Prof. Robot: Differentiable Robot Rendering Without Static and Self-Collisions

Supplementary Material

6. Network Architecture

The architecture of our network, illustrated in Figure 6, presents an elegantly straightforward design that comprises two distinct inputs and a singular dependencies.

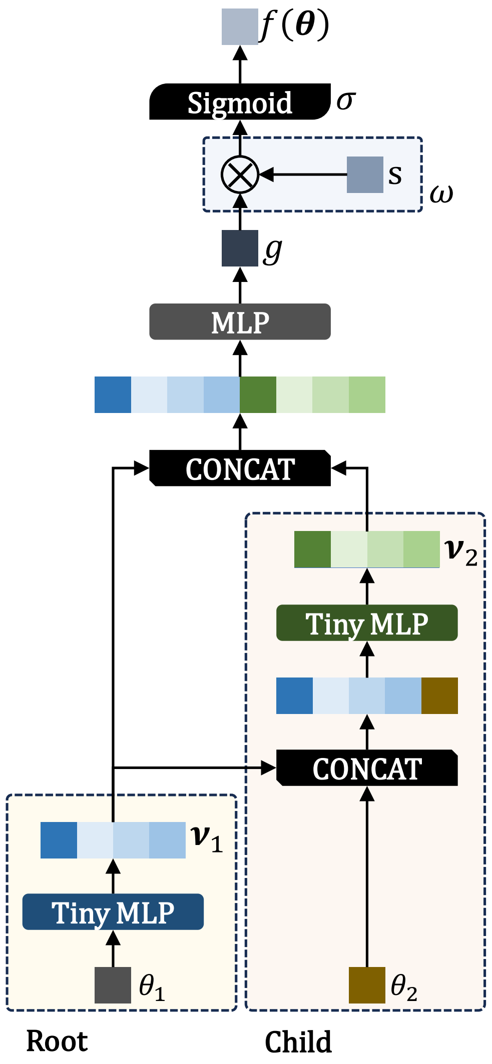

Figure 6. Architecture

7. Collision Resolution

We illustrate the process of optimizing the robot's joint angles for collision resolution in Figure 7. This transformation is further clarified by layering intermediate optimization results in Figure 8, thus offering a clearer depiction of how the robot gradually evolves from an initial state of collision to a final, non-collision posture.

8. Collision-free Trajectory

As outlined in Section 4.4.2, given the initial posture $\boldsymbol{\theta}_{\text{start}}$ and the target posture $\boldsymbol{\theta}_{\text{end}}$, we can delineate the trajectory for the transition from $\boldsymbol{\theta}_{\text{start}}$ to $\boldsymbol{\theta}_{\text{end}}$. We illustrate this process using three distinct methodologies.

8.1. Trajectory of Interpolation

Interpolation within the SO(2) plane offers a naive solution. Assuming the interpolation function in SO(2) is represented as $\mathcal{I}$, we can derive N points along the intermediate trajectory, $\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}_N}$, defined by

$$\hat{\boldsymbol{\theta}}_1, \ldots, \hat{\boldsymbol{\theta}_N} = \mathcal{I}\left(\boldsymbol{\theta}_{\text{start}}, \boldsymbol{\theta}_{\text{end}}\right) \tag{15}$$

However, this elementary interpolation method may yield invalid angles and potential collisions, as illustrated in Figure 11. If we were to rely on the trajectory generated by interpolation to instruct the robot's movements, it would face the risk of becoming ensnared at collision points, as depicted in Figure 12. Thus, it is imperative to employ our SDF model to optimize the control points of the intermediate poses. By incorporating our SDF perception, we enhance the interpolation poses through the following formulation:

$$\mathcal{L} = \gamma_1 \mathcal{L}_{\text{SDF}} + \gamma_2 \mathcal{L}_{\text{TV}} \tag{16}$$

where γ_1 and γ_2 represent scalar weights. The term $\mathcal{L}_{\text{TV}}$ is detailed in Section 4.4.2 and is expressed as:

$$\mathcal{L}_{\text{TV}} = \frac{1}{N-1} \sum_{i=1}^{N-1} \|\hat{\boldsymbol{\theta}}_{i+1} - \hat{\boldsymbol{\theta}}_i\| \tag{17}$$

Utilizing our SDF model, we generate a collision-free trajectory to guide the robot's movements, as shown in Figure 13. This trajectory is subsequently utilized to control the robot's motion, as illustrated in Figure 14.

8.2. Trajectory of Dr. Robot

We render the target posture $\boldsymbol{\theta}_{\text{end}}$ and utilize the intermediate optimized postures as control poses for Dr. Robot. Four views of the target posture $\boldsymbol{\theta}_{\text{end}}$ are presented in Figure 9. Starting from the initial parameters $\boldsymbol{\theta}_{\text{start}}$, the intermediate poses are optimized using the loss functions $\mathcal{L}_{\text{RGB}}$ and $\mathcal{L}_{\text{D}}$, as described in Section 4.4.1. The optimized intermediate postures are illustrated in Figure 15.

Directly using these postures for control can result in collisions, as shown in Figure 16. To mitigate this issue, our

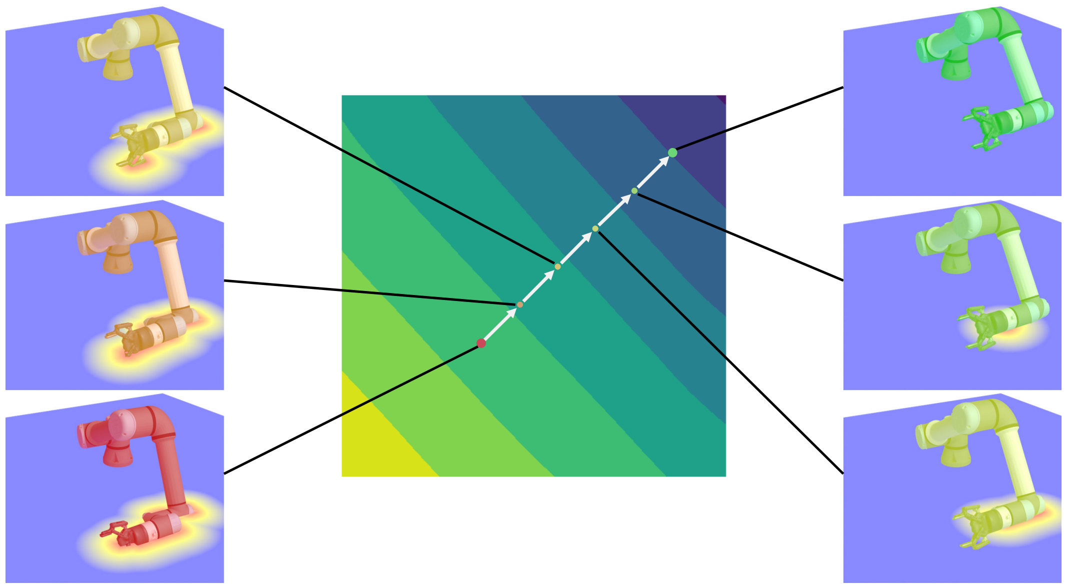

Figure 7. **The Visualization of SDF Optimization**. We adjust the angles of only two of the robot's joints while keeping the remaining joint angles fixed. Using these two joint angles as coordinate axes, we calculate the SDF values corresponding to various joint configurations. Subsequently, we apply the optimization method detailed in Section 4.3, visualizing the joint data points alongside their corresponding states of the robot in a plot. In this visualization, the robot transitions from **red** collision points to **green** non-collision points. For improved clarity, the regions where the robot comes into contact with the plane are depicted in **red**, the collision-free areas in **blue**, and the transitional zones in yellow.

