinforcing NeRFs as a strong foundational model for downstream robotic applications.

Algorithm 1 Grasp pose optimization ([5])

Require: Observation o
Ensure: Successful grasp p^*
$G \leftarrow$ RandomGraspCandidates() ▷ Initialization
while Not Terminate **do** ▷ Termination criterion
  for all $p \in G$ **do**
    $p \leftarrow p + \nabla_p \Psi(p, o)$ ▷ Maximize Ψ
    $p \leftarrow$ PostProcess(p) ▷ Fix pose
  end for
end while
$p^* \leftarrow \arg\max_{p \in G} \Psi(p, o)$ ▷ Grasp with highest value
return p^*

In the following section, we describe our approach to improve the learning of Ψ to enhance the optimization results.

4. Method

The approach in Sóti et al. [5], as detailed in Section 3, focuses on using demonstrated grasp poses to learn the grasp value function Ψ. However, it does not utilize the demonstrated trajectories, which encompass the entire Tool Center Point (TCP) movement.

We aim to incorporate the demonstrated grasp trajectories into the learning Ψ by proposing an augmented loss function that includes an auxiliary loss term. This term aligns the gradients of Ψ with demonstrated TCP movements, ensuring the grasp value function considers the entire TCP trajectory. The intuition behind this is that these gradients should ideally reflect the actual TCP movements observed in successful demonstrations. We hypothesize that this augmented loss improves the grasp value function's alignment with the physical world, leading to better optimization and improved grasp outcomes.

In the following, we detail our proposed enhancements, including the formulation of the auxiliary loss, architectural modifications, and implementation details.

4.1. Auxiliary loss

In Sóti et al. [5] the optimization landscape for pose optimization is shaped by the value loss $\mathcal{L}_{value}$ (Eq. (3)), which only considers the executed grasp pose. To include the grasp trajectories, we augment the grasp value function Ψ to have the following property:

$$p_{t+1} = p_t + \nabla_p \Psi(p_t, o) \tag{4}$$

with p_t as the TCP pose at timestep t and p_{t+1} as the pose at a later timestep $t+1$ during a demonstration. On one hand, this aligns with the gradient-based optimization of the grasp candidates poses (see Algorithm 1). On the other hand, given that we have access to ground truth trajectories from demonstrations, this gradient can be supervised by the displacement of the TCP pose along the trajectory during training as an auxiliary loss:

$$\mathcal{L}_{aux} = -S_C(p_{t+1} \ominus p_t, \nabla_p \Psi(p_t, o)) \tag{5}$$

with S_C the cosine similarity and $p_{t+1} \ominus p_t$ representing the element-wise difference of the pose representations. We believe this straightforward operator is effective in a gradient-based context because small gradient steps allow linear changes in the representations to be sufficient. The rationale behind using cosine similarity is that we are primarily interested in the direction of the gradients rather than their magnitude and leverage the Adam optimizer's capabilities for more stable updates.

Including the auxiliary loss, the total loss for the model:

$$\mathcal{L}_{total} = \mathcal{L}_{value} + \mathcal{L}_{aux} \tag{6}$$

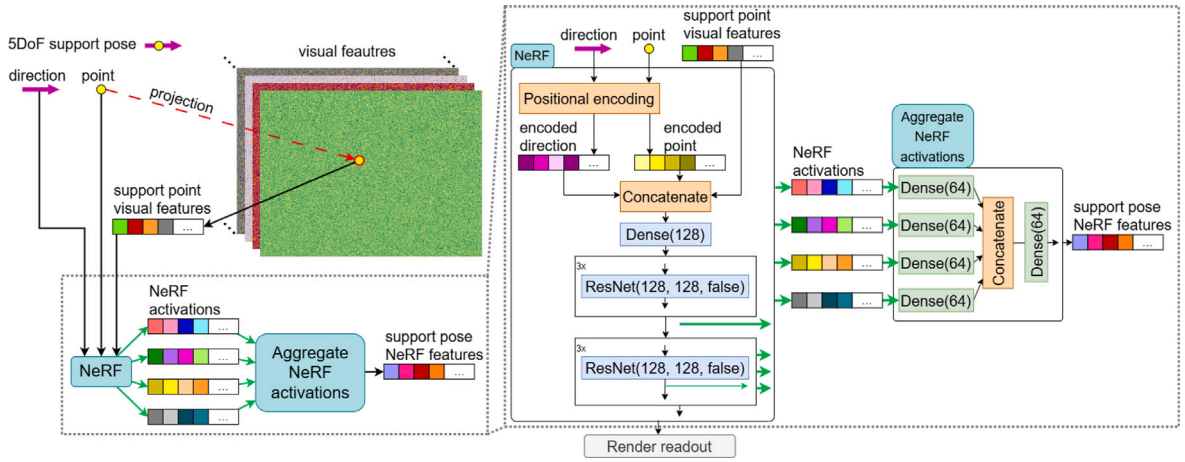


Fig. 4. Computation of support pose NeRF features - Left: Computation of NeRF features for a 5-DoF support pose using its corresponding visual feature vector; Right: network architecture of the NeRF and activation aggregation models. The green arrows represent the activations of the NeRF's last four ResNet blocks that are aggregated to form the NeRF feature corresponding to the input support pose. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

