

Graph-Based Model Predictive Visual Imitation Learning

Graphbasiertes model-prädiktives visuelles Lernen von Imitation

Bachelor thesis by Marek Daniv

Informationssystemtechnik

Date of submission: April 4, 2022

1. Review: Prof. PhD. Jan Peters, FB Informatik

2. Review: MSc. João Andre Correia Cavalho, FB Informatik
Darmstadt



TECHNISCHE
UNIVERSITÄT
DARMSTADT



Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Marek Daniv

Informationssystemtechnik, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 4. April 2022

M. Daniv

Contents

1	Introduction	6
1.1	Related work	7
2	Foundations	9
2.1	Hand Detection And Pose Estimation	9
2.2	Object Pose Estimation	10
2.3	Measuring Distances Between Orientations	11
2.4	Graph-Based Task Representation	12
2.5	Imitation Learning With Model Predictive Control	13
3	Method	16
3.1	Improvements On The Graph-Based Task Representation	16
3.2	Data Collection	24
3.3	Control Algorithm	27
4	Experiments	32
4.1	Task Domain	32
4.2	Implementation Details	34
4.3	Evaluation On Pick And Place Tasks	34
5	Conclusion	43
5.1	Discussion	43
5.2	Future Work	44
6	Appendix	46
6.1	Proof Of Metric Properties	46
6.2	Experiment Hyperparameters	47


List of Figures

1.1	Scope of this work, which includes the collection of demonstration data in the form of a video (<i>left</i>), a graph-based representation of the demonstrated task (<i>middle</i>) and learning as well as execution of control policy to facilitate imitation by a given robot (<i>right</i>)	7
2.1	Hand landmarks as specified by MediaPipe Hands	9
2.2	Visualization of ArUco marker pose estimation	10
3.1	Example of a visual entity graph constructed during step t of a task demonstration. Depicted is the representation of the human hand V_h and 3 nodes V_{o1} , V_{o2} , V_{o3} corresponding to relevant objects. The <i>green piece with a T-shape</i> is currently considered to be the <i>anchor</i> because the human expert manipulates it. Consequently, all edges connected to it are considered active in this VEG.	17
3.2	Extracted human hand pose that serves as an optimization target for the panda gripper (<i>left</i>). This pose consists of a centroid represented by the <i>black dot</i> and x-axis <i>red</i> , y-axis <i>green</i> , z-axis <i>blue</i> . This inferred pose is determined by the detected positions of the 3 annotated landmarks <i>INDEX</i> , <i>THUMB</i> , and <i>INDEX_MCP</i>	19
3.3	Visualization of optimal gripper poses in the relation to the object of interest during the different grasping phases. These phases consist of i) moving to the pre-grasp position, ii) continuing to the final grasping position, iii) closing the gripper, and finally iv) lifting the object vertically.	22
3.4	Visualization of hand-crafted release phases. These phases consist of following the unpolished demonstrated trajectory (<i>top</i>), manipulating the object pose in a way that aligns one of its axes with the table normal vector (<i>middle</i>), opening the gripper, and releasing the object (<i>bottom</i>)	23
3.5	Single frame of a captured demonstration video. The results of our object estimation and the outputs of the hand detection framework are annotated	24

3.6	Visualization of raw and filtered data differences. We compare inferred spatial data from an object labeled <i>U</i> , <i>thumb</i> , <i>index finger</i> , <i>middle finger</i> , <i>ring finger</i> , and <i>pinky finger</i>	26
3.7	Visualization of sampling strategy effects on the overall cost as well as end-effector trajectory. Values in <i>green</i> correspond to modified action sampling, while the value in <i>red</i> belongs to execution with raw Gaussian sampling. We can reduce total cost during execution significantly (<i>left graph</i>) and get much smoother trajectories at the same time (<i>right graph</i>).	31
4.1	Example of a solved “Ubongo” puzzle made up of complex individually colored pieces	32
4.2	Photo of a manually printed “Ubongo” piece with increased scale. The new dimensions per cube are 3.5 cm × 3.5 cm × 3.5 cm as annotated	33
4.3	Initial Configurations: Corresponding time steps are marked in <i>green</i>	35
4.4	Reaching Initial Human Pose	35
4.5	Directly Imitating Demonstrated Task Execution	36
4.6	Reaching The Pre-Grasp Positon	36
4.7	Moving To Grasping Position	37
4.8	Closing The Gripper	37
4.9	Lifting The Object	38
4.10	Directly Imitating Demonstrated Task Execution	38
4.11	Directly Imitating Demonstrated Task Execution	39
4.12	Reaching Pre-Release Pose	39
4.13	Releasing The Held Object	40
4.14	Directly Imitating Demonstrated Task Execution	40
4.15	Simulation of the resulting imitation. Shown here is the first Pick and Place task in the sequence with (<i>top left</i>) initial configuration, (<i>top right</i>) first pre-grasp pose, (<i>bottom left</i>) first grasp, (<i>bottom right</i>) first release	41



4.16 Simulation of the resulting imitation. Shown here is the second Pick and Place task in the sequence with (<i>top left</i>) imitating human transition, (<i>top right</i>) second pre-grasp pose, (<i>bottom left</i>) second grasp, (<i>bottom right</i>) second release	42
---	----



For my family and all friends who accompany me along my way.



Acknowledgement

*This thesis would not have been possible without the outstanding support of my supervisors
João Carvalho and Suman Pal.*

*I am very grateful to all members of the IAS lab, who offered me the possibility to work with
real robotic systems and created a great learning environment.*

Special thanks to my roommate Dorian Wiese for the exceptional concept art he drew for me.

Abstract

One problem of modern robotics is that teaching a robot to perform a particular task requires expert knowledge, time, and many physical trials within its workplace. We as humans can learn a new skill much more efficiently by simply watching another human once. The aim of this thesis is to introduce a complete pipeline from data collection to execution, which enables training a robot with the information encoded in a video of a single human demonstration. Previous methods rely on large amounts of prior training of complex visual detection networks or access to many optimal executions for meta learning. However, we propose a *one-shot* approach that lies in the domain of *visual imitation learning*. To this end, we introduce a method for efficient data extraction from RGBD video, a graph-based task-representation that allows for robust generalization, and finally, a model predictive control algorithm using a composition of task-related costs as well as functions enforcing auxiliary requirements of robot manipulation. We present solutions to problems arising from visual inference, the human-robot dynamics gap, inaccuracies in the system model, and executions within novel environments. We also study our results on a Pybullet simulation of the Franka Panda robot with demonstration videos captured in the real world that revolved around the interesting task domain of the social game “Ubungo”.

1 Introduction

Learning by watching others is an integral way of human skill acquisition. Adapting this technique for robots has become a big topic in machine learning recently because it would enable them to learn complicated tasks from human experts in a way that is very natural for their teachers. Humans already show the great potential of this method. We have the impressive ability to imitate another human being after watching him perform a new skill once while also adapting the motion to our morphology and external environment [1]. Thousands of hours of tutorial videos being uploaded on modern video platforms already encode knowledge about every conceivable task, which allows remote learning of new skills with just a few clicks. Unlocking this database for machine learning approaches or finding a way to transport knowledge from a human teacher in the same way would leverage the need for manual programming or complicated software to train a robot on a new task. We propose a framework covering every aspect of this approach, from capturing a demonstration video with depth data to the imitation of the shown skill by an arbitrary robot. Our method works within a *one-shot* setting only a single video demonstration of the task and requires no prior training. Our technique also can generalize the seen execution to the workspace of a given robot and is invariant against differences in initial poses of task-relevant objects. Combined, this allows the demonstration to be captured in an environment that is natural for a human teacher, like a cluttered office desk, but executed in the environment of the robotic agent.

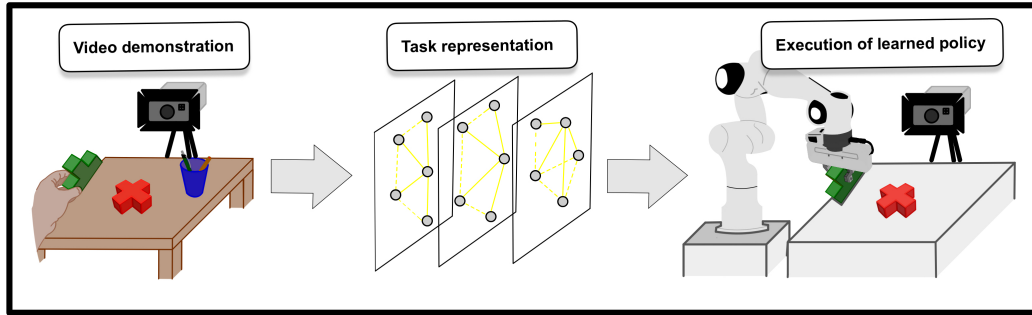


Figure 1.1: Scope of this work, which includes the collection of demonstration data in the form of a video (*left*), a graph-based representation of the demonstrated task (*middle*) and learning as well as execution of control policy to facilitate imitation by a given robot (*right*)

To accomplish these goals we propose i) data collection pipeline, ii) an improvement of graph-based task-representation introduced in this work [2] and iii) a method to learn a state-based cost function from it. This cost is part of a model predictive control algorithm to iv) facilitate the imitation of the demonstrated skill under the consideration of general side-goals of robot manipulation. We show v) results with a simulated Panda robot that could be recreated in a real system in the future without requiring great effort. We only assume an RGB-D video with pre-labeled objects as training data and an approximate dynamics model for the agent’s environment.

1.1 Related work

Learning from demonstration is a common strategy in Reinforcement Learning (RL), with Behavior cloning [3] and Inverse Reinforcement Learning [4] being the most prominent approaches. While both aim to infer knowledge from observing an expert execution of a given task, the first derives an optimal state-action mapping, while the latter tries to learn a cost function. However, many of these techniques require that the given demonstration consists of kinesthetic teaching [5] [6], i.e., physically guiding the robot in the same environment, it is supposed to perform the skill in. This requirement assures that the expert in the demonstration has the same dynamics, action-space, and state-space as the agent during the imitation [7], which does not hold up for learning from video data, especially when a human executes the task. This problem is further compounded by the

difficulty of inferring system states or expert actions from video data in the first place, while humans can do this naturally [8]. Some approaches circumvent the visual inference problem by using full-frame image encodings as latent state representations [9] [10], but in turn, require a setup of complex neural networks that need to be trained with large amounts of training data. They also suffer from biases in observations, such as changes in background or viewing angle. Object or feature-centered state representations [11] have been shown to increase data efficiency, with graph-based approaches [2] [12] being quite popular. However, even very data-efficient learning approaches, known as few-shot or one-shot methods, rely on multiple previous experiments as priors, for example, to derive system models [13] or establish a library of primitive movements [14]. These pre-learned parameters are commonly adjusted to a novel scenario in a single learning phase. We have decided not to use any prior training in a given setting to keep the users' effort as low as possible. Accordingly, we want to derive a control policy from a single optimal task execution that has to be generalized enough to work in unseen environments, and for an arbitrary agent. We also require our control method to suffice general challenges of robot manipulation such as collision- and singularity avoidance, smooth trajectories, and reactivity. These conditions can be problematic for approaches that only output high-level actions [15], but cost-based techniques like model predictive control [16] [17] allow for the parallel completion of multiple side goals while completing a task. They are based on a composition of different goal-specific cost functions and online computation of control policies that are locally optimal regarding the current state of the environment.

2 Foundations

2.1 Hand Detection And Pose Estimation

Our task representation relies on the pose of the human expert’s hand during each frame. To estimate it, we use the state-of-the-art detection framework *MediaPipe Hands* [18], which consists of two stages. First, a palm detector is run over the whole image to locate the hand center. Afterward a more precise model will determine the exact locations of specific landmarks, as seen in fig 2.1. This approach makes the technique is quite robust against partial invisible hands and self-occlusions. However, we discovered during our experiments that there are troubles with long sleeves, which is probably caused by a bias in the training data. We use this framework to get the locations of the three landmarks *THUMB_TIP*, *INDEX_FINGER_TIP*, and *INDEX_FINGER_MCP* in pixel coordinates for each frame. These coordinates can be transformed into spatial 3D coordinates, based on the point cloud provided by our *Microsoft Azure Kinect DK* RGBD camera.