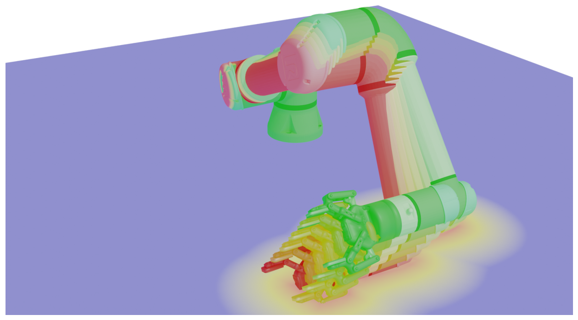

Figure 8. **The Stack of SDF Optimization**. To effectively illustrate the robot's gradual transition from a collision state to a non-collision state, we stack the intermediate results from the optimization process shown in Figure 7 for visualization purposes.

method refines the pose optimization using the following formulation:

$$\mathcal{L} = \gamma_1 \mathcal{L}_{\text{SDF}} + \gamma_2 \mathcal{L}_{\text{TV}} + \gamma_3 \mathcal{L}_{\text{SDF}} \tag{18}$$

The results are shown in Figure 17. We further refine the optimization by employing $\mathcal{L}_{\text{SDF}}$ and $\mathcal{L}_{\text{TV}}$, leading to the control outcomes illustrated in Figure 18. These results clearly demonstrate that our optimization method significantly reduces collisions between the robot and its surrounding environment.

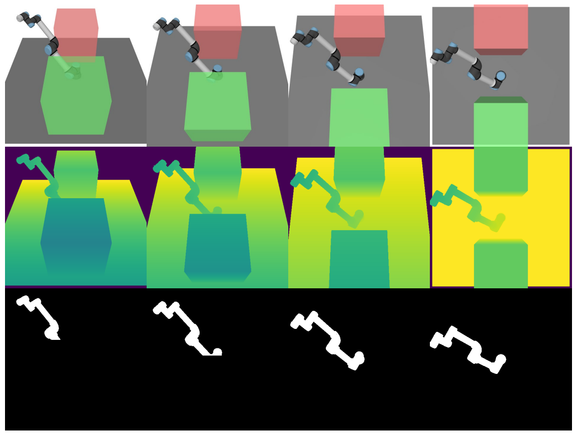

Figure 9. **The Dataset for Trajectory Optimization from Dr. Robot.** We use four different views as supervision, providing image, depth, and segmentation data for each perspective.

8.3. Trajectory of Tangent

Given the unique properties of the SDF, its value consistently increases along the gradient direction while remaining unchanged along tangential directions. However, with the multitude of possible tangential directions within the SDF, a crucial question emerges: which tangential direction should be chosen for gradient updates?

To address this, we leverage additional available information to guide the gradient. By projecting this external information onto the tangential plane of the SDF gradient, we can define an appropriate update direction. We propose Algorithm 1, which facilitates movement within a plane where the SDF value remains constant. This approach effectively enables the generation of a collision-free trajectory. The resulting outputs are depicted in Figure 19.

Algorithm 1 Gradient-Based Optimization with SDF Constraints

```
Require: $\boldsymbol{\theta}_{\text{start}}$, $\boldsymbol{\theta}_{\text{end}}$, $\delta$, maxIterations, $\varepsilon_{\text{tolerance}}$
Ensure: Optimized $\boldsymbol{\theta}$
 1: Initialize $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{\text{start}}$
 2: for iteration $= 1$ to maxIterations do
 3:     Compute the SDF $g(\boldsymbol{\theta})$ and the gradient $\boldsymbol{G}_{\text{SDF}}$
 4:     Normalize the $\boldsymbol{G}_{\text{SDF}}$: $\boldsymbol{G}_{\text{SDF}}^{\text{norm}} \leftarrow \frac{\boldsymbol{G}_{\text{SDF}}}{\|\boldsymbol{G}_{\text{SDF}}\|}$
 5:     Calculate the L1 distance to $\boldsymbol{\theta}_{\text{end}}$: $\mathcal{L}_{\boldsymbol{\theta}} \leftarrow |\boldsymbol{\theta} - \boldsymbol{\theta}_{\text{end}}|$, and compute the gradient of $\mathcal{L}_{\boldsymbol{\theta}}$: $\boldsymbol{G}_{\boldsymbol{\theta}}$
 6:     Determine the projection of $\boldsymbol{G}_{\boldsymbol{\theta}}$ onto the tangent of $\boldsymbol{G}_{\text{SDF}}^{\text{norm}}$: $\boldsymbol{D} \leftarrow \boldsymbol{G}_{\boldsymbol{\theta}} - (\boldsymbol{G}_{\text{SDF}}^{\text{norm}} \cdot \boldsymbol{G}_{\boldsymbol{\theta}})\boldsymbol{G}_{\text{SDF}}^{\text{norm}}$
 7:     Update $\boldsymbol{\theta}$: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \delta \boldsymbol{D}$
 8:     if $\mathcal{L}_{\boldsymbol{\theta}} < \varepsilon_{\text{tolerance}}$ then
 9:         break
10:     end if
11: end for
```

9. Comparison with Differentiable Simulator

We frame Prof. Robot as a lightweight plug-in that can be seamlessly integrated into other skill-learning pipelines. In specific, Prof. Robot has a compact JIT model size of only 3.1 MB, as opposed to differentiable simulation, which requires a storage of nearly 4.7 GB and additional robot assets of nearly 22.1 MB. When optimizing poses, our model and the PyTorch environment require just 712 MiB of GPU memory. In comparison, Brax consumes 11,445 MiB, making our approach approximately 16 times more memory-efficient. When optimizing poses, our approach approximately 16 times more memory-efficient. Additionally, while differentiable simulators are capable of backpropagating gradients from collisions to actions, they come with significant limitations, such as computational overhead and inaccurate gradient estimation [42]. Moreover, many simulators, such as MuJoCo and Isaac Gym, are not differentiable, and our Prof. Robot is compatible with these simulators.