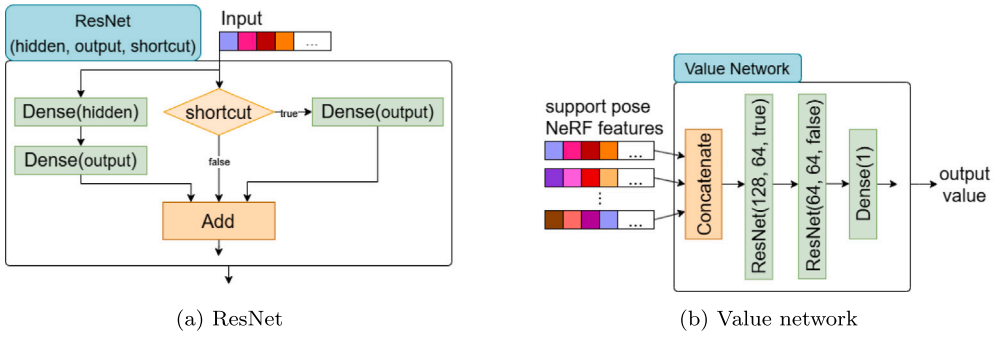


Fig. 5. Network architectures - Value Network: The network processes the support pose NeRF features to compute the final grasp value for the implicit policy; ResNet: Used in NeRF and Value Networks. Transforms the residual shortcut if the input and output dimensions are not equal.

4.2. Pose representation

The optimization process relies on the Adam optimizer to adapt the input poses of Ψ in such a way that its output is maximized. This gradient-based pose optimization makes the selection of pose representation crucial.

We split the pose into a position vector and an orientation representation. While [5] use quaternions for orientation, we implement the 6D orientation representation proposed by Zhou et al. [32], which is also utilized by Chi et al. [2] for diffusion policies. For both orientation representations, it is essential to ensure they remain valid after the gradient-based update. Quaternions are normalized and for 6D representations, consisting of the first two column vectors of the rotation matrix, the vectors are orthonormalized.

To compute $\mathcal{L}_{aux}$ Eq. (5), we process the cosine similarity of the gradient and $p_{t+1} \ominus p_t$ for the position and orientation representations independently, and then sum them. Additionally, in case of the 6D orientation representation, the two column vectors are handled independently.

It is worth noting that without the auxiliary loss, we can freely interchange pose representations, as they are only used during the optimization process and not during training. However, with the auxiliary loss, the gradient used in Eq. (5) depends on the pose representation, requiring a new model for each representation.

4.3. Architecture

Since our goal is to incorporate the gradient of Ψ into the loss function and training, we need to consider the gradients of the newly added error function $\mathcal{L}_{aux}$ with respect to the trainable weights θ of Ψ. When training Ψ, a pre-trained NeRF Φ is used with frozen weights, thus only the weights of NeRF activations aggregation network (Fig. 4) and the value network (Fig. 5(b)) are trained and belong to θ.

To compute the gradients for the weight update from $\mathcal{L}_{aux}$ we use the gradients of the grasp vale model itself, $\nabla_p \Psi(p_t, o) = \frac{\partial \Psi}{\partial p}$, which also involves evaluating the NeRF Φ:

$$\frac{\partial \mathcal{L}_{aux}}{\partial \theta} = \frac{\partial \mathcal{L}_{aux}}{\partial \frac{\partial \Psi}{\partial p}} \frac{\partial \frac{\partial \Psi}{\partial p}}{\partial \theta} = \frac{\partial \mathcal{L}_{aux}}{\partial \frac{\partial \Psi}{\partial p}} \frac{\partial (\frac{\partial \Psi}{\partial \Phi} \frac{\partial \Phi}{\partial p})}{\partial \theta} \tag{7}$$

Since Φ is independent of θ resolving the partial differential $\frac{\partial (\frac{\partial \Psi}{\partial \Phi} \frac{\partial \Phi}{\partial p})}{\partial \theta}$ results in:

$$\frac{\partial (\frac{\partial \Psi}{\partial \Phi} \frac{\partial \Phi}{\partial p})}{\partial \theta} = \frac{\partial \frac{\partial \Psi}{\partial \Phi}}{\partial \theta} \frac{\partial \Phi}{\partial p} + 0 \tag{8}$$

This makes:

$$\frac{\partial \mathcal{L}_{aux}}{\partial \theta} = \frac{\partial \mathcal{L}_{aux}}{\partial \frac{\partial \Psi}{\partial p}} \frac{\partial^2 \Psi}{\partial \Phi \partial \theta} \frac{\partial \Phi}{\partial p} \tag{9}$$