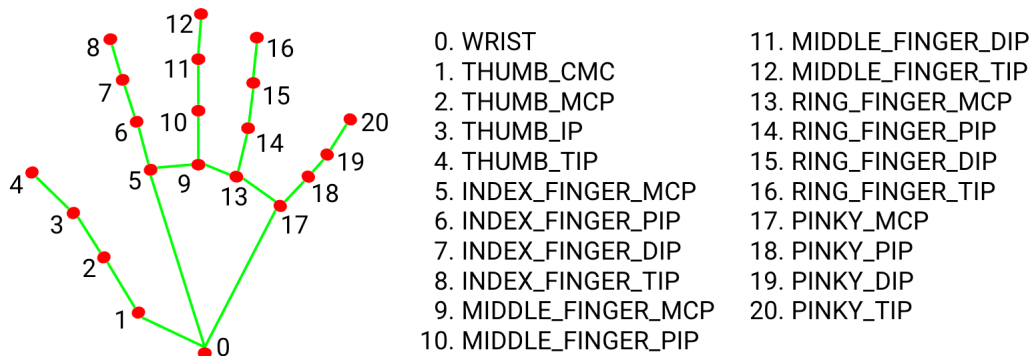


Figure 2.1: Hand landmarks as specified by MediaPipe Hands

2.2 Object Pose Estimation

To facilitate object pose estimation from video data and during task execution, we chose a popular approach in Computer Vision: Fiducial markers. More specifically, we worked with the *ArUco* library [19] in *OpenCV*[20]. Although this approach has lower accuracy and a high vulnerability to occlusions, it requires a minimal setup in turn compared to similar methods like *Optitrack*[21]. After pre-generating a set of fiducial markers and placing them on task-relevant objects, we can estimate their 6D poses directly from 2D images. Since we calculate the pose of the marker, the user has to specify a manual translation and rotation to the origin of the object coordinate axes system. This transformation can be calculated for individual objects since each marker encodes a unique id.

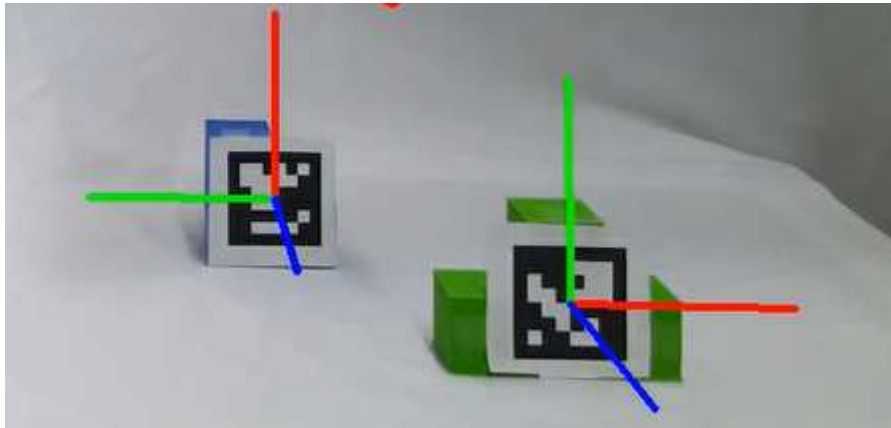


Figure 2.2: Visualization of ArUco marker pose estimation

2.3 Measuring Distances Between Orientations

Orientations of entities, in general, can be described as the rotation from the world axes onto their local axes. A common method in robotics and computer graphics to represent rotations in three-dimensional space is using unit quaternions. We will denote these quaternions, their norm and their conjugate as:

$$\begin{aligned}\mathbf{q} &= q_0 + iq_1 + kq_2 + lq_3 \\ \|\mathbf{q}\| &= \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1 \\ \mathbf{q}^H &= q_0 - iq_1 - kq_2 - lq_3\end{aligned}$$

Other forms of expression, Euler Angles for example, suffer from noncontinuous spaces and more time-intensive calculations. Our cost function partly relies on distances between orientations, which requires a robust formulation of this distance. One metric is the magnitude of the rotation that transforms one orientation into the other. We define the mentioned rotation as the unit quaternion \mathbf{q}_{Diff} , which solves the constraint:

$$\mathbf{q}_{\text{Diff}}\mathbf{q}_1 = \mathbf{q}_2 \Leftrightarrow \mathbf{q}_{\text{Diff}} = \mathbf{q}_1^H\mathbf{q}_2 \quad (2.1)$$

We require our magnitude $\|\cdot\|_{\text{mag}}$ to be proportional to the size of the rotation angle θ encoded in \mathbf{q} and direction invariant, while still being normalized:

$$\begin{aligned}\lim_{|\theta| \rightarrow 0} \|\mathbf{q}\|_{\text{mag}} &\rightarrow 0 \\ \lim_{|\theta| \rightarrow \pi} \|\mathbf{q}\|_{\text{mag}} &\rightarrow 1,\end{aligned}$$

This angle is tied to \mathbf{q} by $q_0 = \cos \frac{\theta}{2}$. Accordingly, we formulate $\|\cdot\|_{\text{mag}}$ as

$$\|\mathbf{q}\|_{\text{mag}} = 1 - |q_0| \quad (2.2)$$

which suffices the constraints above. We combine (2.1) and (2.2) to a distance function $d_0 : Q \times Q \rightarrow [0, 1]$:

$$d(\mathbf{q}_1, \mathbf{q}_2) = \|\mathbf{q}_1^H\mathbf{q}_2\|_{\text{mag}}$$

Proof that $\|\cdot\|_{\text{mag}}$ suffices the mathematical properties of a true metric can be found in the appendix.

2.4 Graph-Based Task Representation

The basis of our task representation are *Visual Entity Graphs* (VEGs) introduced in this work [2]. They encode a video that contains optimal task execution by a human expert, into a sequence of T graphs $[G_D^0, G_D^1, \dots, G_D^T]$, with T being the total amount of frames. During the imitation, the agent encodes its environment into a VEG G_I at every time step. Each graph $G^t = \{V^t, \epsilon^t\}$ contains three types of different nodes representing different observations: V_h for the human hand or robot gripper in G_I respectively, V_o for each task-relevant object currently detected, and V_p for visual key points on those objects. They all share the same type of embedding, which consists of the respective entity’s position in 3D (X, Y, Z) coordinates denoted as \mathbf{x}_i .

An edge is tied between any pair of two nodes making the whole graph a clique. Each edge also defines an attention value that denotes whether a corresponding connection is *active*, i.e., considered relevant for the current motion or *inactive*. An edge is *active* if connects to an *anchor object*. At a specific time step, the anchor object is defined as the object that is currently in motion. Or if none are at the moment, the object that will be next. On the one hand, this formulation essentially allows a description of the desired agent trajectory in the object’s frame that he is supposed to manipulate. On the other hand, this encodes a time-dependent focus on a single object whose relations to the environment will primarily impact the cost function. A similar technique is used to describe trajectories in this approach [22], but the concept of attention allows for the sequential manipulation of multiple objects. Edges connected to key point nodes V_p are only active if tied to a node representation V_o corresponding to the object from which they have been sampled.

The same work, which formulated the VEGs, also proposes a cost function used for policy optimization in a Reinforcement Learning (RL) problem. This cost is based on graph-similarity and compares the relative distances between visual entities in the demonstration graph G_D^t with the corresponding relations in the current imitation graph G_I . Using a metric that relies on relative displacements makes the resulting policy invariant against absolute displacements of the whole object set and, to a certain degree, individual variations. This formulation also encodes orientations with distances of key point V_p nodes to their respective object center V_o . Adding edge weights $\omega \in [0, 1]$, that dependent on tied node types, this cost function is postulated as:

$$C(G_I, G_D^t) = \sum_{i=1}^n \sum_{j=i+1}^n \omega(\epsilon_{i,j}^D) \cdot \text{att}(\epsilon_{i,j}^D) \cdot \|(\mathbf{x}_i^D - \mathbf{x}_j^D) - (\mathbf{x}_i^I - \mathbf{x}_j^I)\|,$$

with the attention function $\text{att}(\epsilon) \rightarrow \{0, 1\}$. We improve upon the graph-based task representation and the cost function in the method section.

2.5 Imitation Learning With Model Predictive Control

We base our description of the *Imitation Learning* problem, and its solution on the formulations from this paper [16].

2.5.1 Problem Definition

We formalize our learning problem as a Markov Decision Process (MDP) with finite horizon H . An MDP consists of the state space S , the action space A , the distribution $P(s_{t+1}|s_t, a_t)$, which determines the probability that action a in state s at time t will lead to state s_{t+1} , as well as a per step incurred cost $c(s_t, a_t)$. We are trying to find a policy $\pi(a_t|s_t)$ that optimizes

$$\pi^* = \arg \min_{\pi \in \Pi} \mathbb{E}_{\pi, P} \left[\sum_{t=0}^{H-1} c(s_t, a_t) \right],$$

with Π being the space of all policies. This equation is a classical *Reinforcement Learning* problem [23]. H is much smaller than the total range of the motion, and the resulting policy will be optimal in regard to the next H steps in time.

Imitation learning is introduced to this problem by shaping a task-specific cost function $c(s_t, a_t)$. Akin to *Inverse Reinforcement Learning* approaches [24], we learn a cost function from not necessarily optimal expert demonstrations. We base our task-related cost function on dissimilarity between states observed in the human demonstration and those encountered during imitation at the corresponding phase of the task. This function will be combined with more cost terms to achieve the general side goals of robot manipulations. More about that in the method section.

2.5.2 Solving The Optimization Problem

Solving for a complex globally optimal policy is hard, especially since the task objective c could be sparse or difficult to optimize. Model predictive control can be viewed as a practically motivated strategy that simplifies the overall problem by focusing only on the states the robot encounters online during execution and rapidly re-calculating a “simple” locally optimal policy [16]. It also doesn’t require the exact dynamics of the system, but can work with an approximate dynamics model \tilde{F} .

We are working with an existing implementation of *Model Predictive Path Integral Control* [25] using Covariance Variable Importance Sampling (MPPI) [26] as a basis for our control algorithm. Sampling-based MPC iteratively optimizes simple policy representations such as time-independent Gaussians over open-loop controls with parameters θ_t such that $\Pi_{\theta_t} = \prod_{h=0}^{H-1} \pi_{\theta_t, h}$. Here, θ_t represents the sequence of means $\mu_t = [\mu_{t,0}, \dots, \mu_{t,H-1}]$ and a parameterized covariance Σ at every step along the horizon H .

At every iteration, the optimization proceeds by sampling a batch of N control sequences of length H , $u_n \in [0, N)$, $h \in [0, H)$, from the current distribution, followed by rolling out the approximate dynamics function F using the sampled controls to get a batch of corresponding states $\tilde{x}_{n \in [0, N), h \in [0, H)}$ and costs $\tilde{c}_{n \in [0, N), h \in [0, H)}$. The policy parameters are then updated using a sample-based gradient of the objective function. After $K \geq 1$ optimization iterations, we execute the mean of the resulting distribution. We describe how the distribution is updated next. Consider the function

$$\tilde{C}(x_t, u_t) = \sum_{h=0}^{H-1} \gamma^h \tilde{c}(\tilde{x}_{t,h}, u_{t,h})$$

where $\gamma \in [0, 1]$ is a discount factor that is used to favor immediate rewards. We do not specify a terminal cost $q(\tilde{x}_{t,H}, u_{t,H})$ for the sake of simplification.

A widely used objective function is the exponentiated utility or the risk-seeking objective,

$$L = \mathbb{E}_{\pi_{\theta}, \tilde{P}} \left[\exp \left(\frac{-1}{\beta} \tilde{C}(x_t, u_t) \right) \mid \tilde{x}_0 = x_t \right],$$

where β is a temperature parameter. For this choice of objective, the mean is updated using a sample-based gradient as,

$$\mu_{t,h} = (1 - \alpha_{\mu})\mu_{t-1,h} + \alpha_{\mu} \frac{\sum_{i=1}^N \omega_i u_{t,h}}{\sum_{i=1}^N \omega_i}$$

where α_{μ} and α_{σ} are step-sizes that regularize the current solution to be close to the previous one. With

$$\omega_i = \exp \left(\frac{-1}{\beta} \sum_{h=0}^{H-1} \gamma^h \tilde{c}(\tilde{x}_{t,h}, u_{t,h}) \right)$$

increasing the weight of successful action sequences.

We provide pseudocode for our algorithm here, using the parameters total length T , number of samples N , number of optimization cycles K , horizon length H , initial distribution parameters θ_0 and the approximate system dynamics $\tilde{F}(\tilde{x}_t, u_t) \rightarrow \tilde{x}_{t+1}$. We denote a sampled control sequence batch $(u_{0...N,0}, u_{0...N,1}, \dots, u_{0...N,H-1})$ as \mathbf{u}_t , a computed batch of state sequences $(x_{0...N,0}, x_{0...N,1}, \dots, x_{0...N,H-1})$ as \mathbf{x}_t and resulting batch of cost sequences $(c_{0...N,0}, c_{0...N,1}, \dots, c_{0...N,H-1})$ as \mathbf{c}_t . Since a distinct distribution is defined for each step within the horizon we define a sequence of distribution parameters $((\mu_0, \Sigma), (\mu_1, \Sigma), \dots, (\mu_{H-1}, \Sigma))$ as θ_t .

```

for t = 1 ... T do
   $x_0 \leftarrow \text{GET\_CURRENT\_STATE}()$ 
   $\pi_{\theta_t} \leftarrow \text{SHIFT}(\theta_{t-1})$ 
  for i = 1 ... K do
     $\mathbf{u}_t \leftarrow \text{SAMPLE\_CONTROLS}(\pi_{\theta_t}, H, N)$ 
     $\mathbf{x}_t, \mathbf{c}_t \leftarrow \text{COMPUTE\_ROLLOUTS}(F, H, N, x_0, \mathbf{u}_t)$ 
     $\theta_t \leftarrow \text{UPDATE\_DISTRIBUTION}(\mathbf{c}_t, \mathbf{u}_t)$ 
  end for
   $\text{EXECUTE}(\mu_0(\theta_t))$ 
end for

```

Algorithm 1: Sampling-Based MPC Algorithm

3 Method

3.1 Improvements On The Graph-Based Task Representation

We propose an improvement upon the task representation described in the foundation section. We also use a VEG sequence to encode a given video demonstration. However, we made significant changes to the embeddings of the involved VEGs and to the inferring of underlying data. We only consider two types of nodes for each graph, with V_h representing the human hand or robot gripper, while V_o represents each task-relevant object in the demonstration. Additional to the entity’s 3D (X, Y, Z) position, we also include its orientation directly into each node embedding, expressed as a unit quaternion. This formulation simplifies data collection since we do not rely on time-consuming pre-trained keypoint detectors, which also reintroduce issues that we explicitly try to avoid with an object-centered task representation. Furthermore, the properties of quaternions allow for a much more intuitive and reliable metric for the distance measurement between two orientations, as shown in the foundation section. Finally, having a streamlined graph embedding enables us to make hand-crafted modifications, which influence the optimization goal during critical motion phases.