10. Train with Alternative Regularizers

Although optimizations involving Eikonal equations are known to be unstable, recent work by [68] proposes a stable Eikonal loss (StEik) formulation. While incorporating StEik into our framework yields a modest 0.29% improvement in classification accuracy, we observe that its effectiveness is constrained by fundamental factors: StEik primarily targets optimization of complex geometric configurations, whereas our pose optimization task does not strictly require full adherence to the Eikonal property. Similarly, imposing Lipschitz constraints [31] improves classification accuracy by 0.35% through gradient stabilization, but introduces unintended consequences. This regularization relaxes the critical gradient normalization condition, thereby compromising essential SDF properties: (1) the geometric interpretation of SDF values as exact distance measurements to zero-level surfaces. (2) the directional significance of SDF gradients for surface orientation. Such properties prove crucial for downstream tasks like pose interpolation [55].

11. Manipulation with Dr. Robot

To showcase the efficacy of our method in robot control, we undertook a training regimen for Dr. Robot, enabling it to learn and execute actions based on video sequences. The effectiveness of this learning was then verified through a series of simulation experiments. We created a multi-view video of a pick operation using the Mujoco engine and employed Dr. Robot to optimize the actions required for each video frame. Subsequently, we assessed these optimized actions within the simulation environment, with a focus on analyzing their interaction with the desk to detect any collisions.

In the robot demonstration video, each frame corresponds to a control parameter $\boldsymbol{\theta}_i$. Our objective is to infer these $\boldsymbol{\theta}_i$ values from the video and subsequently use them to control the robot. While Dr. Robot has the capability to manipulate a robot, no demonstrations of this functionality have been presented so far. We are confident that with high-quality video input, Dr. Robot can effectively manipulate a real robot.

For the sake of brevity, we do not focus on the quality of the survival model; instead, we directly learn the robot's actions from the demonstration video. Using the Mujoco engine, we generate a robot manipulation video, as shown in Figure 10. To ensure precise manipulation data, we adopt a course-to-fine strategy. Finally, we incorporate our collision-aware module to prevent any incidents of collision during robot operation.

During the course stage, we utilize gradient accumulation, leveraging Dr. Robot to propagate the gradients from the three views before updating the parameters. In this phase, we adopt a larger learning rate and recursively optimize the parameters of the robot for each frame, performing five parameter updates before retaining the optimization results. In the fine stage, we still employ the method of gradient accumulation, but with a smaller learning rate and incorporate the TV loss mentioned in Section 4.4.2 to mitigate jitter during the optimization process, conducting ten

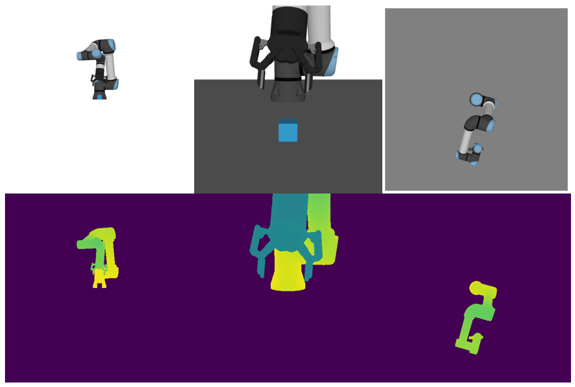

Figure 10. **The Dataset for Robot Manipulation**. Our dataset includes three perspectives captured across 160 frames. For supervision, we provide images alongside their corresponding depth data of the robot. To address the singularity of actions, we ensure multi-view consistency by utilizing three distinct perspectives. Additionally, to supervise fine-grained manipulation, we use a telephoto camera to capture close-up images of the robot's gripper.

parameter updates. The results are presented in Figure 20.

However, in our demonstration scenario, directly utilizing Dr. Robot for robot control can result in collisions. To circumvent this issue, we optimize the parameters $\boldsymbol{\theta}$ estimated by Dr. Robot using our SDF model. We categorize the robot's movements into two types: movement, which typically does not emphasize the posture of the end effector but focuses on avoiding collisions during motion, and operations, which prioritize the end effector's posture. Consequently, we classify the parameters $\boldsymbol{\theta}$ inferred by Dr. Robot into these two categories. For the movement parameters, we apply SDF loss for supervision, while for the operation parameters, we supervise based on the posture of the end effector. Throughout the optimization process, we also employ TV loss to ensure smoother postures and to prevent excessive jitter. Our evaluation primarily focuses on collision avoidance between the robot arm and the desktop during movement, as illustrated in Figure 21. The results demonstrate that our method effectively mitigates robot collisions.

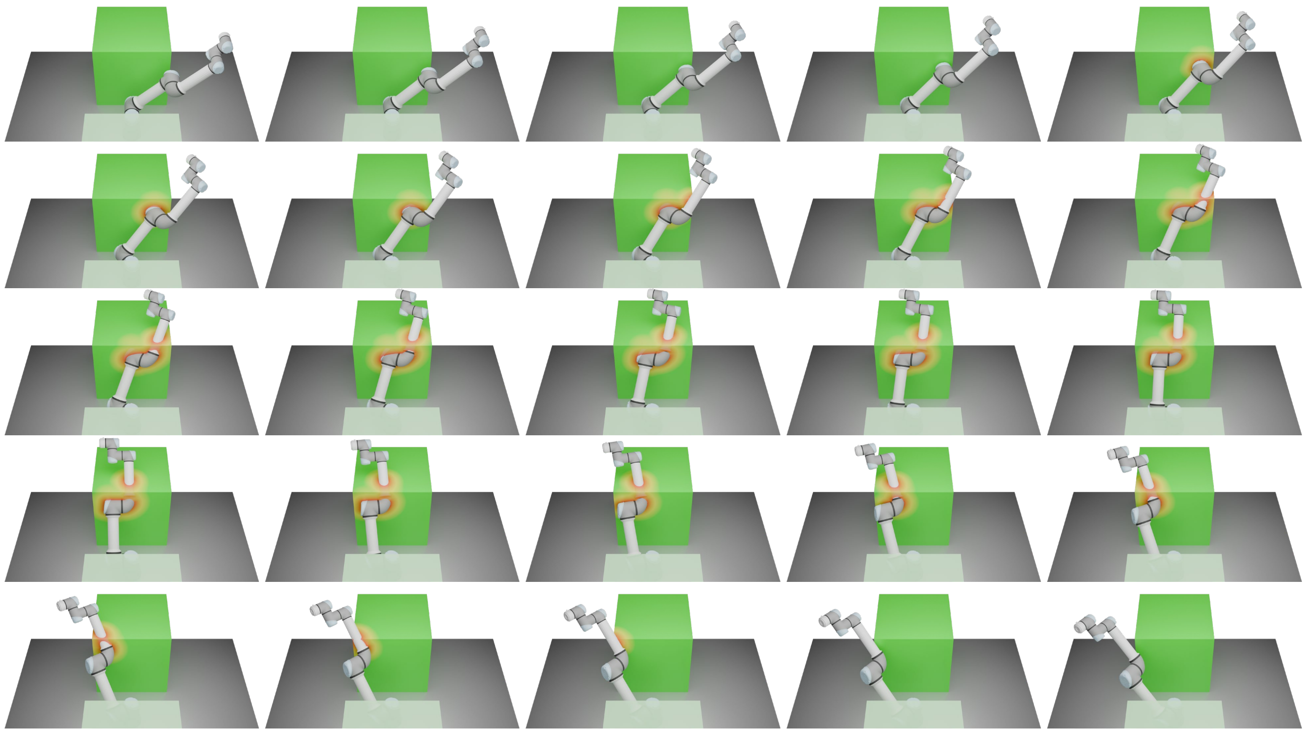

Figure 11. **Visualization of the interpolation trajectory.** The reading sequence progresses from left to right and top to bottom, adhering to this pattern consistently. When the robotic arm makes contact with the **green** block, the collision area will be highlighted in **red**, with a yellow gradient transitioning in between, a pattern that is replicated in other instances as well.