The expression shows, that Ψ has a mixed partial derivative with respect to its weights θ and the activations of Φ. This means, that discontinuities in its first derivative should to be avoided, otherwise its second derivative could destabilize the weight update process. Since we are in the context of neural networks, we only have to make sure, that the derivatives of the employed activation functions are continuous. In order to be able to use $\mathcal{L}_{aux}$, we replace the ReLU activation functions of the NeRF activations aggregation network (Fig. 4) and the value network (Fig. 5(b)) with ELU. As for the NeRF Φ, it is sufficient to be

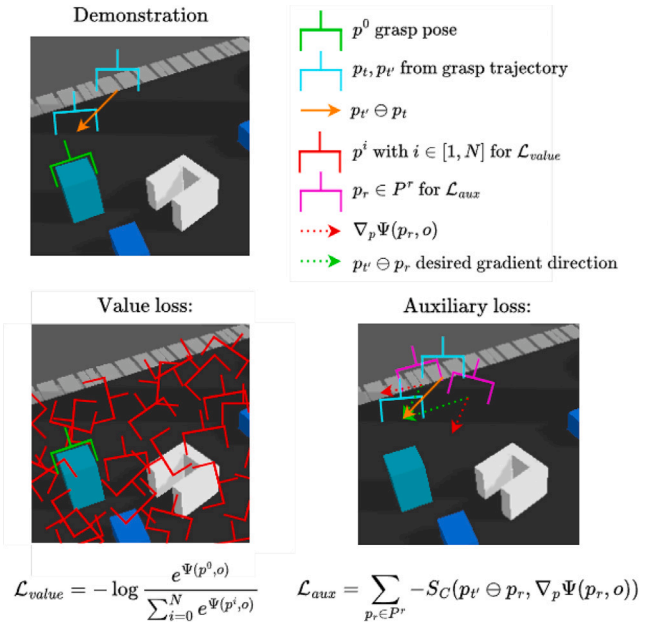


Fig. 6. Loss functions - The value loss contributes to selecting the correct grasp (green) from the randomly generated ones (red). The auxiliary loss encourages the gradients of the value function to align with the demonstrated TCP movement. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

differentiable with respect to the input TCP pose p. This means that we still can use the pre-trained NeRF as it is.

4.4. Training

For training, both the value loss $\mathcal{L}_{value}$ and the auxiliary loss $\mathcal{L}_{aux}$ are used. For $\mathcal{L}_{value}$, the demonstrated grasp pose is augmented with negative samples as described by Sóti et al. [5].

We also augment the data from the grasp trajectory to compute the auxiliary loss. For a given observation o and current and future TCP poses p_t and p_{t+1}, we sample a set of poses P^r in the proximity of p_t. The loss is then computed on $p_{t+1} \ominus p_r$ and $\nabla_p \Psi(\Phi(p_r, o))$ for each sampled pose $p_r \in P^r$:

$$\mathcal{L}_{aux} = \sum_{p_r \in P^r} -S_C(p_{t+1} \ominus p_r, \nabla_p \Psi(p_r, o)) \tag{10}$$

We rationalize this decision with when a pose is near p_t then moving towards p_{t+1} should still lead towards successful grasps. Fig. 6 visualizes the differences between $\mathcal{L}_{value}$ and $\mathcal{L}_{aux}$: $\mathcal{L}_{value}$ aims at learning to identify good grasps, and $\mathcal{L}_{aux}$ learns how to get there.

4.5. Optimization process

The improved optimization landscape resulting from applying the auxiliary loss $\mathcal{L}_{aux}$ allows us to implement a synchronous optimization process. Instead of optimizing positions and rotations sequentially like [5], we can now optimize both simultaneously.

To tune the optimization process we employ Bayesian hyperparameter optimization over 100 iterations. Hyperparameters include the optimizer's initial learning rates for the pose representations, their corresponding decay rates, and the number of optimization steps.

5. Experiments and results

To evaluate the grasp policies, we conduct a series of experiments in both simulated and real-world environments. These tasks are designed to test the models' ability to generalize across familiar (in-distribution) and unfamiliar (out-of-distribution) scenarios and their adaptability to real-world conditions.