We use the edges from the original postulation except for the tied weights. We drop these since they no longer serve a purpose due to our reduced node variety.

We also keep the *attention* function and its purpose, but with a changed definition of an *anchor* object. We consider the anchor to be the object currently manipulated by the human expert or if none are at the moment, the object that will be next. Consequently, we infer the entity of the greatest importance by proximity to the human hand, while the original approach derives relevance based on motion saliency as postulated here [11]. Our approach is supposed to be more robust against estimation errors and occlusion.

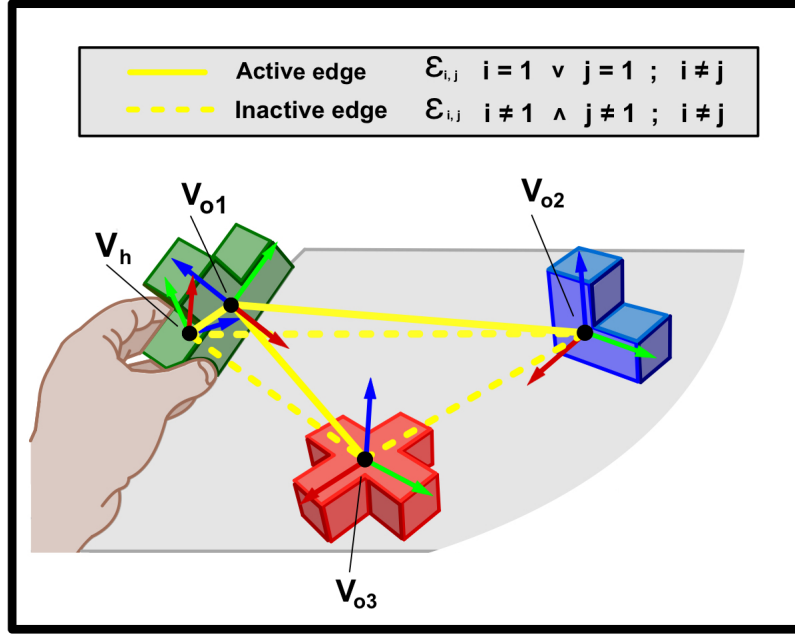


Figure 3.1: Example of a visual entity graph constructed during step t of a task demonstration. Depicted is the representation of the human hand V_h and 3 nodes V_{o1} , V_{o2} , V_{o3} corresponding to relevant objects. The *green piece with a T-shape* is currently considered to be the *anchor* because the human expert manipulates it. Consequently, all edges connected to it are considered active in this VEG.

To derive a task-related cost function, we use graph similarity as a basis. We require a one-to-one correspondence between nodes in both graphs, assuming a set of n a priori labeled objects in video. These entities must be reliably detected during the imitation and must be represented in every graph. Data from single steps in the demonstration, where specific entities are not detected or show a significant amount of pose estimation error, is interpolated to ensure the continuous presence of entities. If an object can not be properly detected during the execution, we use its last known pose as its embedding.

The modified graph structure allows the cost function to be split into a component dependent on positions:

$$C_P(G_I, G_D^t) = \sum_{i=1}^n \sum_{j=i+1}^n \text{att}(\epsilon_{i,j}^{D,t}) \cdot \|(\mathbf{x}_i^{D,t} - \mathbf{x}_j^{D,t}) - (\mathbf{x}_i^I - \mathbf{x}_j^I)\|$$

We also require another component that depends on orientation:

$$C_O(G_I, G_D^t) = \sum_{i=1}^n \sum_{j=i+1}^n \text{att}(\epsilon_{i,j}^{D,t}) \cdot d_O \left((\mathbf{q}_i^{D,t})^H \mathbf{q}_j^{D,t}, (\mathbf{q}_i^I)^H \mathbf{q}_j^I \right),$$

which are linearly combined with the weights $\omega_P, \omega_O \in \mathbb{R}^+$ to encode importance of the respective aspect and to bridge their gap in scale. We formulate the resulting cost function as:

$$C(G_I, G_D^t) = \omega_P C_P(G_I, G_D^t) + \omega_O C_O(G_I, G_D^t)$$

We use this function as an incentive for online policy optimization. Its input consists of hypothetical imitation graphs, which the robot action could induce within the present environment and the current optimization goal VEG from the encoded demonstration.

3.1.1 Human To Robot Domain Transfer

Imitating human expert demonstrations is limited by mismatches in dynamics, view-point, and embodiment from teacher to robot [1]. Because we extract object and hand poses from visual data first and before learning from them, our task representation is invariant against changing viewing angles. However, differences in the embodiment, dynamics, and state and action spaces remain significant. For example, a human hand can grasp an object from every angle and at every point, due to its freely configurable fingers. A robot with a simple gripper made of two prismatic joints has to find a point of attack that fits its hardware constraint. The resulting grasping pose might significantly diverge from the human demonstration.

There are multiple approaches to overcome these issues in RL, such as learning a direct state to state mapping or learning a correspondence model between observations and robot actions [27]. We resolved to solve the dynamics problem by completely dropping observed actions from the task representation. Similar to this work [6], we incentivize the imitation agent to recreate the state sequence observed in the human demonstration.

This approach, in turn, requires a one-to-one mapping between demonstrator and imitator state space.

We achieve this in two ways. First of all, we break down observations of the human hand into a 6D pose and a grasping state per time step, and since we only have access to the positions of specific landmarks, we have to design a corresponding pose. The goal is to enable the robot gripper to follow the human trajectory and imitate its grasping approach as much as possible.

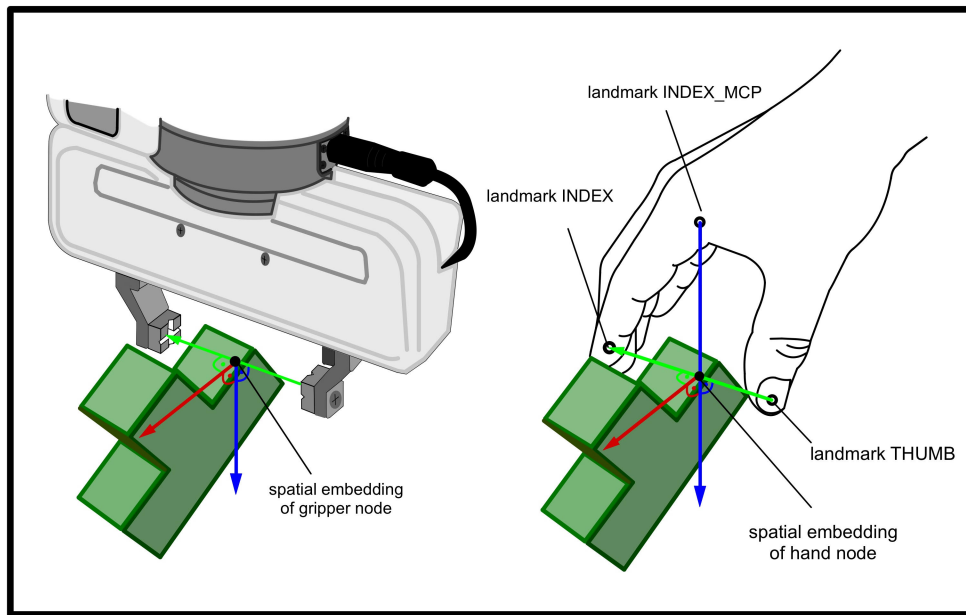


Figure 3.2: Extracted human hand pose that serves as an optimization target for the panda gripper (*left*). This pose consists of a centroid represented by the *black dot* and x-axis *red*, y-axis *green*, z-axis *blue*. This inferred pose is determined by the detected positions of the 3 annotated landmarks *INDEX*, *THUMB*, and *INDEX_MCP*

Our method has been developed with the gripper of the Panda robot in mind but applies to many standard gripper models.

We use the cartesian coordinates of three specific points on the human hand as a basis for our pose derivation. We estimate their positions per frame as described in the foundation section.

We denote their (X Y Z) coordinates as \mathbf{I}_p for the *INDEX* mark, \mathbf{T}_p for the *THUMB* mark, and \mathbf{M}_p *INDEX_MCP* mark, as shown in figure 3.2. The local \mathbf{x} -axis of the human hand is determined by the direction of the other axis and the right-hand rule:

$$\mathbf{x} = \mathbf{y} \times \mathbf{z}$$

We choose the local \mathbf{y} -axis as the connecting vector between fingertip and thumb tip:

$$\mathbf{y} = \mathbf{I}_p - \mathbf{T}_p$$

The \mathbf{z} -axis is constructed in a such a way that it runs orthogonal to \mathbf{y} while having a common interception point. It is also required that the point *INDEX_MCP* lies on \mathbf{z} , which together can be expressed as:

$$s = 1 - \frac{\mathbf{y} \cdot (\mathbf{M}_p - \mathbf{T}_p)}{\|\mathbf{y}\|^2} \quad \mathbf{z} = (1 - s) \cdot \mathbf{y} - (\mathbf{M}_p - \mathbf{T}_p)$$

Afterwards we have to compute the quaternion representation of the calculated hand orientation, which is defined by the rotation matrix constructed with the local axes:

$$R = \begin{pmatrix} \frac{\mathbf{x}}{\|\mathbf{x}\|} & \frac{\mathbf{y}}{\|\mathbf{y}\|} & \frac{\mathbf{z}}{\|\mathbf{z}\|} \end{pmatrix}$$

The origin of this axes system, i.e the inferred hand position p is chosen to be the mean of the detected thumb and index tip positions:

$$\mathbf{p} = \frac{\mathbf{I}_p + \mathbf{T}_p}{2}$$

Note that this method assumes that a plane can be spanned between all 3 landmarks and that the *INDEX_MCP* point is located between *THUMB* and *INDEX*. Due to estimation errors, these conditions are violated sometimes. The hand orientation for the respective time steps is interpolated with surrounding valid values as described in the data collection section. The resulting pose is used as the embedding of the hand nodes V_h .

The other way we bridge the dynamics gap is by making hand-crafted modifications to the inferred grasping approaches. While the described pose extraction allows the robot to imitate the human hand movements to a reasonable degree, it can not compensate for the flexibility of the human hand. Since a static gripper can only pick up objects from very specific angles and high precision is required to achieve a proper grasp, we can not directly follow the approach taken by the human demonstrator. Fig 3.2 shows an edge case where direct imitation would be possible, but this can not be assumed because the human teacher may choose a different way to grasp an object.

The solution is applying hand-crafted modifications to graph embeddings around detected grasping events t_{grasp} , i. e., the grasp state switches from open to closed. These modifications allow us to rely on precise relative trajectories during critical moments while keeping the same optimization method.

For example, we define an optimal grasping angle for each object. We formulate this angle as an alignment of $\mathbf{z}_{\text{gripper}}$ with $\mathbf{y}_{\text{object}}$ and $\mathbf{y}_{\text{gripper}}$ with $\mathbf{z}_{\text{object}}$ assuming optimal placement of the corresponding Aruco marker. To enforce this configuration as an optimization target, we overwrite the encoded orientation of hand nodes that belong to demonstration graphs near t_{grasp} . We chose the grasped object's center as an optimal point of attack, and the position of the hand node is set to it accordingly.

We introduce complex support for grasping and releasing objects by splitting such motions into multiple distinct phases, where demonstration graph embeddings and control parameters are modified. In this way, high-level action planning is introduced into our control algorithm. We make use of the following phases that modify a range of VEGs each, which we denote as r_{phase} respectively:

Moving To Pre-Grasp Position: We modify VEGs in the interval $[t_{\text{grasp}} - r_{\text{pre_grasp}} - r_{\text{move_to_grasp}}, t_{\text{grasp}} - r_{\text{move_to_grasp}} - 1]$ for this phase. The position of the hand node is set to a specified height $h_{\text{pre_grasp}}$ above the position of the anchor object's node. The hand node's orientation is set to the optimal grasping angle as described above. We do not set a new optimization target until the robot has reached the desired position within a certain tolerance.