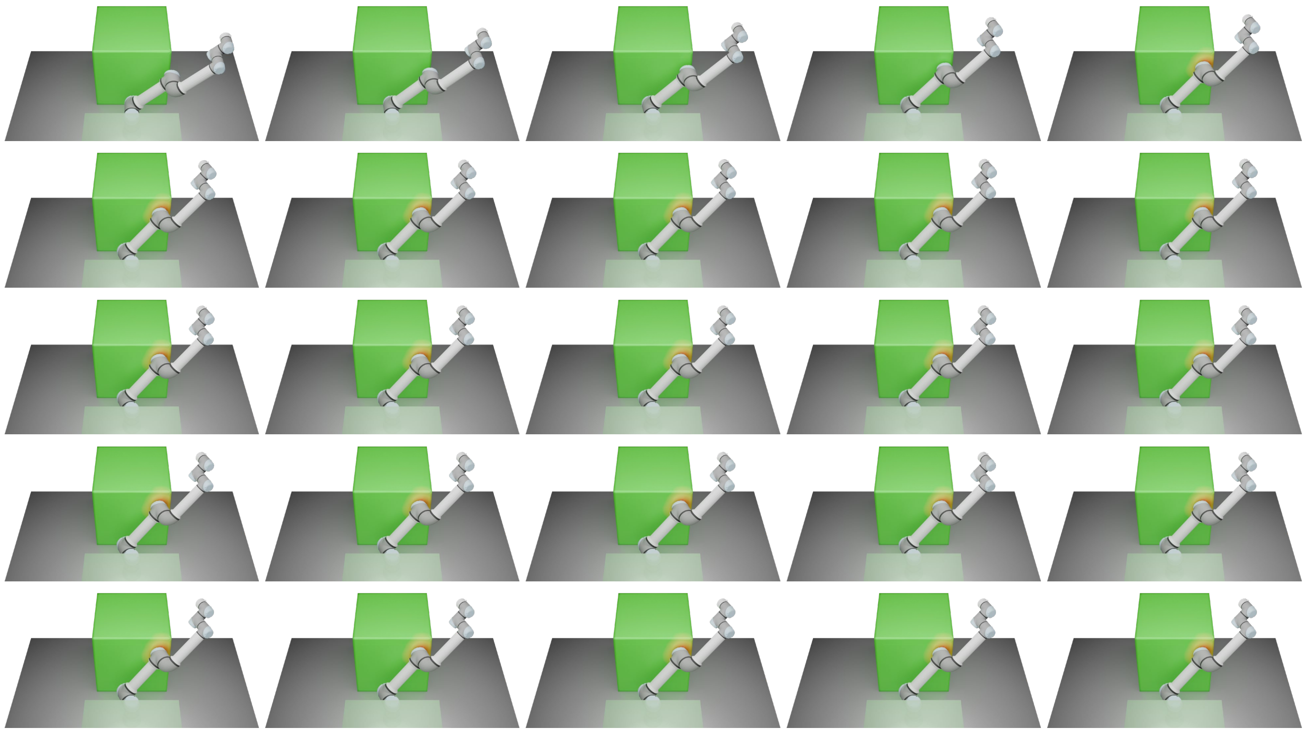

Figure 12. **Visualization of the use of the interpolation trajectory for robot control.**

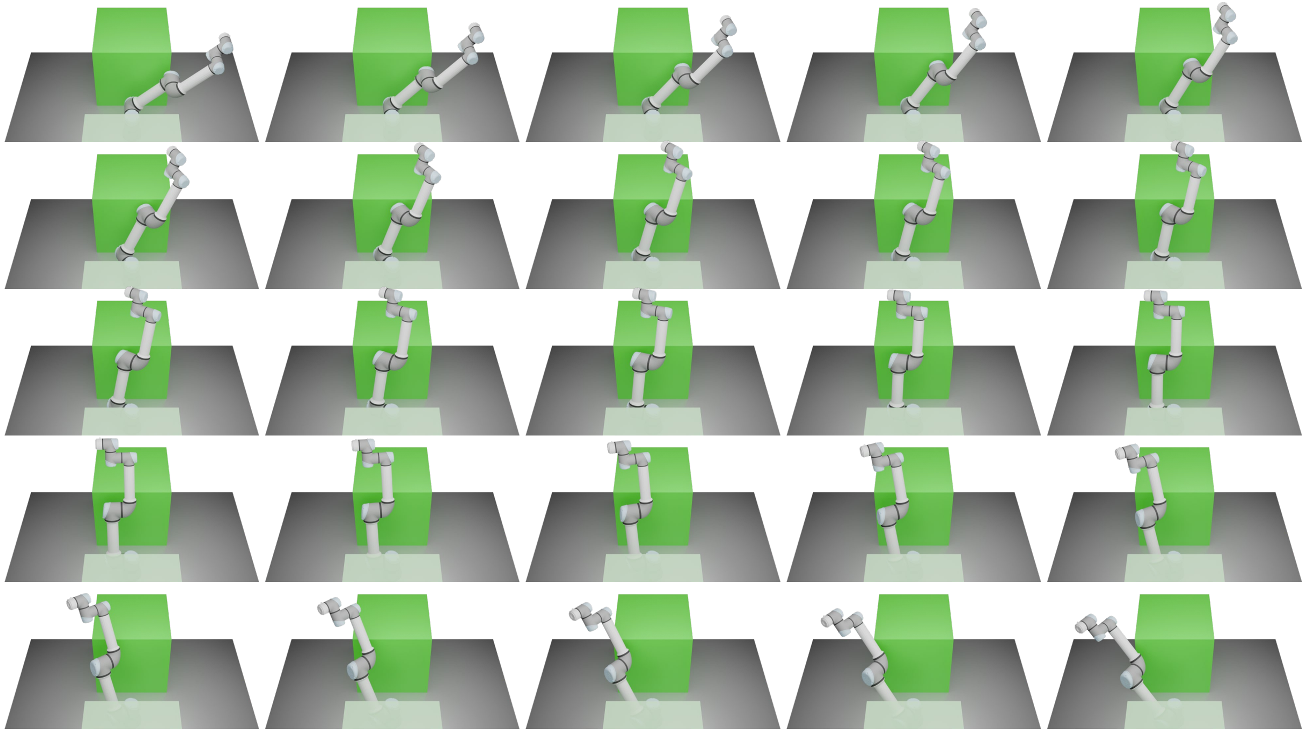
Figure 13. Visualization of our optimization trajectory derived from interpolation.