5.1. Tasks

We use the three simulated and a real-world task described in Sóti et al. [5] to measure the grasp success rate of a policy for testing:

- Tasks in a pybullet simulated environment. Grasping is successful if an object is enveloped by the gripper fingers and was lifted up after the physics-based grasp execution
  - **simple:** This task assesses basic grasping capabilities. The workspace contains up to five monochromatic prismatic objects, each placed at a distance from the others (Fig. 7(a)). The goal is to successfully grasp one of these objects.
  - **clutter:** This task tests the model's performance on out-of-distribution pose data. The scenario is a cluttered workspace containing five monochromatic prismatic objects (Fig. 7(b)). The objective is to grasp all objects one after the other.
  - **novel objects:** This task assesses the model's ability to handle out-of-distribution pose, shape, and texture data. The workspace features one previously unseen object selected from the YCB dataset [33] (Fig. 7(c)). The goal is to grasp this object. Objects used: banana, foam brick, gelatin box, hammer, Master Chef can, pear, power drill, strawberry and tennis ball.
- **real-world:** This task tests the transferability of the model to the real world. In this task, a single everyday object is randomly placed in the workspace of an real robot (Fig. 7(d)). The task is considered successful if the robot can securely grasp and lift the object. Objects used: crochet ball, a shuffled Rubik's cube, large Lego tire, canned tomato, rubber duck, hiking boot, dental floss, power drill, shampoo bottle and a 3D printed block.

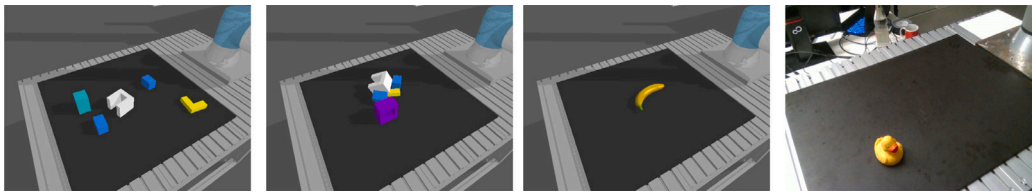
All tasks feature a UR10 robot on a workbench, equipped with a Robotiq 2f-140 gripper and an Intel RealSense D415 camera. Examples of the tasks are shown in Fig. 7.

5.2. Training and hyperparameter tuning

During training, all models are exclusively exposed to the training dataset of the simulated **simple** scenario, including the pre-training of the NeRF and the grasp value models. This means, that neither of the models have ever seen objects in close proximity to each other, with complex textures and shapes or in poses other than upright during training.

In each of the grasp value models we use the same pre-trained NeRF, which was trained on 2.5k randomly set up **simple** tasks with 50 randomly sampled camera perspectives for 1600 epochs as described in Sóti et al. [5]. The training data for the grasp value models consists of 512 demonstrated grasps, generated in simulation, containing RGB observations and the grasp trajectory. Successful grasp poses are determined by an oracle with access to the simulation's state. The observations are recorded from 16 randomly sampled perspectives pointing to the center of the robot's workspace before the action execution. The grasp models were trained for 400 epochs. We configure both the NeRF and grasp value models to process a single image from a single perspective at the time.

The grasp value models can be used in combination with differently configured policies using sequential or synchronous position



(a) simple (b) clutter (c) novel objects (d) real-world

Fig. 7. Example observations from the test datasets.