Moving To Grasp Position: We modify VEGs in the interval $[t_{\text{grasp}} - r_{\text{move_to_grasp}}, t_{\text{grasp}} - 1]$ for this phase. The position of the hand node is set to the object center while its orientation is kept as before. We do not set a new optimization target until the robot has reached the desired position within a certain tolerance.

Closing The Grasp: We modify the VEG at t_{grasp} for this phase. The Position and orientation of the hand node stay the same. The endeffector movement is disabled. We do not continue until the robot's gripper has closed below a certain width threshold, which is encouraged by the regular cost function without further modification.

Lifting Post-Grasp: We modify VEGs in the interval $[t_{\text{grasp}} + 1, t_{\text{grasp}} + r_{\text{lift_grasp}}]$ for this phase. To avoid shifting the held object and collision with the surface we set its spatial embedding to $h_{\text{pre_grasp}}$ above its previous position.

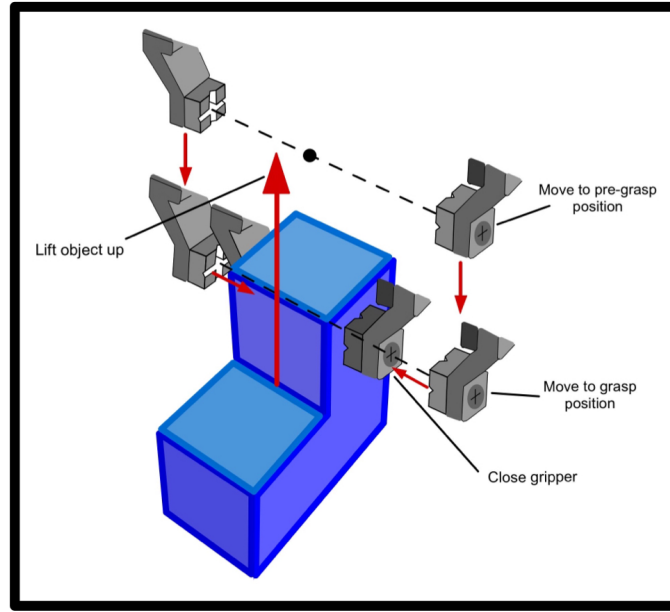


Figure 3.3: Visualization of optimal gripper poses in the relation to the object of interest during the different grasping phases. These phases consist of i) moving to the pre-grasp position, ii) continuing to the final grasping position, iii) closing the gripper, and finally iv) lifting the object vertically.

Moving To Pre-Release pose: We modify VEGs in the interval $[t_{\text{release}} - r_{\text{pre_release}}, t_{\text{release}} - 1]$ for this phase. Orientation of the current anchor object node is set to the optimal release angle. Its position remains the same. We consider the optimal release angle the closest orientation, where one of the local object axes is aligned with the normal vector of the workplace surface.

Open gripper: We modify the VEG at time step t_{release} for this phase. We set the Position and orientation of the hand node as in the previous phase. The endeffector movement is disabled. We do not progress to the next step until the gripper has opened above a certain width threshold, which is encouraged by the regular cost function without further modification.

Lifting Post-Release: We modify VEGs in the interval $[t_{\text{release}} + 1, t_{\text{release}} + r_{\text{lift_release}}]$ for this phase. Any rotation of the endeffector is disabled. To avoid toppling over the released object, we set the position of the hand node to $h_{\text{post_release}}$ above the release position.

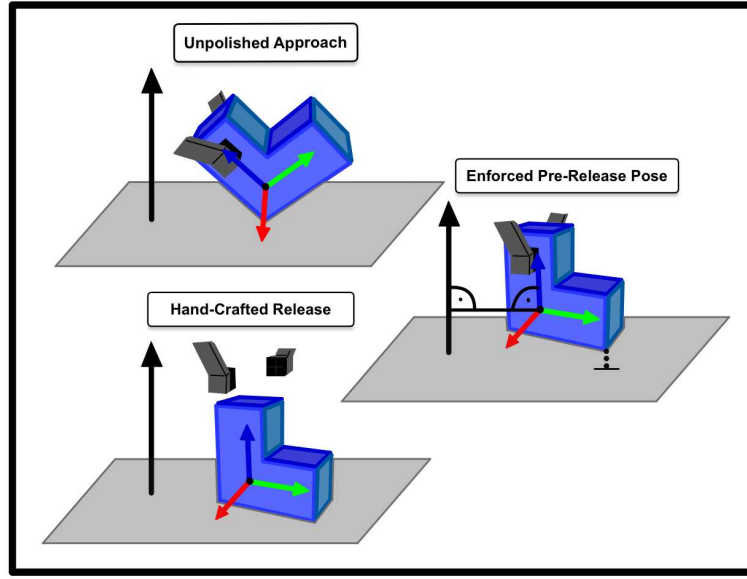


Figure 3.4: Visualization of hand-crafted release phases. These phases consist of following the unpolished demonstrated trajectory (*top*), manipulating the object pose in a way that aligns one of its axes with the table normal vector (*middle*), opening the gripper, and releasing the object (*bottom*)

Following the trajectory of the human hand, instead of only recreating the object trajectories, is serving the purpose of guiding the robot into a pose where it can smoothly transition into those hand-crafted approaches. Also, overwriting VEGs in a reasonable big range surrounding grasping and releasing events assures that we continue to imitate the human motion only when having a safe distance to the object involved in manipulation.

3.2 Data Collection

3.2.1 Capturing A Demonstration

Creating demonstration data consists of recording a video of optimal task execution. However, we added some more functionality to this process for debugging purposes and the automatic removing of data, which is not actually related to the task. To achieve the first goal, we save demonstration videos with annotations that display the results of our pose estimation, i.e., detected hand landmarks and object axes. To facilitate the cutting of irrelevant data, we define a starting point and an endpoint in the recording. Video frames outside this range are not encoded into the graph sequence. The demonstration start is specified as the time step, where all fingers can be cleanly detected for the first time. The content end is set to the frame that contains the last valid detection. This formulation eliminates the need for a precise starting and stopping of the recording by allowing the user to physically cover the camera until he has reached a good initial position. The teacher can also terminate the recording in the same way. This method made any manual cutting of the raw data unnecessary during our experiments.

The graph embeddings are stored in camera coordinates and have to be transformed with an externally provided robot camera transformation before an imitation occurs. This approach allows the demonstration to be captured outside the robot's workplace.

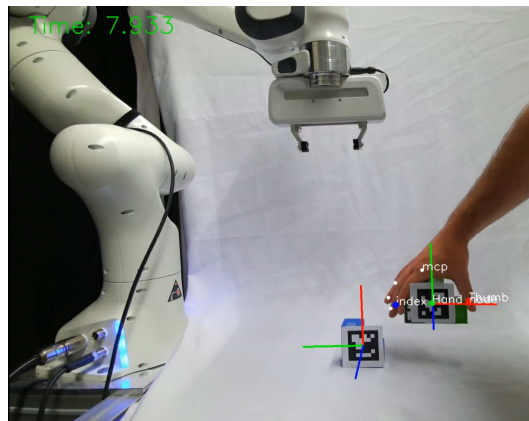


Figure 3.5: Single frame of a captured demonstration video. The results of our object estimation and the outputs of the hand detection framework are annotated

3.2.2 Filtering And Interpolation

Since pose estimations solely based on visual data are quite prone to errors, we have introduced multiple stages of data filters, which are applied to object and hand landmark trajectories.

We use the following techniques to filter spacial data. First of all, we deal with a type of error that is prominent in our hand detection outputs, i.e., groups of outlier points with very large positive values. We chose an outlier filter to eliminate these, as described in the “Novelty and Outlier Detection” section of *scikit-learn* [28] since common methods like average filters or B-Splines are more suitable for errors that are Gaussian in nature or consist of single spikes. Our outlier filter is applied on a set N of neighboring data points. We choose a subset S from this group, with μ_S being their mean, i.e., the center of this subgroup. This selection is determined to contain minimal variance:

$$S = \arg \min_S \sum (\mu_S - s)^2, S \subset N$$

Afterwards, we compute the average \bar{D} of all mahalanobis distances $D(s, \mu_S)$ [mahlab source] in S regarding its center. Every point n in N that does not suffice

$$D(n, \mu_S) \leq f \cdot \bar{D}, f \in \mathbb{R}^+$$

is considered to be an outlier and will be replaced by interpolated data later.

To remove regular Gaussian noise and smooth the trajectories, we apply a median filter combined with a Savitzky-Golay filter [source savgol] afterward.

We purify collected orientation data by using a modified average filter. Our goal is to recompute each rotation trajectory to reduce sudden jumps caused by estimation errors without diminishing ground truth angular velocities. Since we represent the orientation for each entity by quaternions, we can use spherical linear interpolation (SLERP)[29]. SLERP is a widespread technique in computer graphics to generate smooth animations by computing intermediate steps between distant orientations. We use it similarly to calculate a point $\tilde{\mathbf{q}}_t$ on the unit quaternion hypersphere that lies on the shortest path between the previous orientation \mathbf{q}_{t-1} and the current detection \mathbf{q}_t :

$$\tilde{\mathbf{q}}_t = \text{SLERP}(\mathbf{q}_{t-1}, \mathbf{q}_t, \omega),$$

with the parameter $\omega \in [0, 1]$ determining how close the result is to the target quaternion \mathbf{q}_t . We achieved good results by making ω anti-proportional to the actual distance between a given pair of orientations.

Accordingly, we define a temperature parameter $\beta \in \mathbb{R}$ and calculate ω as:

$$\omega = 1 - \exp\left(-\frac{1}{\beta} d_o(\mathbf{q}_{t-1}, \mathbf{q}_t)\right),$$

This formulation works essentially like a weighted mean that reduces large jerks and keeps true angular velocities. Note that we do need not apply any complex transformations on our demonstration data, such as Whitening or PCA.

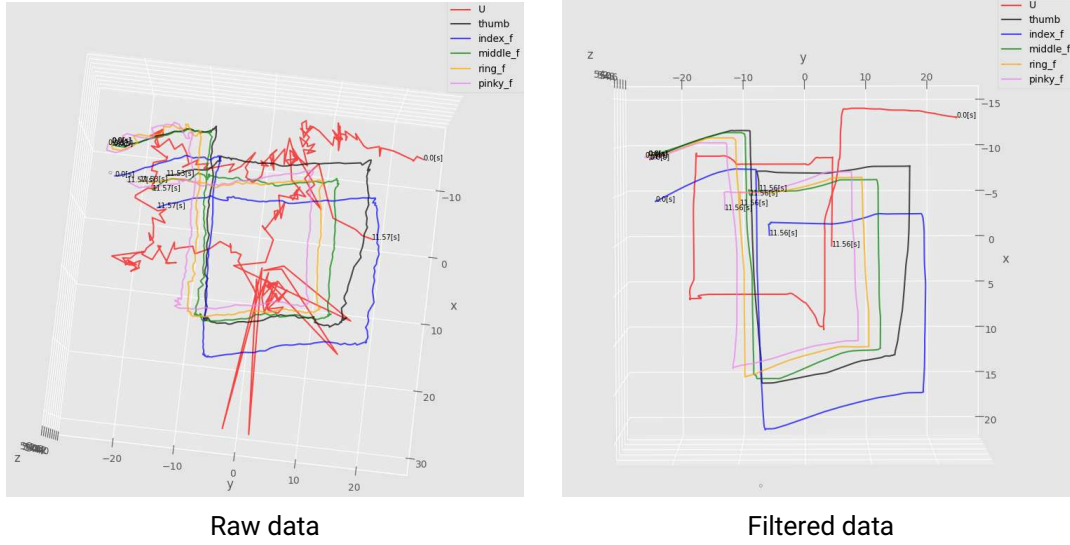


Figure 3.6: Visualization of raw and filtered data differences. We compare inferred spatial data from an object labeled *U*, *thumb*, *index finger*, *middle finger*, *ring finger*, and *pinky finger*

Discarded or missing data points are interpolated in the final data processing step. For spacial data, we use cubic splines, which are a well-known method to get smooth trajectories. To compute missing orientations we once again utilize SLERP by calculating intermediate rotations between valid data points.

3.3 Control Algorithm

3.3.1 State And Action Design

We define our state $\tilde{x}_t \in \mathbb{R}^N$ as the concatenation of the endeffector position \mathbf{x}_{ee} , the endeffector orientation as a quaternion \mathbf{q}_{ee} , its gripper width g and a similar 6D pose $[\mathbf{x}_o, \mathbf{q}_o]$ for every involved object. With N objects this results in the vector $[\mathbf{x}_{ee}, \mathbf{q}_{ee}, g, [\mathbf{x}_{o1}, \mathbf{q}_{o1}], \dots, [\mathbf{x}_{on}, \mathbf{q}_{on}]]$ of the length $N = 3 + 4 + 1 + n \cdot (3 + 4)$.