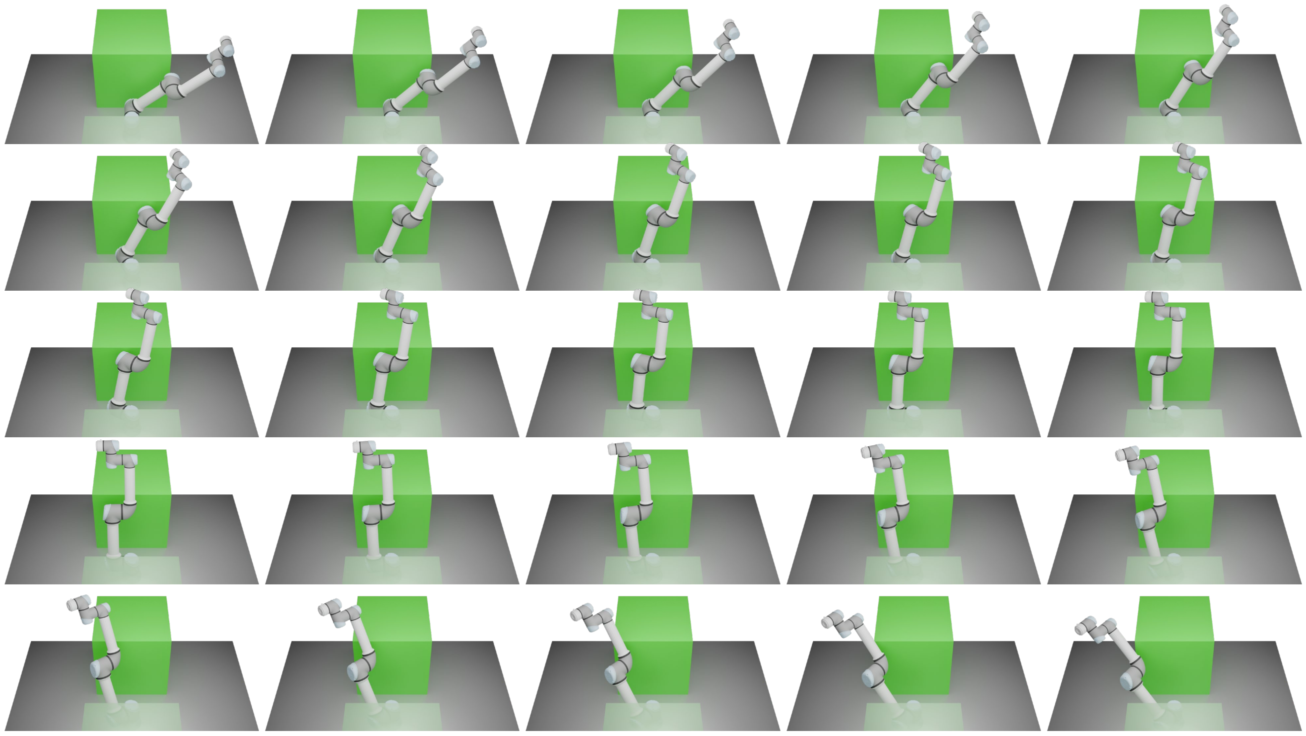
Figure 14. **Visualization of controlling the robot using our optimization trajectory from interpolation.**

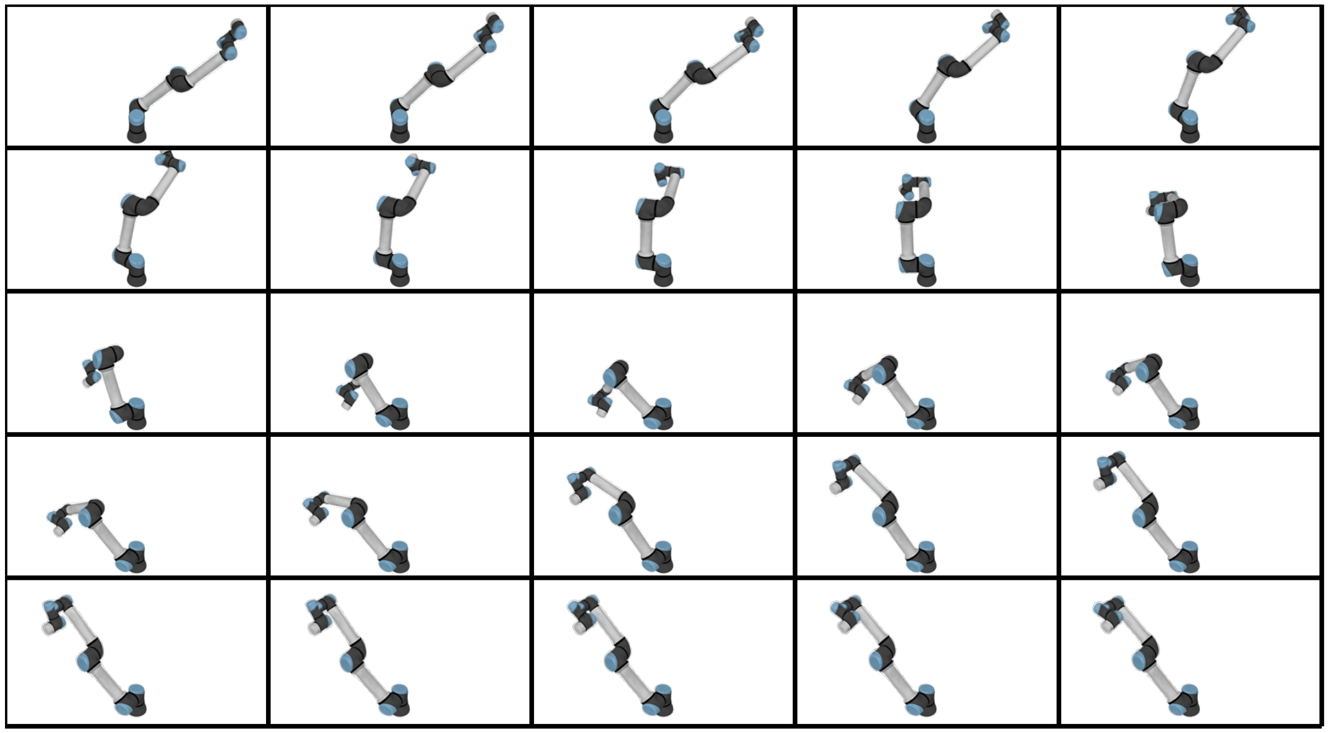
Figure 15. **Intermediate trajectory of Dr. Robot's optimization process.**