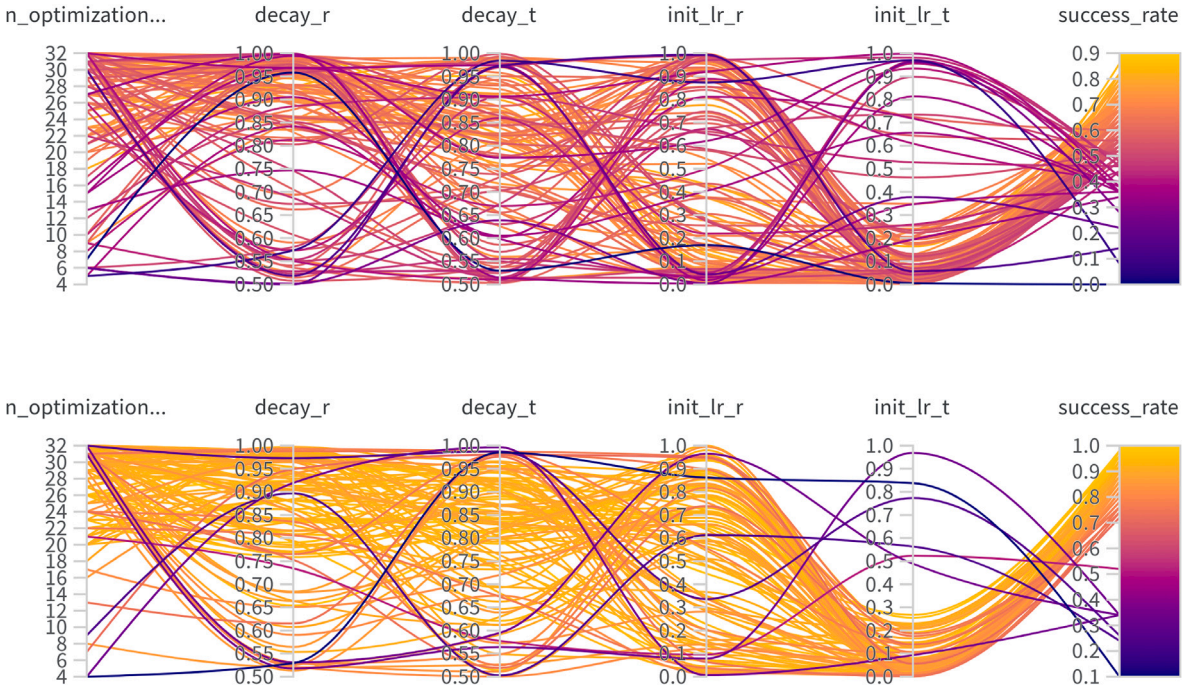


Fig. 8. Bayesian hyperparameter optimization - The image shows the configurations tested during hyperparameter optimization of a policy using a model trained only with the value loss $\mathcal{L}_{value}$ (top) and for a policy also employing the auxiliary loss $\mathcal{L}_{aux}$ (bottom).

and orientation optimization or in case of using the value loss $\mathcal{L}_{value}$ only, the rotation representation can also be altered. After training the grasp value models we run Bayesian hyperparameter optimization to tune each policy using them. This narrows down the required number of pose optimization steps and the learning and decay rates for the pose optimization in order to improve success rates. Success rates are determined on a validation dataset of the **simple** task with simulated grasp execution. The hyperparameters tuned are described in 4.5. Fig. 8 shows the configurations tested during tuning policies with and without employing the auxiliary loss $\mathcal{L}_{aux}$. The overall higher success rates when using $\mathcal{L}_{aux}$ indicate that the learned grasp value models are more suitable for optimization, more robust for optimization parameter perturbations and even lead to higher success rates.

5.3. Results

Due to the different policy and model configurations (using $\mathcal{L}_{aux}$, rotation representation, sequential or synchronous pose optimization) and policy parameters, we trained, tuned and tested several policy and model combinations.

During our experiments we found that when not using $\mathcal{L}_{aux}$, sequential optimization in the policy leads to higher success rates, and that 6D rotation representation tends to outperform quaternions which aligns with the findings of Zhou et al. [32]. When using $\mathcal{L}_{aux}$ however, quaternions perform better.

Our baseline policy $\texttt{Base}_{\texttt{quat}}$ does not use $\mathcal{L}_{aux}$ and uses quaternions as rotation representation with sequential optimization, aligning with the description in Sóti et al. [5]. Additionally, we include $\texttt{Base}_{\texttt{6d}}$ as a second baseline, with employing 6D rotation representation as the only difference.

While this work does not include comparisons with additional baseline models, the focus here is on demonstrating the efficacy of the proposed auxiliary loss in improving the optimization landscape for NeRF-based policies. Specifically, we aim to evaluate how this auxiliary loss influences performance, particularly in the context of learning from very simple demonstrations and generalizing to the real world without explicitly addressing the domain gap. Comparing our approach with state-of-the-art methods would require significant effort to adapt these methods, as they often operate under different paradigms. For example, current state-of-the-art methods are primarily closed-loop systems, while our approach is open-loop. Additionally, most state-of-the-art methods rely on in-domain data, such as real-world data for real-world deployment, whereas our focus is on generalization, including sim-to-real. Given these differences, a direct comparison in the current study is outside the scope of this paper. Future work will address this limitation by reducing the differences in these paradigms and conducting a rigorous comparison under more comparable conditions.

In the following we present results of the best performing policy using $\mathcal{L}_{aux}$ with sequential optimization $\texttt{dGrasp}$ and also with synchronous optimization $\texttt{dGrasp}_{\texttt{sync}}$. Both use quaternions as rotation representation. For all models during policy inference, we use 3 input