Let the actions $\mathbf{u}_t \in \mathbb{R}^7$ be $[\Delta x_{ee}, \Delta y_{ee}, \Delta z_{ee}, \Delta yaw_{ee}, \Delta pitch_{ee}, \Delta roll_{ee}, \Delta g]$, which are displacements of the endeffector as well as a change in the gripper width Δg . We specify the desired rotation of the endeffector in Euler angles because they are a lot easier to sample than unit quaternions.

Using this design allows us to forgo any calculation of the robot kinematics or kinetics, which will be done by the robot controller when executing the displacement commands. The construction of an imitation graph is also straightforward with this state representation because it contains the embedding of all involved nodes already.

3.3.2 Approximate Dynamics Function

We can make do with a very simplified model of the environment and object interactions without taking into account any friction, weights, robot dynamics, joint singularities, or collisions by relying on the error-correcting property of the MPC Paradigm, which allows for computationally cheap re-optimizing of the policy at every time step. This is reasonable for our work because we focus on pick- and place tasks and displacement controls are executed by a robust hardware controller. Fringe cases such as collision, where our proposed dynamics would break down, are also avoided by introducing incentives into our MPC cost function, Smaller uncertainties are compensated by the mentioned properties of this algorithm.

Our dynamics function consists of updating the pose of the endeffector according to the displacement commands, while also applying the same translation and rotation to the pose of the currently held object. If any object is considered grasped is determined by thresholding the robot gripper width, which is also contained in our state. What object is held, is derived from the anchor object in the current target demonstration VEG.

This computation can be done in parallel for whole state and control batches, which improves computational efficiency significantly. However, F has to be applied sequentially for every step along the horizon. Pseudocode for this function is given below:

```

INPUT  $\tilde{x}_t, \mathbf{u}_t, \mathbf{I}$  : anchor_obj_idx
PARAMETER  $g_{th}$  : grasping_treshhold,  $g_{min}$  : minimal gripper width,  $g_{max}$ : maximal gripper width

 $\mathbf{x}_{ee,t+1} = \mathbf{x}_{ee,t} + [\Delta x_{ee}, \Delta y_{ee}, \Delta z_{ee}]$ 
 $\mathbf{q}_{ee,t+1} = \text{QUATERNION}(\Delta yaw_{ee}, \Delta pitch_{ee}, \Delta roll_{ee}) \cdot \mathbf{q}_{ee,t}$ 
 $g_{t+1} = \text{MAX}(\text{MIN}(g_t + \Delta g, g_{min}), g_{max})$ 

for n in N do
     $[\mathbf{x}_{on,t+1}, \mathbf{q}_{on,t+1}] = [\mathbf{x}_{on,t}, \mathbf{q}_{on,t}]$ 
end for

if  $g_t \leq g_{th}$  then
     $\mathbf{x}_{oI,t+1} += \mathbf{x}_{oI,t} + [\Delta x_{ee}, \Delta y_{ee}, \Delta z_{ee}]$ 
     $\mathbf{q}_{oI,t+1} = \text{QUATERNION}(\Delta yaw_{ee}, \Delta pitch_{ee}, \Delta roll_{ee}) \cdot \mathbf{q}_{oI,t}$ 
end if

return  $[\mathbf{x}_{ee,t+1}, \mathbf{q}_{ee,t+1}, g_{t+1}, [\mathbf{x}_{o1,t+1}, \mathbf{q}_{o1,t+1}], \dots, [\mathbf{x}_{oN,t+1}, \mathbf{q}_{oN,t+1}]]$ 

```

Algorithm 2: Approximate dynamics function F

3.3.3 Cost Function

Our cost function \tilde{c} is a linear combination of multiple cost functions that induce different high level behaviours each:

$$\tilde{c}(\tilde{x}_{t,h}, u_{t,h}) = \tilde{c}_{\text{task}}(\tilde{x}_{t,h}, u_{t,h}) + \sum \tilde{c}_{\text{side goal},i}(\tilde{x}_{t,h}, u_{t,h})$$

These components either serve a task related goal or a specific auxiliary requirement. Their calculation is explained in the next section.

Task-Related Cost

This cost enforces task completion by inducing relative object and manipulator trajectories as well as grasping according to observations from the optimal execution. It consists of the step depended on graph-similarity-based component $C(G_I, G_D^t)$ previously proposed and a function $c_{\text{grasp}}(\tilde{u}_{t,h})$ that compares the change of the robot gripper width with the human grasping intention at the corresponding point in the demonstration:

$$\tilde{c}_{\text{task}}(\tilde{x}_{t,h}, u_{t,h}) = C(G_I(\tilde{x}_{t,h}), G_D^{t,h}) + c_{\text{grasp}}(\Delta g_{t,h}),$$

with the VEG G_I being generated from the current state $\tilde{x}_{t,h}$, i.e. the agent's observations, while $G_D^{t,h}$ represents the optimal state at step $t + h$. We have extrapolated this optimization target from data extracted from the expert demonstration. We ensure time alignment of imitation to the human motion by tying t to a global optimization step, which is increased every time a command is issued to the robot. This progression is modified during the previously mentioned high-level motion phases.

The grasp cost $c_{\text{grasp}}(\tilde{u}_{t,h})$ ensures that the robot is punished for having a different grasping intention than the expert at the corresponding time step. We formulate it as:

$$c_{\text{grasp}}(\Delta g_{t,h}) = \begin{cases} \omega_{\text{grasp}}, & \delta(t+h) \cdot \Delta g_{t,h} \geq 0, \\ 0, & \text{else} \end{cases}$$

$\delta(t)$ is a function being evaluated for every time step t of the demonstration so that is 1 if the expert is observed to be grasping at t and -1 otherwise. We base this cost on the change of gripper width instead of its current state to keep it steady at its maximum or minimum value respectively.

We dismissed first using thresholding to determine whether the robot manipulator is open or closed and afterward comparing this state to the human grasp, as we did in our dynamics. We made the experience that this would encourage small oscillations in the gripper span within the respective range set by the threshold, which in turn made the held object drop sometimes or lead to toppling it over during a release attempt.

Side Goal-Related Cost

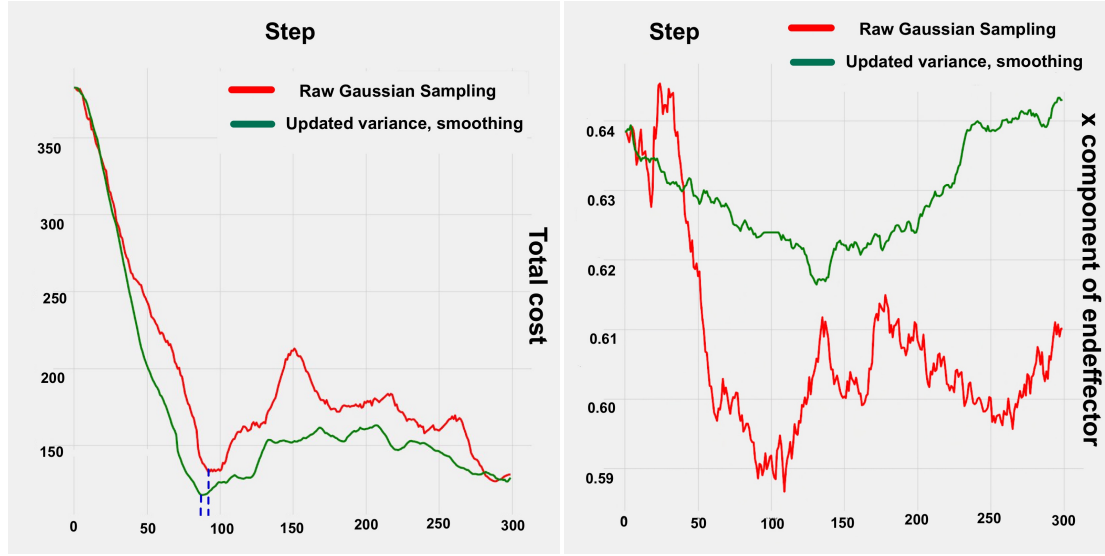
In our setting we were able to achieve good results with a single auxiliary requirement enforced by our costs, namely avoiding floor collision. We ensure that the robot manipulator, or the held object respectively, always stays above a parameterized height h_{\min} relative to the world origin. We formulate the corresponding cost as:

$$c_{\text{coll}}(\tilde{x}_{t,h}) = \begin{cases} \omega_{\text{coll}}, & g > g_{\text{th}} \wedge z_{\text{ee}} \leq h_{\min} \\ \omega_{\text{coll}}, & g \leq g_{\text{th}} \wedge z_{\text{o}} \leq h_{\min} \\ 0, & \text{else} \end{cases}$$

z_{ee} and z_{o} are representing the z-components of the robot gripper or carried object respectively.

3.3.4 Sampling Strategy For Control Sequences

We further expand on the basic sampling strategy implemented in the MPPI framework. The method used to generate controls from the optimized Gaussian policy has to enforce fast convergence as well as smooth trajectories and furthermore ensure a good hot start for the next optimization cycle. We continue taking the mean of the final Gaussian distribution as the next action as in the basic implementation, but apply a Savgol filter along the horizon of each dimension to smooth the sampled trajectories. Since this framework, as most MPPIs on real systems [Mppi src1] [Mppi src2], does not employ any kind of covariance update for the sampling distribution, commands in dimensions with low impact on the total error show large oscillations around zero. To reduce the resulting jitter of the endeffector we additionally make use of a median filter.



Plot of total cost at corresponding time stepPlot of x-axis component of the robot endeffector during execution. Reaching of initial position is marked in blue

Figure 3.7: Visualization of sampling strategy effects on the overall cost as well as end-effector trajectory. Values in green correspond to modified action sampling, while the value in red belongs to execution with raw Gaussian sampling. We can reduce total cost during execution significantly (*left graph*) and get much smoother trajectories at the same time (*right graph*).

We also introduce a basic covariance update for the spatial displacements by distributing the combined initially specified covariances Σ_0 across the corresponding action dimensions according to their respective share in the total positional task-related error.

$$\Sigma_{\Delta x_{ee}} = \frac{C_{P,x}}{C_P} \cdot 3 \Sigma_0, \quad \Sigma_{\Delta y_{ee}} = \frac{C_{P,y}}{C_P} \cdot 3 \Sigma_0, \quad \Sigma_{\Delta z_{ee}} = \frac{C_{P,z}}{C_P} \cdot 3 \Sigma_0$$

This increases exploring in directions with a large distance to the optimal position while reducing movement in well-met axes. We have only introduced this update for the spatial dimensions because we can directly infer their individual impact on the total task-related error.

4 Experiments

4.1 Task Domain

We have chosen to demonstrate the capabilities of our framework within a task domain that incorporates many challenges from robot manipulation. We chose to let our robot play the social game “Ubongo”, which basically consists of solving geometric 3D puzzles by finding the correct arrangement of building blocks. These pieces feature a fairly complex shape. This game combines pick- and place with peg-in hole manipulation tasks while solving each puzzle even requires high-level action planning. It is also an interesting field for computer vision applications such as pose estimation, due to having colorful predefined objects, but also having to deal with a lot of occlusions.

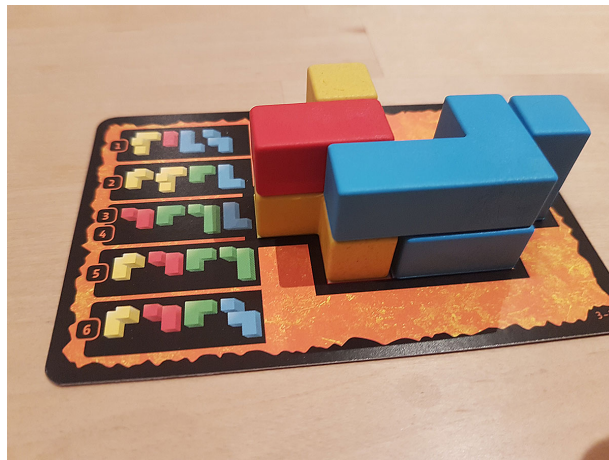


Figure 4.1: Example of a solved “Ubongo” puzzle made up of complex individually colored pieces

We have decided to focus on the pick and place aspect of this game since it is more appropriate for our achieved level of precision and is reasonably well approximated by our proposed system dynamics.

We faced the problem that the original “Ubungo” blocks were too small for the Panda to manipulate. That’s why we first created CAD models of every object, up-scaled these, and used a 3D printer with colored filaments to print bigger versions. Each piece is structurally made of an arrangement of cubes, which we re-scaled to have a side length of 3.5 cm. We also ensured easy manipulability by keeping them at a low weight, which requires a design with minimal internal support structure.