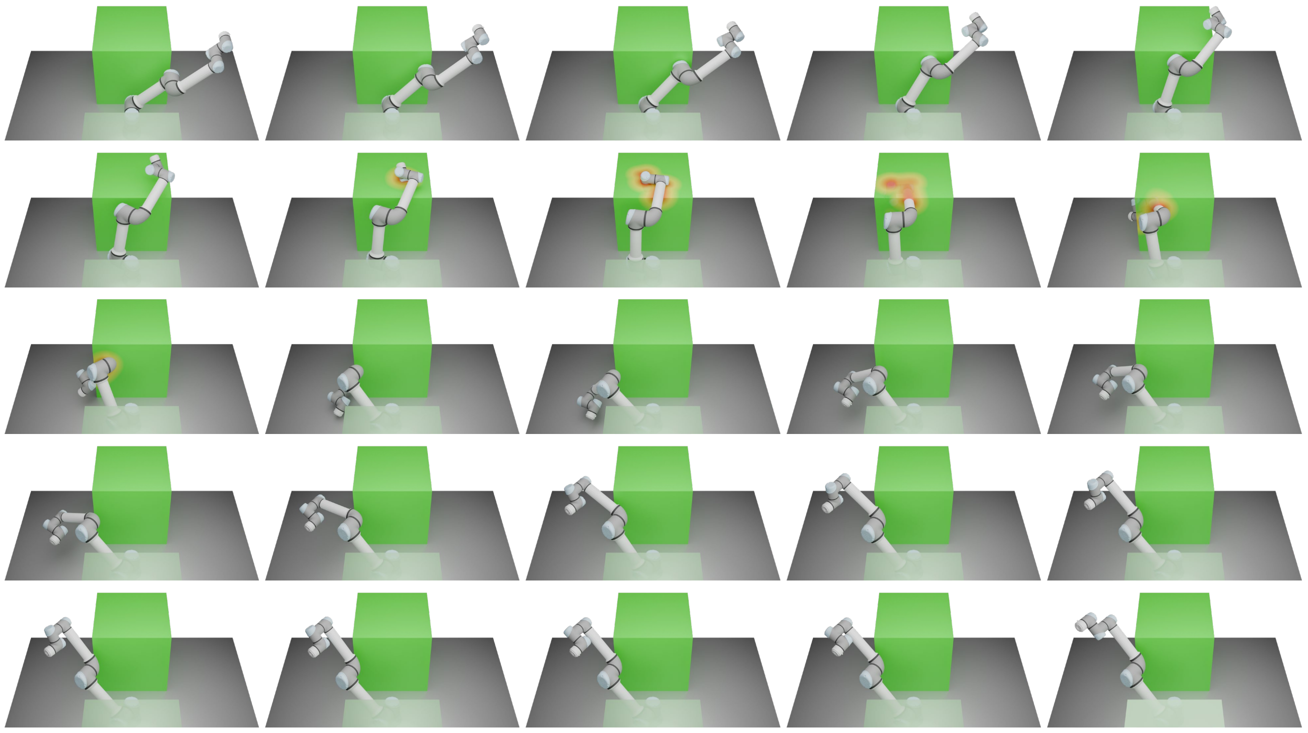
Figure 16. Visualization of the trajectory generated by Dr. Robot.

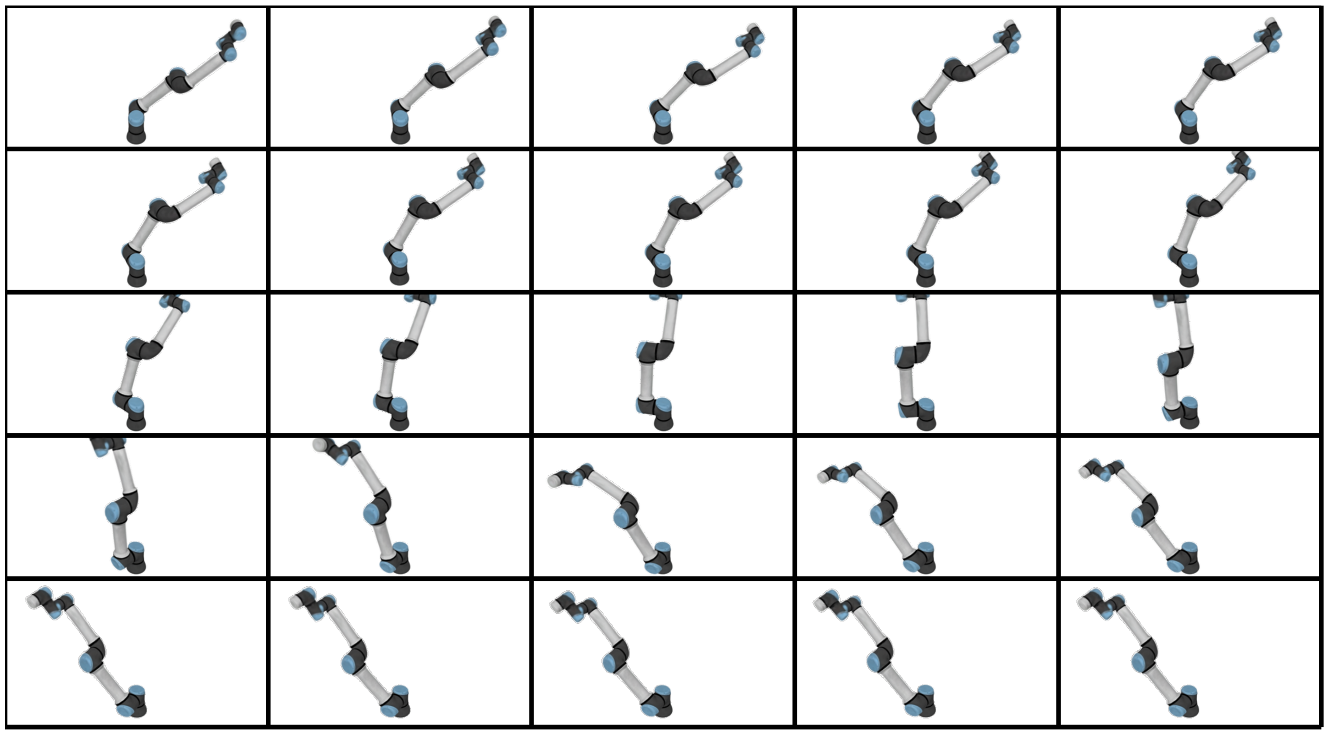
Figure 17. **Intermediate trajectory of Dr. Robot's optimization utilizing our SDF model.**