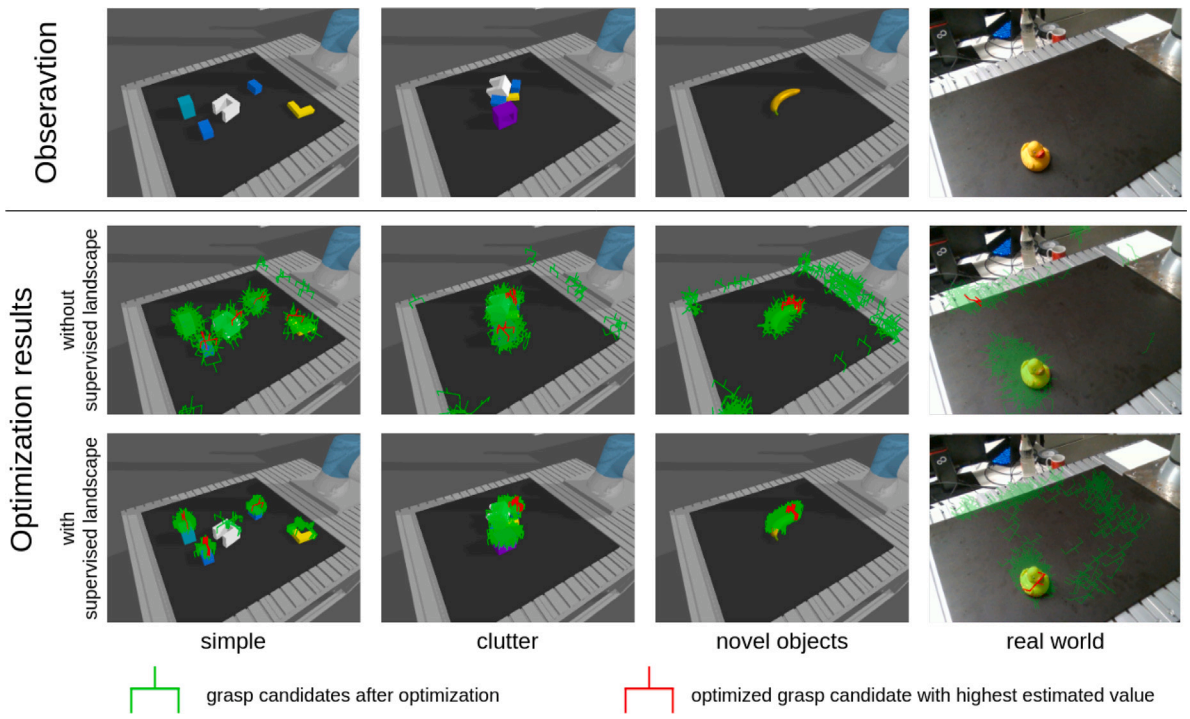


Fig. 9. Grasp pose optimization results - The optimization process adapts randomly sampled grasp candidates to improve their grasp value and finally the pose with the highest value is selected for execution. The figure shows the final state of the poses (green) on different tasks. The baseline policy in shown in the middle row and the policy with a supervised optimization landscape is shown in the bottom row. The best predicted grasp candidates are highlighted in red. Using the auxiliary loss improves pose-to-object alignment and real-world performance. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Grasp success rates - Mean and standard deviation for simulated tasks and success rate for the real task.

	Simple	Clutter	ycb	Real
`Base`	0.77 ± 0.04	0.66 ± 0.03	0.62 ± 0.04	0.28
$\texttt{Base}_{\texttt{6d}}$	0.79 ± 0.02	0.68 ± 0.03	**0.64** ± 0.03	0.30
`dGrasp`	**0.91** ± 0.02	**0.70** ± 0.04	0.62 ± 0.04	0.56
$\texttt{dGrasp}_{\texttt{sync}}$	**0.91** ± 0.02	0.67 ± 0.01	0.60 ± 0.02	**0.60**

images that are processed independently by the value function but their sum is used as an objective function during optimization, as described in Sóti et al. [5].

The simple and ycb test datasets contain 100 different scenes where the robot is allowed to execute a single grasp. The clutter test dataset contains 20 different scenes with 5 objects in a clutter and the robot is allowed to attempt to remove an object 10 times. Grasp success is determined by execution in simulation and we repeat each simulated experiment 6 times. In the real task, each of the 10 objects is placed 5 times randomly in the robots workspace with the robot executing a single grasp every time. Table 1 presents the average grasp success rates and their standard deviation in the simulated task and the success rate in the real task for the policies. An example of the final results of a policy's pose optimization for models without and with $\mathcal{L}_{aux}$ in Fig. 9 shows the improved convergence properties when supervising the optimization landscape during training, especially in simulated environments.

There is no significant difference between model performance on the clutter and ycb tasks, however an analysis of variance (ANOVA) on the success rates yielded significant variation among the models ($F(3, 20) = 41.03, p < .001$) in case of the simple task. A post hoc Tukey test revealed that both the `trajectory-quat` model and the `trajectory-quat-sync` model had significantly higher success rates compared to the `goal-quat` and the `goal-6d` models. This shows the effectiveness of the auxiliary loss in shaping the grasp value function while also enabling synchronous pose optimization.

Both `dGrasp` and $\texttt{dGrasp}_{\texttt{sync}}$ significantly outperform the base policies on the real task doubling the success rates. Even though we filter the final optimization results to only contain poses that point downwards (which is more of a safety feature) and adjust the final pose along the z axis by 1 cm to compensate the errors in camera calibration, the results indicate a significant improvement in zero-shot sim-to-real transfer capability.

Regarding the objects in the real world, the simpler objects like the rubber duck, the crochet ball and the 3D printed block were grasped reliably by all policies. All struggled however, with more complex objects like the power drill and the hiking boot. The predicted grasps commonly just collided with the objects. The policies also often failed to grasp the shampoo bottle, however mainly due to slippage. While `Base` and $\texttt{Base}_{\texttt{6d}}$ never grasped the Lego tire, the shuffled Rubik's cube, the dental floss and the canned tomato, both `dGrasp` and $\texttt{dGrasp}_{\texttt{sync}}$ were successful in 75% of the trials. In case of the Lego tire, the `Base` policies ended up in one of the corners of the robots workspace, which we believe can be accounted for the tire having a similar dark color as the plate in the workspace. Both, the dental floss and the canned tomato requires precise positioning, the first due to being small and the second because it is heavy and slippery. The performance of the `dGrasp` policies on these objects indicate an improved utilization of the geometric representation provided by the NeRF.