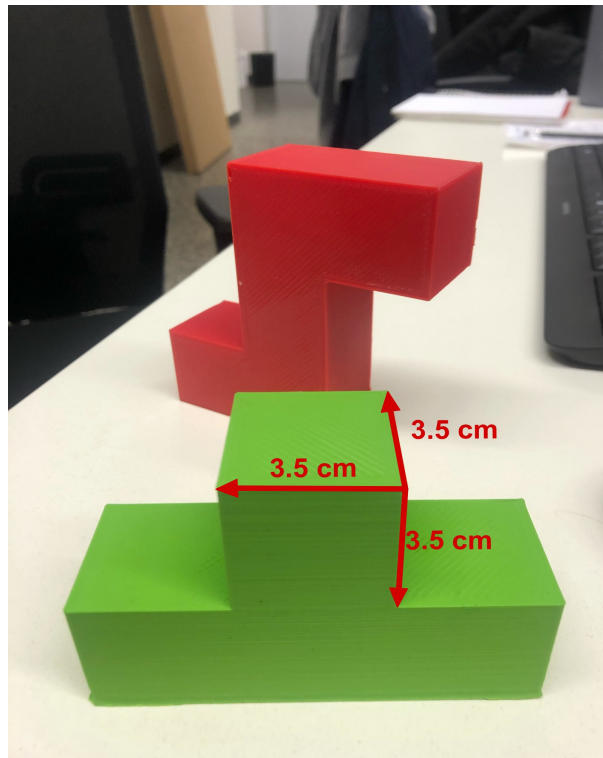


Figure 4.2: Photo of a manually printed “Ubungo” piece with increased scale. The new dimensions per cube are 3.5 cm × 3.5 cm × 3.5 cm as annotated

4.2 Implementation Details

We have tested our work in a Pybullet [30] simulation using a model of the Franka robot *Panda*. We have also created models for each “Ubongo” piece involved in our tests. Demonstration videos are captured in the real world and their graph representations are pre-computed once. Our online policy optimization takes approximately 0.5 seconds to calculate a single command but has not been fully optimized for compatibility with GPU yet. A complete list of hyperparameters for every aspect of our pipeline can be found in the appendix. This set of parameters does not have to be set for individual experiments specifically but has achieved good results across different tasks in the pick and place domain.

4.3 Evaluation On Pick And Place Tasks

Through our experiments, we aim to analyze whether our framework can successfully imitate a task execution in the domain of pick and place if provided solely with an RGBD video of a human demonstration. Furthermore, we want to show that our method generalizes the observed skill performance to a new environment, adapts to varying initial configurations of involved objects, and can compensate for the different embodiment of the human teacher, especially during grasping. The completion of the task has to suffice task-unrelated side goals, as mentioned in the previous section.

As a direct measurement of our policy optimization quality, we plot the value of our formulated cost function for each system state reached during the imitation. We also compare different parts of the human motion and corresponding robot approaches to show that we reproduce demonstrated trajectories as close as possible, while also adapting to the robot’s hardware and its auxiliary requirements.

To this end, we first analyze a demonstration video of a simple pick and place task involving only 2 “Ubongo” pieces, since the resulting cost function is quite intuitive. This scenario also allows for the biggest variation in the initial configuration of the scene, which includes the starting pose of the human hand i.e the robot gripper, absolute object poses as well as relative object poses. The given task consists of picking up one of the objects, moving it with a specific trajectory, and placing it back down. This is done relative to the other static object. We consider the imitation done as soon as our global optimization step reaches the last VEG in the extracted task representation.

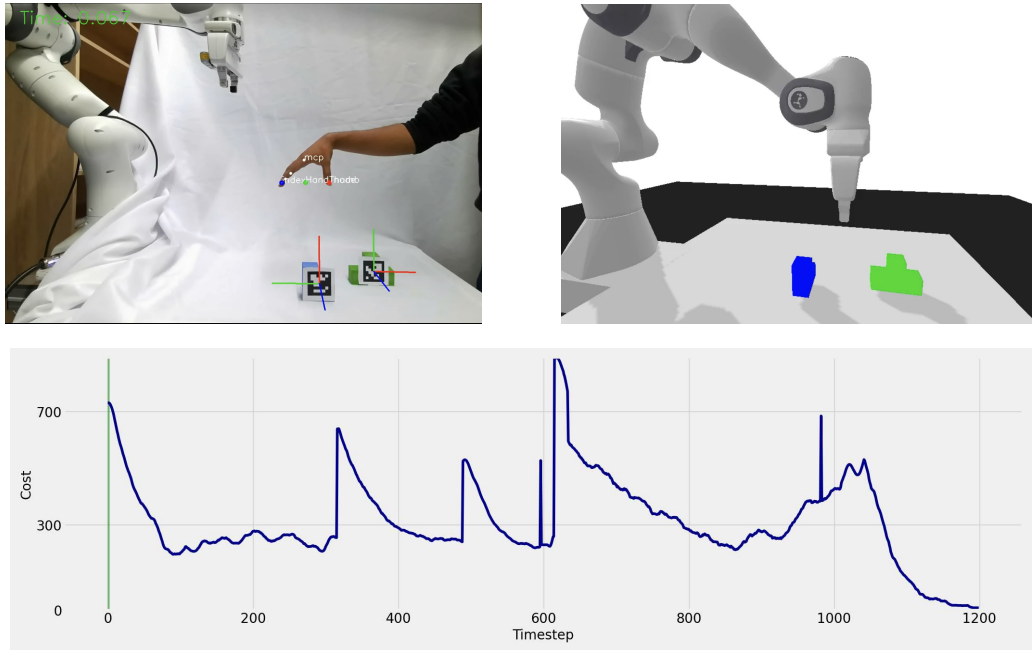


Figure 4.3: Initial Configurations: Corresponding time steps are marked in *green*

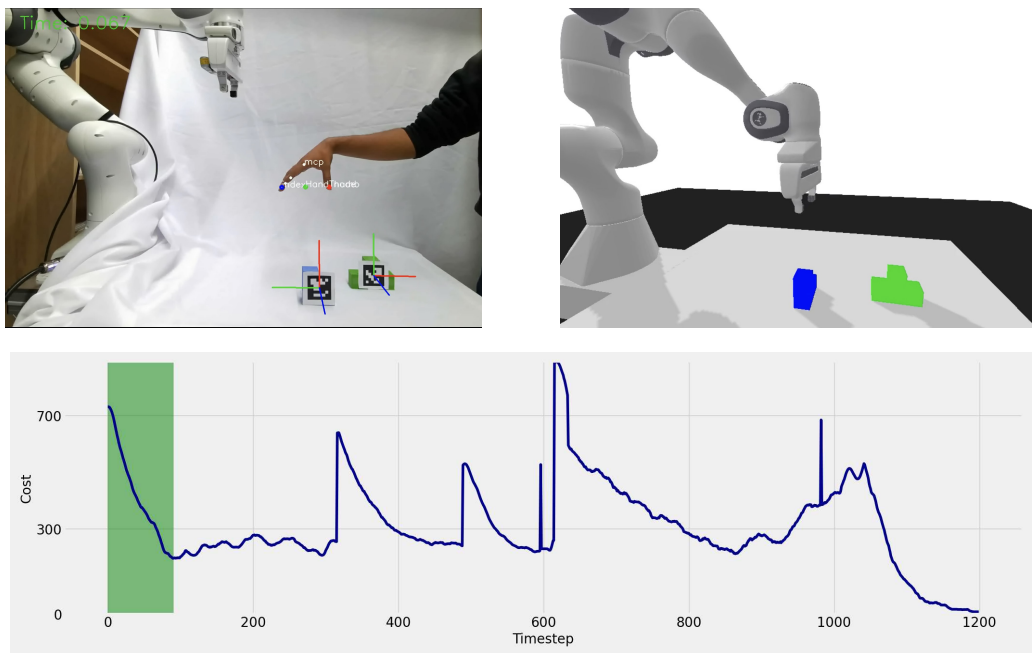


Figure 4.4: Reaching Initial Human Pose

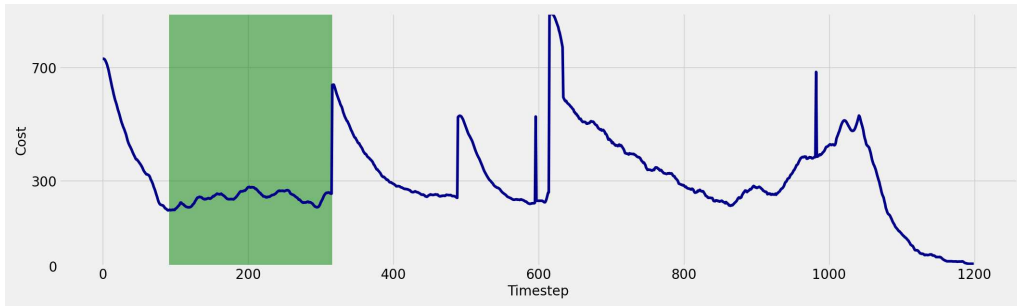
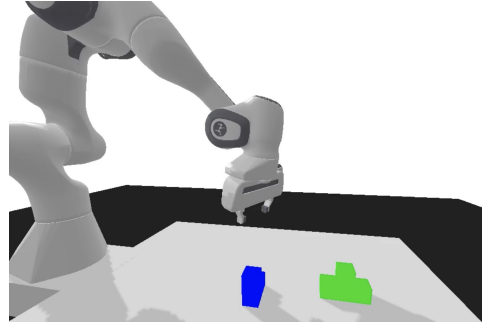
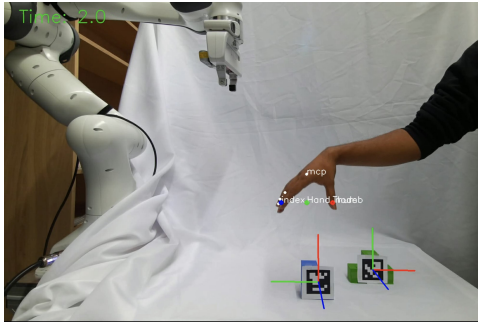


Figure 4.5: Directly Imitating Demonstrated Task Execution

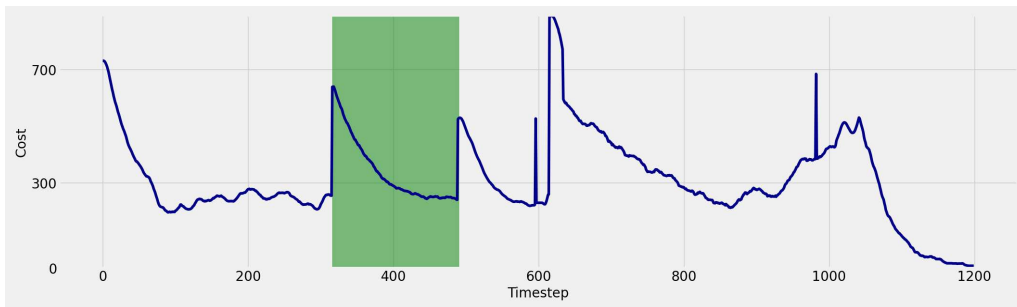
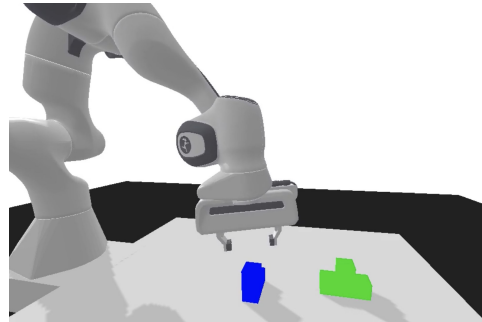
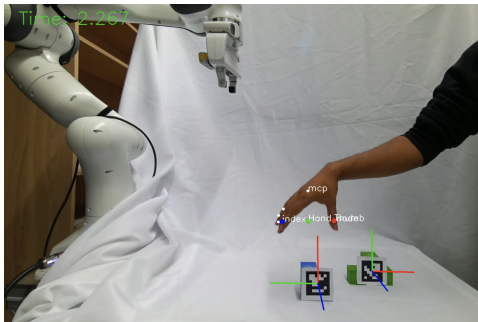


Figure 4.6: Reaching The Pre-Grasp Positoin

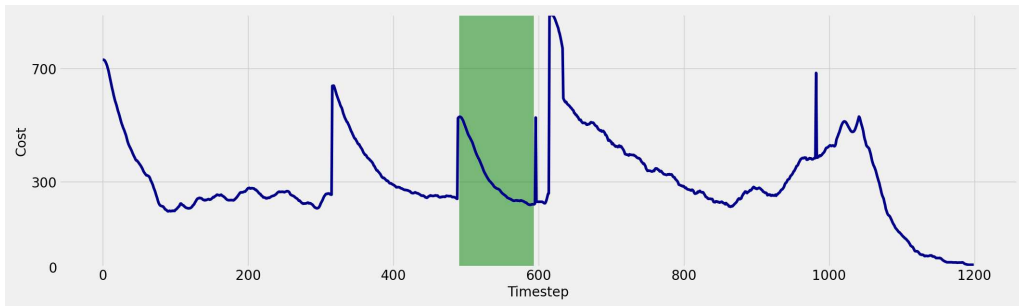
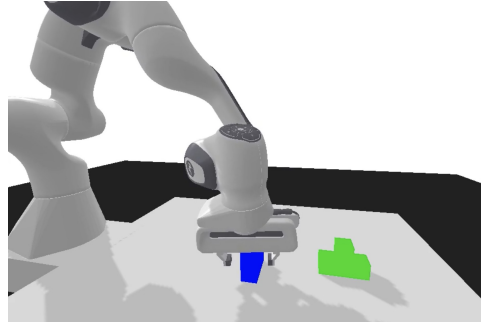
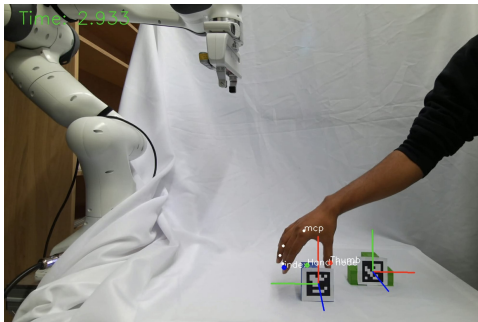