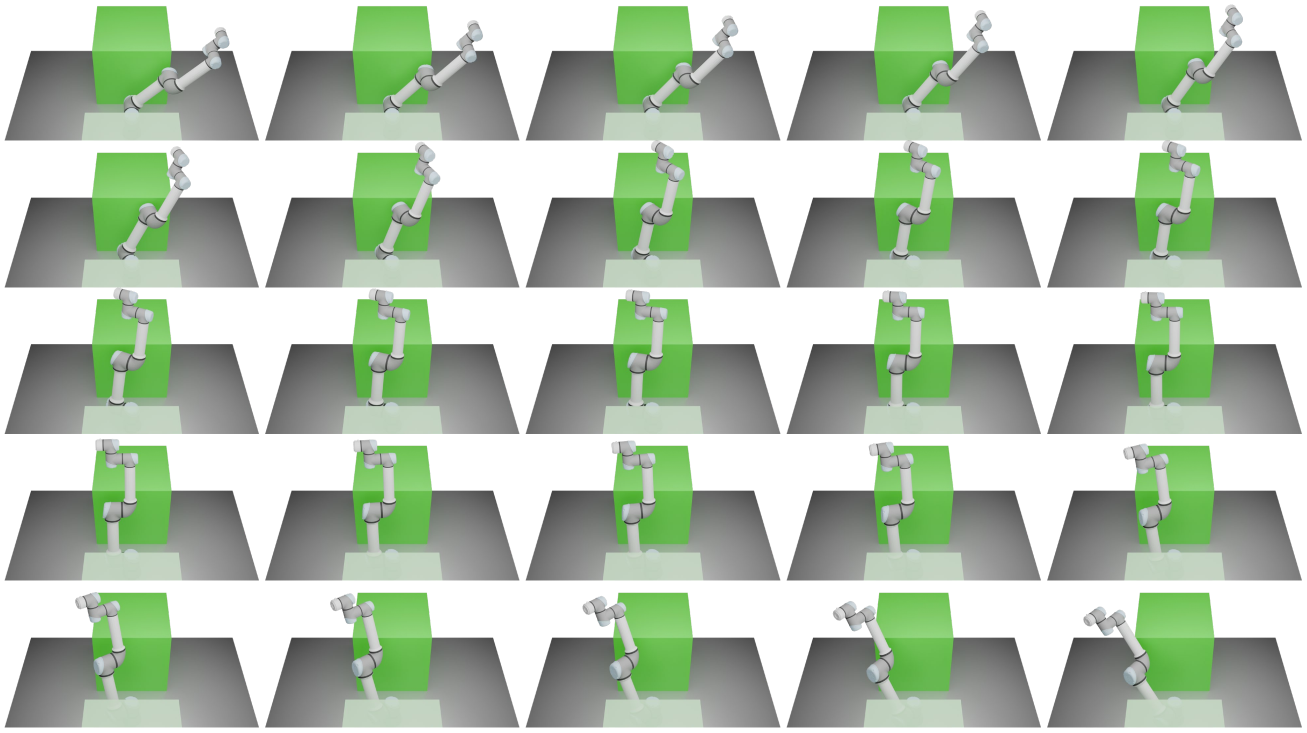
Figure 18. **Visualization of Dr. Robot's trajectory augmented by our SDF model.**

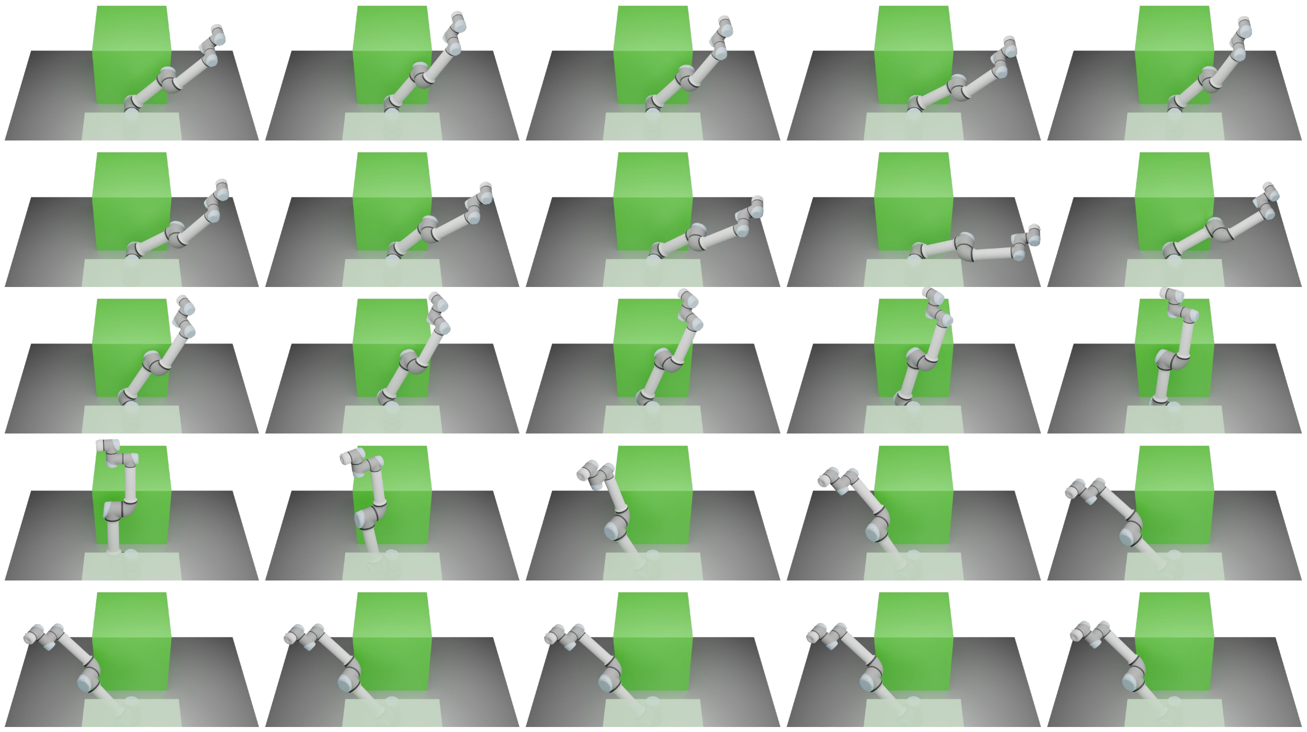
Figure 19. **Visualization of our tangent trajectory**. Given that we employed only gradient descent without the inclusion of momentum for optimization, while assigning a substantial value to δ to assist $\boldsymbol{\theta}$ in circumventing local minima, this approach led to considerable jitter in the robot depicted in the image.