The failed grasps in the real-world experiments, which may be attributed to two primary factors: inaccuracies in the objective function and limitations in the optimization process of the implicit policy. The objective function relies heavily on precise camera calibration. Any calibration errors can propagate into the grasp evaluation process, leading to suboptimal or misaligned candidates. Additionally, the optimization process was parameterized based on simulated experiments. While the use of NeRFs as a foundational representation partially bridges the domain gap between simulation and the real world, residual discrepancies can still influence the optimization dynamics. These discrepancies may

amplify the occurrence of extreme local extrema in the optimization landscape, potentially causing the optimizer to get stuck or converge to unintended solutions.

Overall, the results show that integrating the demonstration trajectories via the auxiliary loss $\mathcal{L}_{aux}$ into the training process significantly enhances the performance of the grasping policies, particularly in zero-shot sim-to-real transfer.

6. Conclusion

In this work, we propose an auxiliary loss for augmenting the training of a NeRF-informed grasp value function, aimed at improving the optimization landscape of an implicit grasp policy. This auxiliary loss supervises the gradients of the value function using demonstrated grasp trajectories, and requires second order optimization during the training of the neural network model. Our experiments focus on the generalization capabilities of these policies, training models on simple simulated grasps and testing them on cluttered and novel objects in simulation and also in real-world settings through zero-shot sim-to-real transfer. The results demonstrate significant improvements in one out of three simulated tasks and in zero-shot sim-to-real transfer, suggesting that the auxiliary loss contributes to more stable policy optimization, thereby enabling more reliable identification of successful grasp poses.

Despite these advancements, the model's generalization to complex objects remains a challenge, which could likely be addressed with a more diverse dataset. While current success rates may not be sufficient for high-demand real-world applications, the promising results achieved with a small, simple dataset underscore the potential of our method.

Moreover, the proposed approach is not limited to grasping tasks and could be extended to a wide range of policies. The use of NeRFs as a scene representation, in particular, seems to significantly enhance generalization, offering the possibility of developing highly capable general models that can be easily fine-tuned for specific tasks. Although our implementation was in an open-loop context, the findings on synchronous pose optimization suggest that transitioning to a closed-loop framework could be viable, opening the door to integrating these methods into more complex and sophisticated tasks. This highlights the potential of NeRFs and our approach as a robust framework for developing more reliable, adaptable, and scalable robotic systems.

We acknowledge the absence of additional baseline comparisons as a limitation of this study. This work aims to establish the foundation for NeRF-based implicit policies. Future studies will include thorough comparisons with other state-of-the-art approaches to validate the broader applicability of the proposed method.

CRediT authorship contribution statement

Gergely Sóti: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Xi Huang:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization. **Christian Wurll:** Resources, Funding acquisition. **Björn Hein:** Writing – review & editing, Resources, Project administration, Funding acquisition.

Declaration of Generative AI and AI-assisted technologies in the writing process

Statement: During the preparation of this work the author(s) used ChatGPT in order to improve readability of the manuscript. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is being conducted as part of the KI5GRob project funded by the German Federal Ministry of Education and Research (BMBF) under project number 13FH579KX9.

Data availability

Data will be made available on request.

References

[1] P. Florence, C. Lynch, A. Zeng, O.A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, J. Tompson, Implicit behavioral cloning, in: Conference on Robot Learning, PMLR, 2022, pp. 158–168.

[2] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, S. Song, Diffusion policy: Visuomotor policy learning via action diffusion, 2023, arXiv preprint arXiv:2303.04137.

[3] B. Mildenhall, P.P. Srinivasan, M. Tancik, J.T. Barron, R. Ramamoorthi, R. Ng, NeRF: Representing scenes as neural radiance fields for view synthesis, in: ECCV, 2020.

[4] K.-E. Lin, Y.-C. Lin, W.-S. Lai, T.-Y. Lin, Y.-C. Shih, R. Ramamoorthi, Vision transformer for nerf-based view synthesis from a single input image, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2023, pp. 806–815.

[5] G. Sóti, X. Huang, C. Wurll, B. Hein, 6-DoF grasp pose evaluation and optimization via transfer learning from NeRFs, 2024, arXiv:2401.07935.

[6] Y. Du, S. Li, J. Tenenbaum, I. Mordatch, Improved contrastive divergence training of energy based models, 2020, arXiv preprint arXiv:2012.01316.

[7] J. Bohg, A. Morales, T. Asfour, D. Kragic, Data-driven grasp synthesis—a survey, IEEE Trans. Robot. 30 (2) (2013) 289–309.