Figure 4.7: Moving To Grasping Position

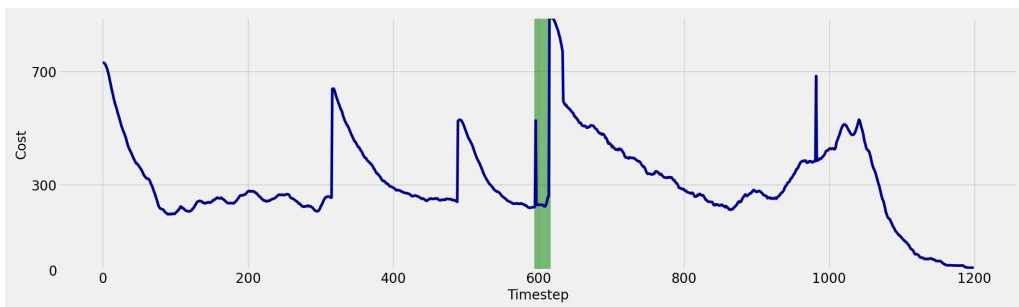
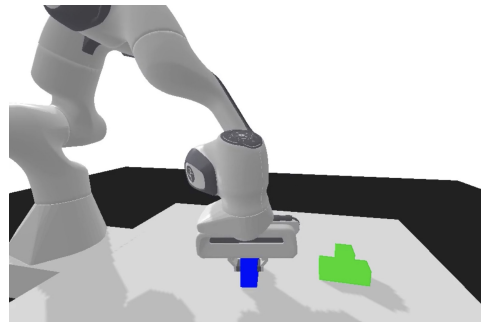
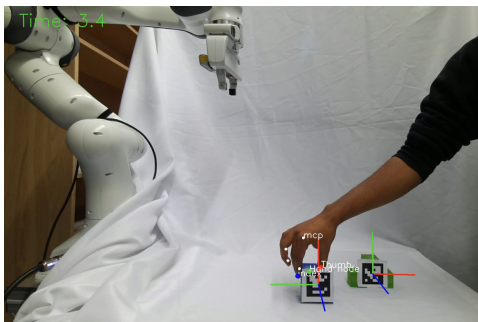


Figure 4.8: Closing The Gripper

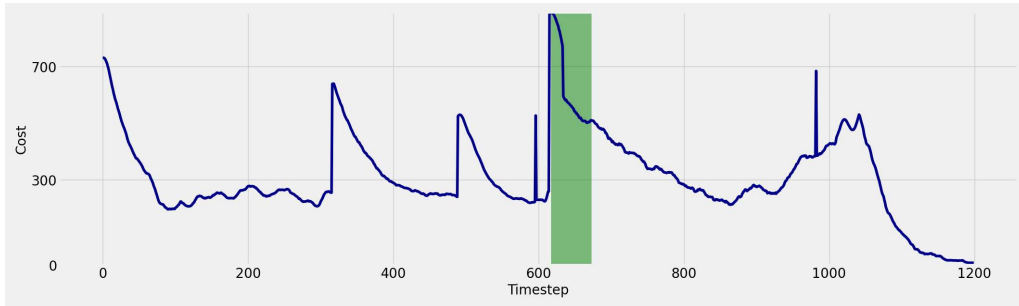
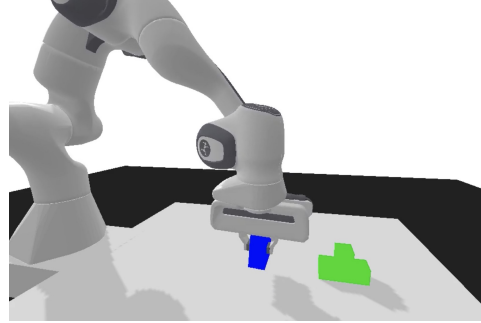
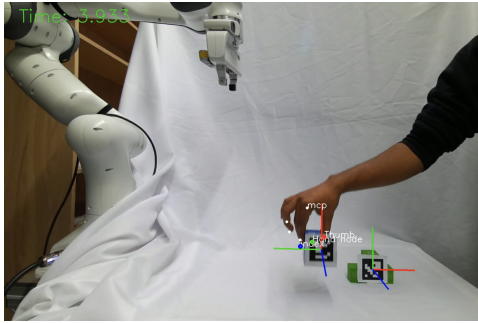


Figure 4.9: Lifting The Object

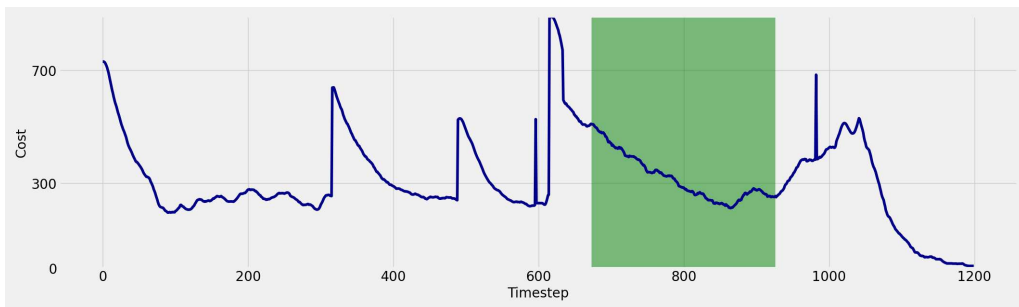
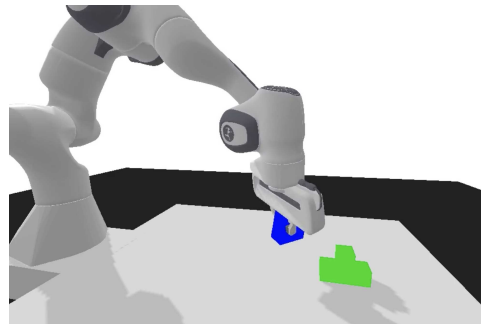
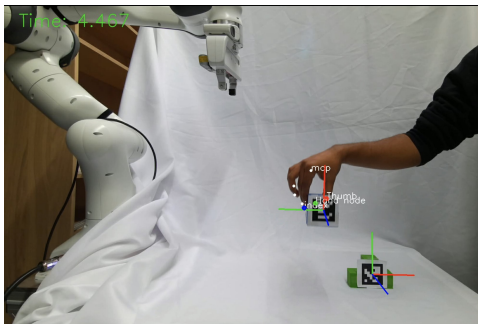


Figure 4.10: Directly Imitating Demonstrated Task Execution

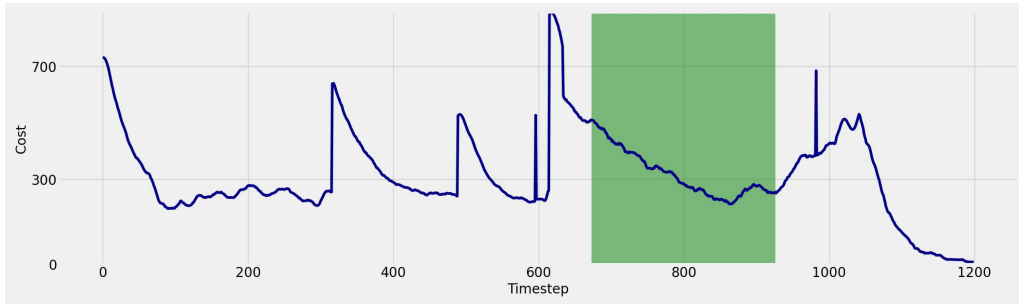
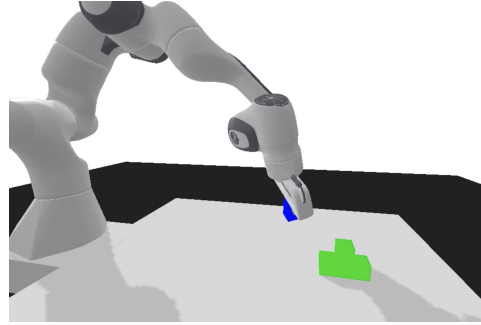
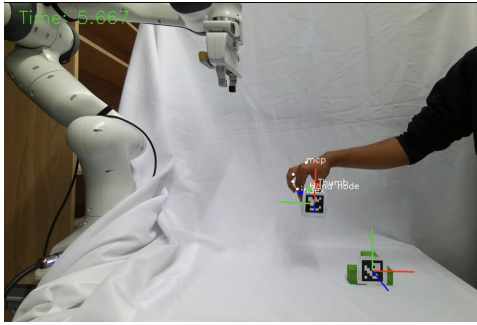


Figure 4.11: Directly Imitating Demonstrated Task Execution

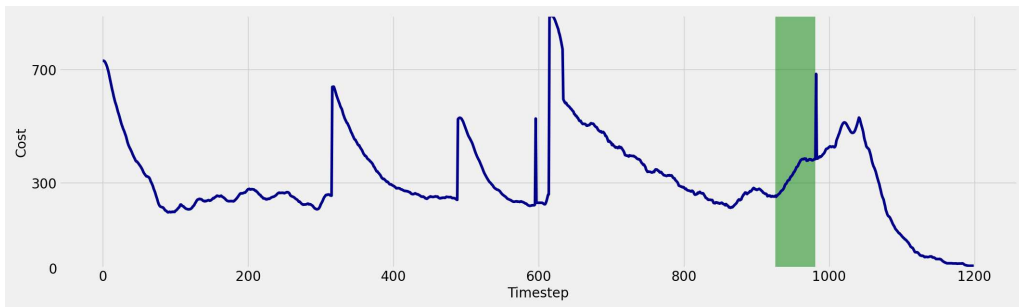
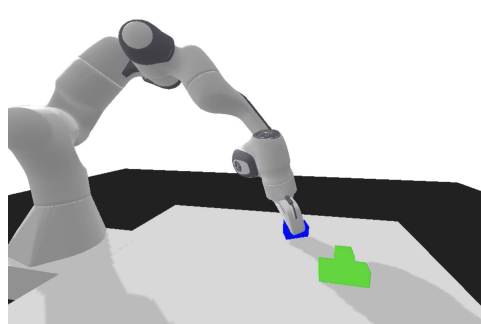
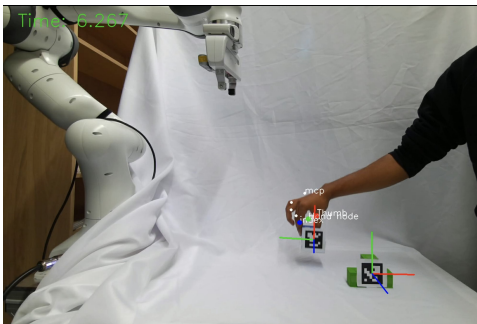