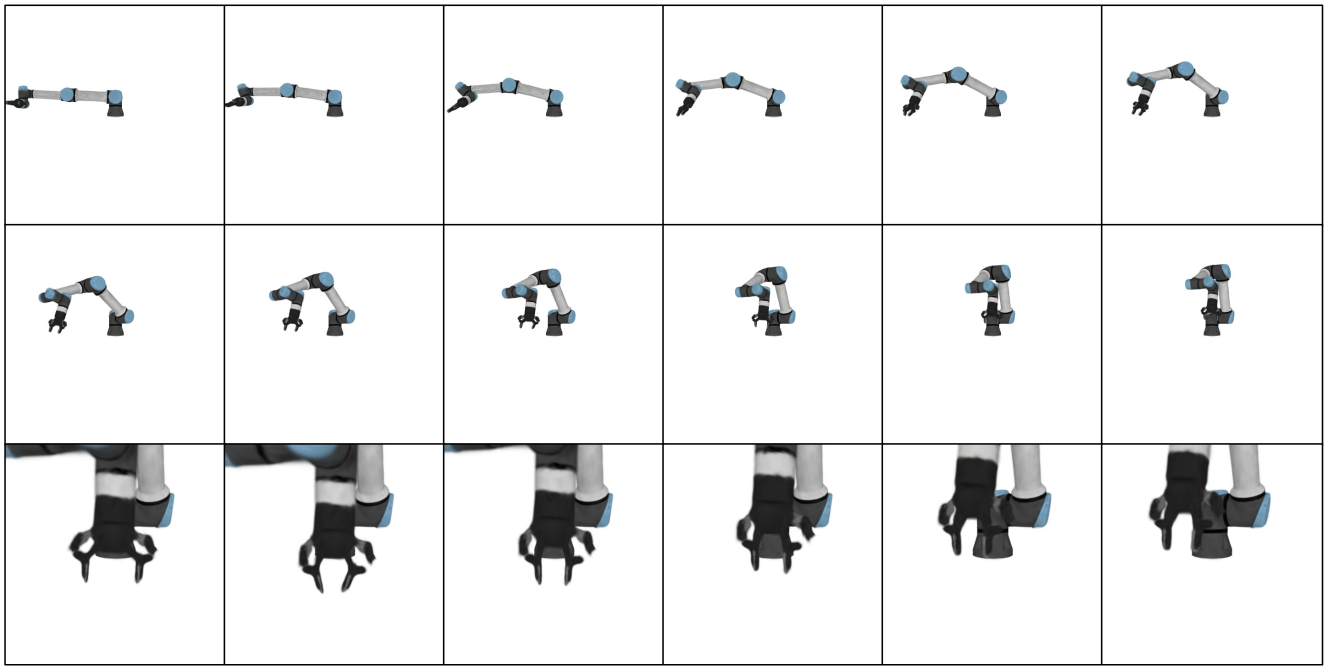
Figure 20. **Rendering results of robot control utilizing Dr. Robot.** The first two rows display operational images from a singular perspective, rendered by Dr. Robot, while the final row illustrates the gripper being precisely manipulated by Dr. Robot. Upon learning the parameter $\boldsymbol{\theta}$, Dr. Robot proceeds to control the robot.

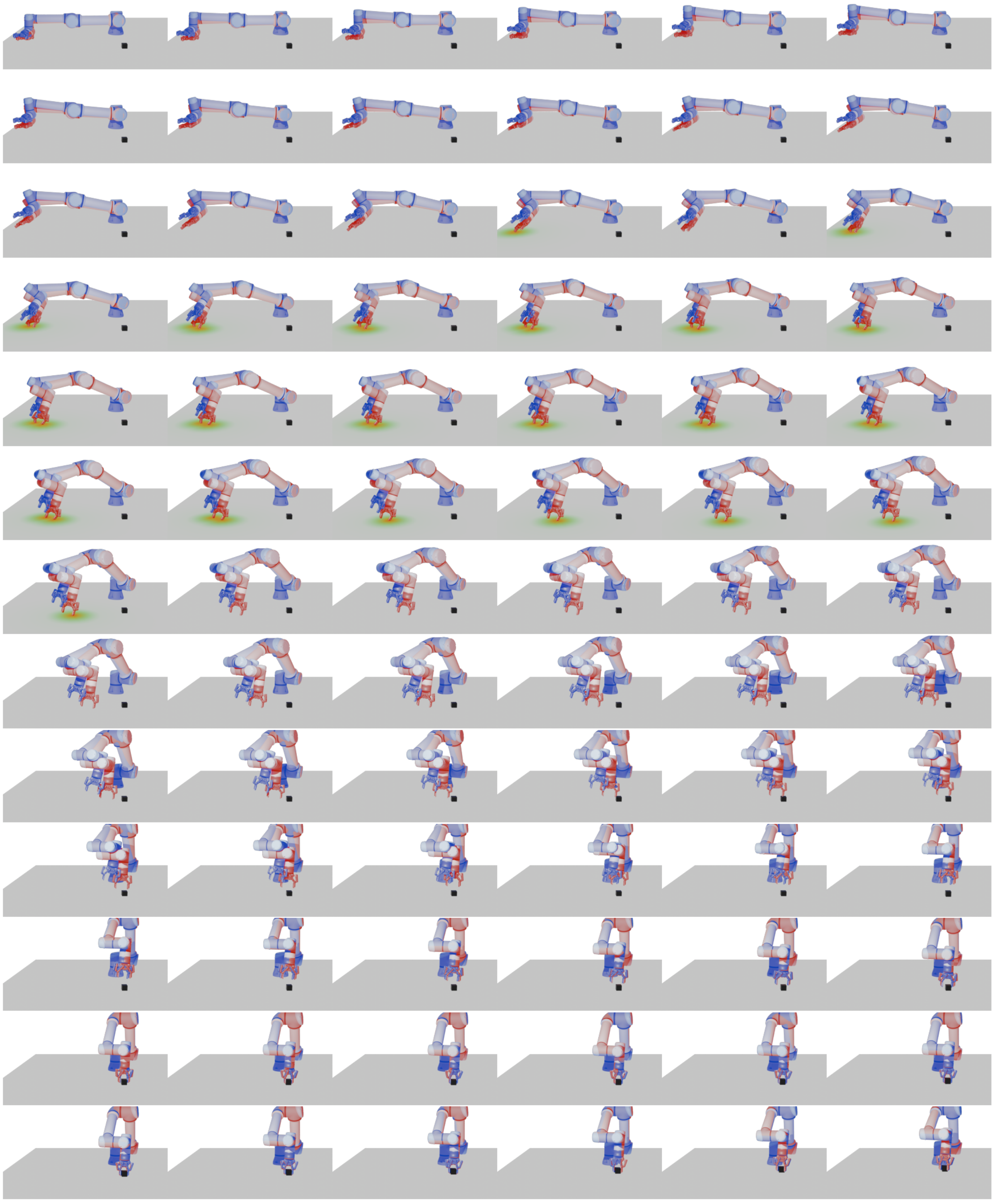

Figure 21. **Results of robot control employing the parameters θ learned by Dr. Robot**. The **red** robot arm in the image denotes direct control based on the original parameters learned by Dr. Robot, while the **blue** robot arm represents control following optimization through our SDF model. To highlight potential collisions, we elevated the plane of the initial 60 views by 0.005 m solely during the rendering process, with collision points marked in **red** on the **gray** desktop.