[8] K. Kleeberger, R. Bormann, W. Kraus, M.F. Huber, A survey on learning-based robotic grasping, Curr. Robot. Rep. 1 (2020) 239–249.

[9] R. Newbury, M. Gu, L. Chumbley, A. Mousavian, C. Eppner, J. Leitner, J. Bohg, A. Morales, T. Asfour, D. Kragic, et al., Deep learning approaches to grasp synthesis: A review, IEEE Trans. Robot. (2023).

[10] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, in: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2017, pp. 23–30.

[11] Z. Dong, S. Liu, T. Zhou, H. Cheng, L. Zeng, X. Yu, H. Liu, Ppr-net: point-wise pose regression network for instance segmentation and 6d pose estimation in bin-picking scenarios, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2019, pp. 1773–1780.

[12] K. Kleeberger, M.F. Huber, Single shot 6d object pose estimation, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2020, pp. 6239–6245.

[13] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, D. Quillen, Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, Int. J. Robot. Res. 37 (4–5) (2018) 421–436.

[14] S. Song, A. Zeng, J. Lee, T. Funkhouser, Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations, IEEE Robot. Autom. Lett. 5 (3) (2020) 4978–4985.

[15] L. Berscheid, C. Friedrich, T. Kröger, Robot learning of 6 dof grasping using model-based adaptive primitives, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 4474–4480.

[16] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, K. Goldberg, Learning ambidextrous robot grasping policies, Science Robotics 4 (26) (2019) eaau4984.

[17] C. Eppner, A. Mousavian, D. Fox, ACRONYM: A large-scale grasp dataset based on simulation, in: 2021 IEEE Int. Conf. on Robotics and Automation, ICRA, 2020.

[18] Y.-C. Chen, A. Murali, B. Sundaralingam, W. Yang, A. Garg, D. Fox, Neural motion fields: Encoding grasp trajectories as implicit value functions, 2022, arXiv preprint arXiv:2206.14854.

[19] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, et al., Transporter networks: Rearranging the visual world for robotic manipulation, in: Conference on Robot Learning, PMLR, 2021, pp. 726–747.

[20] Y. Avigal, L. Berscheid, T. Asfour, T. Kröger, K. Goldberg, Speedfolding: Learning efficient bimanual folding of garments, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2022, pp. 1–8.

[21] P. Florence, L. Manuelli, R. Tedrake, Self-supervised correspondence in visuomotor policy learning, IEEE Robot. Autom. Lett. 5 (2) (2019) 492–499.

[22] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, S. Levine, Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 3758–3765.

[23] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[24] J. Ichnowski, Y. Avigal, J. Kerr, K. Goldberg, Dex-NeRF: Using a neural radiance field to grasp transparent objects, 2021, arXiv preprint arXiv:2110.14217.

[25] J. Kerr, L. Fu, H. Huang, Y. Avigal, M. Tancik, J. Ichnowski, A. Kanazawa, K. Goldberg, Evo-nerf: Evolving nerf for sequential robot grasping of transparent objects, in: 6th Annual Conference on Robot Learning, 2022.

[26] Q. Dai, Y. Zhu, Y. Geng, C. Ruan, J. Zhang, H. Wang, GraspNeRF: multiview-based 6-dof grasp detection for transparent and specular objects using generalizable nerf, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2023, pp. 1757–1763.

[27] V. Blukis, T. Lee, J. Tremblay, B. Wen, I.S. Kweon, K.-J. Yoon, D. Fox, S. Birchfield, One-shot neural fields for 3D object understanding, 2023, arXiv:2210.12126.

[28] T. Weng, D. Held, F. Meier, M. Mukadam, Neural grasp distance fields for robot manipulation, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2023, pp. 1814–1821.

[29] R. Fletcher, Practical methods of optimization, John Wiley & Sons, 2000.

[30] L. Yen-Chen, P. Florence, J.T. Barron, A. Rodriguez, P. Isola, T.-Y. Lin, Inerf: Inverting neural radiance fields for pose estimation, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2021, pp. 1323–1330.

[31] G. Sóti, B. Hein, C. Wurll, Gradient based grasp pose optimization on a NeRF that approximates grasp success, in: Intelligent Autonomous Systems 18, Springer, 2023.

[32] Y. Zhou, C. Barnes, J. Lu, J. Yang, H. Li, On the continuity of rotation representations in neural networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5745–5753.

[33] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, A.M. Dollar, The ycb object and model set: Towards common benchmarks for manipulation research, in: 2015 International Conference on Advanced Robotics, ICAR, IEEE, 2015, pp. 510–517.

Gergely Sóti received his M.Sc. degree in Computer Science from the Karlsruhe Institute of Technology (KIT), Germany, in 2018, where he is presently pursuing his doctoral studies. Currently, he is a Research Assistant at the Karlsruhe University of Applied Sciences, his work primarily involving learning-based robotic manipulation. His research interests are in scene representation and understanding, learning from behaviors, and human–robot collaboration.