Figure 4.12: Reaching Pre-Release Pose

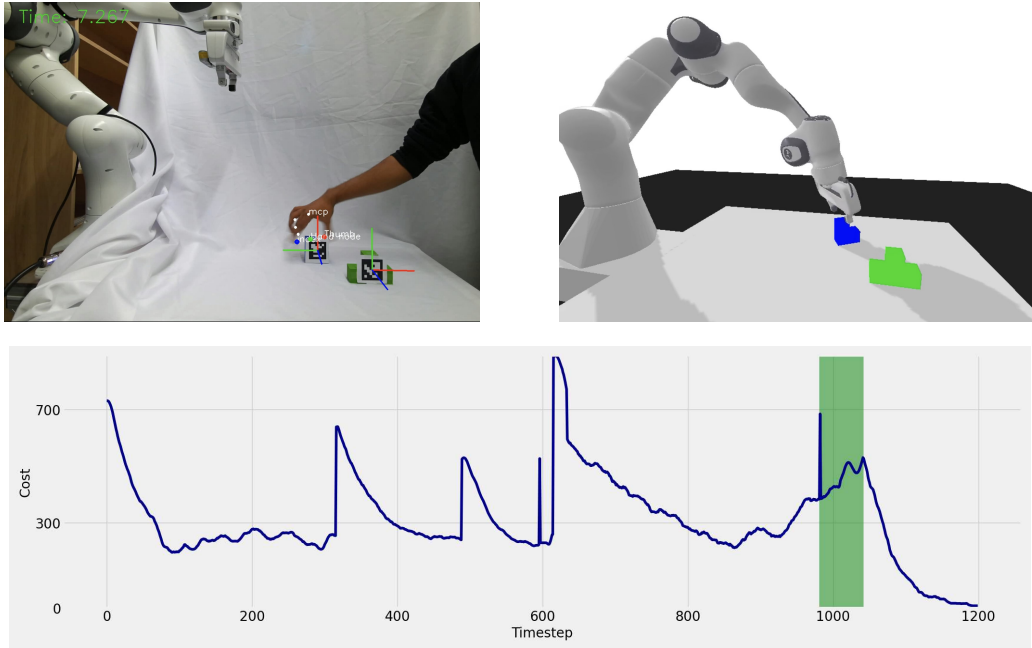


Figure 4.13: Releasing The Held Object

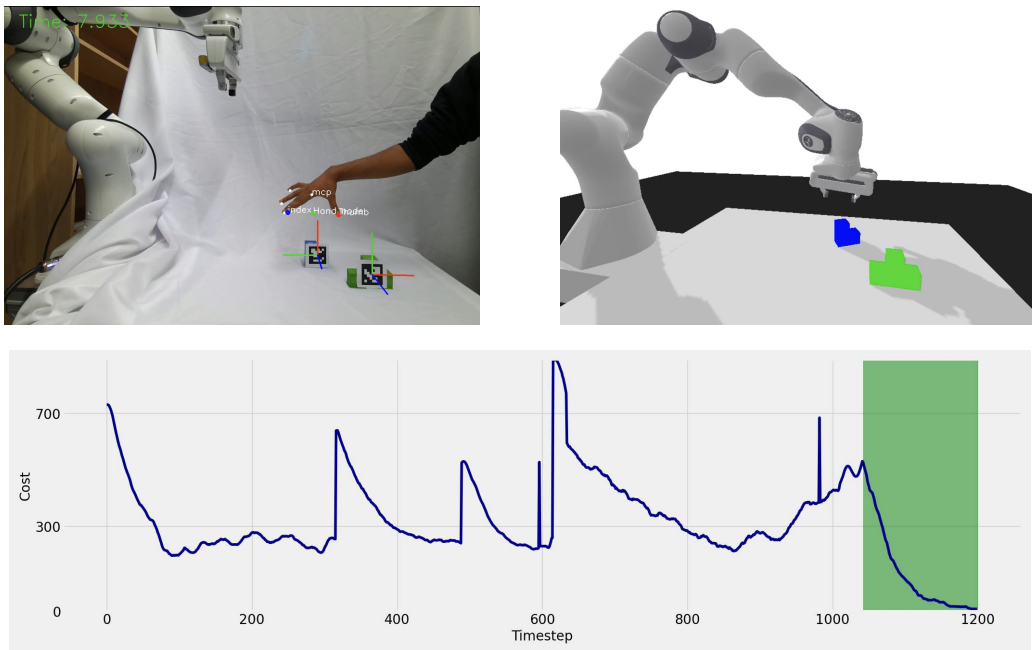


Figure 4.14: Directly Imitating Demonstrated Task Execution

Second, we intend to show that we can sequence multiple distinct pick and place tasks involving different objects within a demonstration. The conditions of this experiment are similar to the previous, but we introduce the constraint that initial relative object poses have to be equal to their counterparts in the observed execution since the generalization capability of our method is limited for multiple involved objects. The results of our imitation are provided below, with a visualization of individual entity orientations and the next action to be executed.

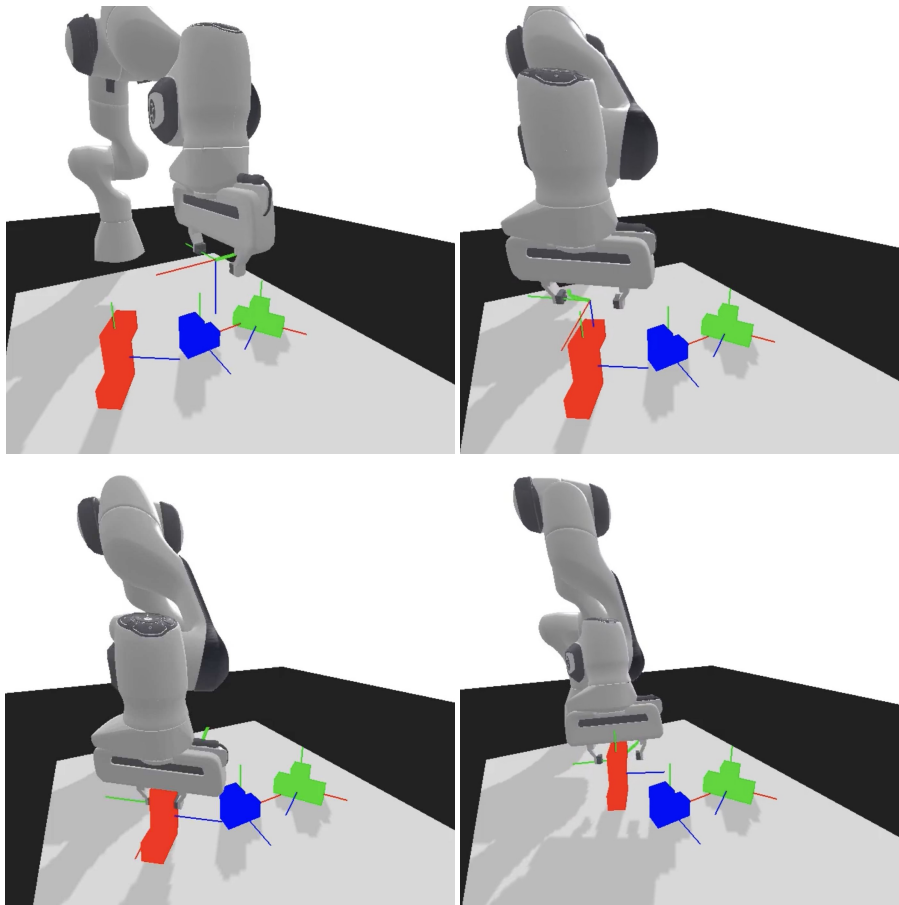


Figure 4.15: Simulation of the resulting imitation. Shown here is the first Pick and Place task in the sequence with (*top left*) initial configuration, (*top right*) first pre-grasp pose, (*bottom left*) first grasp, (*bottom right*) first release

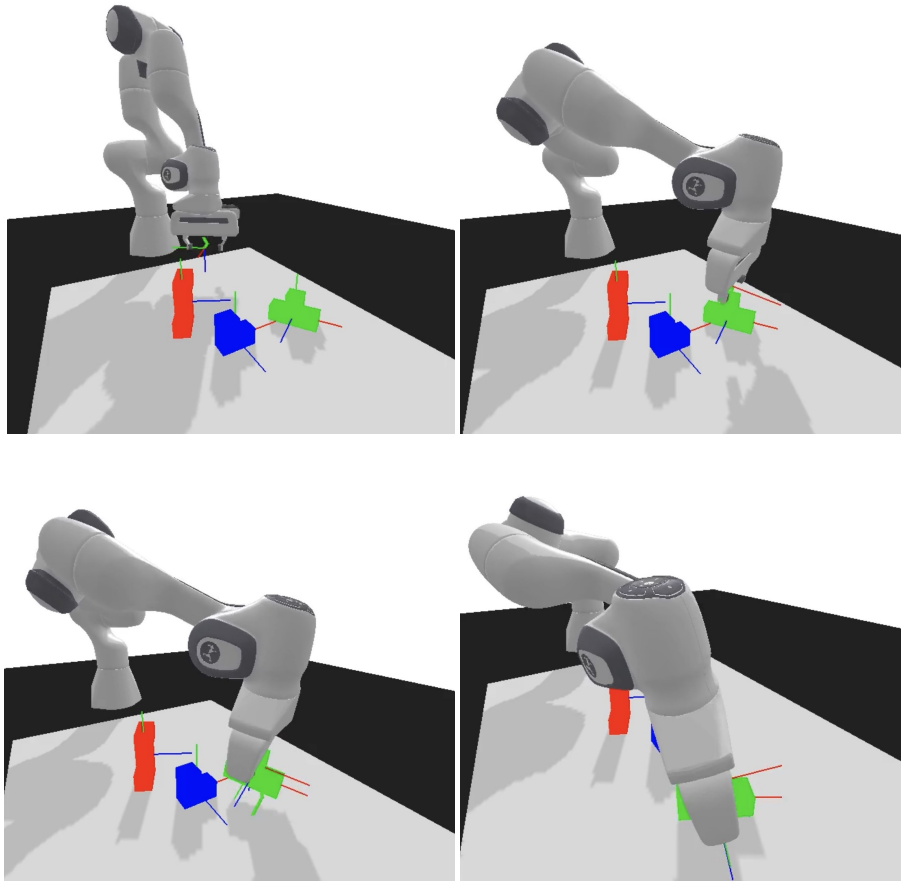


Figure 4.16: Simulation of the resulting imitation. Shown here is the second Pick and Place task in the sequence with (*top left*) imitating human transition, (*top right*) second pre-grasp pose, (*bottom left*) second grasp, (*bottom right*) second release

5 Conclusion

5.1 Discussion

The global convergence of our cost indicates that we can generalize from a single observed execution to a novel scenario quite well, even if different tasks are sequenced within a single demonstration. However, if more than 2 objects are present, we require their relative poses to be consistent with those observed. Otherwise, our method will optimize for a mean of relative distances. Furthermore, we can efficiently modify critical motion aspects to fit our robot’s embodiment better and achieve smooth transition by imitating the approach of the human hand in between. As demonstrated, we allow the teacher of a particular skill to perform it as naturally as possible, i.e., in an arbitrary environment and with a grasping approach of his choosing. This freedom is still restricted partially by our pose estimation techniques, which require all Aruco markers to be unoccluded and that as much as possible of the human hand is visible. Despite using a very basic dynamics model, our MPC implementation has proven to produce policies that can quickly minimize the overall cost as well as local spikes. The latter is caused by changes in the optimization target, which on a small scale correspond to a natural progression of the human motion or on a big scale to the beginning of high-level motion phases. Our control algorithm also enforced desired high-level behaviors parallel to task completion.

We proposed a framework for visual imitation learning that can work with single human video demonstrations per task. We introduced a graph-based task representation that is invariant against many problems of visual inference, bridges the human-robot domain gap, and allows for hand-crafted modifications of the motion to be integrated directly. We also described how to extract and purify the relevant data from RGBD video to make its collection as easy as possible. Furthermore, we have formulated a composition of cost functions that incentivizes the reproduction of relative object and gripper trajectories while also encouraging more general goals of robot manipulation. This function was used as part of a control algorithm based on model predictive control, which can make do with

simple system models and efficiently derive locally optimal policies online. We finally showed the combination of each element to a complete pipeline, that robot imitation of a performed skill within a few minutes, without requiring expert knowledge or prior training.

5.2 Future Work

Object Detection: Our current framework relies on Aruco markers to facilitate 6D pose estimation of task-related objects. Unfortunately, fiducial-based methods break down immediately when any degree of occlusion occurs, which happens regularly during grasping motions and if multiple objects are involved in a task. Replacing this method with a more reliable pose estimation technique is a straightforward improvement. Since we are using a predefined set of colorful rigid objects with known CAD-Models, the generation of large amounts of synthetic training data is relatively easy. We also assume access to RGBD input, which meets the requirements for more sophisticated learning-based tracking methods as [31] that are trained on specific object models and estimate their most likely pose under the current observation. These are very robust against occlusion, use previous detections as priors and also require no physical preparations.

Refining the MPC control: Since we use a composite cost for our MPC algorithm, the addition of more individual terms, which enables the completion of more side goals, is quite easy and only requires tuning of the corresponding weights as well as a proportional increase of the MPC's sample size. Desirable objectives in robot manipulation are collision avoidance with objects, path planning around joint singularities, or staying within certain velocity or acceleration limits. Also, more sophisticated sampling strategies that employ permanent covariance updates in all action dimensions, low discrepancy sampling of the control sequences, and efficient smoothing could significantly speed up the optimization process and improve the resulting trajectories. All previous points have been extensively explored in this work [16], which uses a wide array of auxiliary non-task-related requirements, Halton-sampling, and enforcement of low-jerk trajectories to achieve good results with little computation time.

Learning edge attention from metadata: A very interesting direction of future extension is meta-learning of edge attention based on the task relevance of a specific object-object relation across multiple demonstrations of the same task. This learned attention encodes a fundamental understanding, which would leverage the current limitations of generalizability and also remove uncertainties included in our heuristic to measure human attention. Our setup already allows for the collection of great amounts of processed training data in a short time, which is one of the main requirements for meta-learning approaches.

6 Appendix

6.1 Proof Of Metric Properties

We have introduced a distance metric $d_O : Q \times Q \rightarrow [0, 1]$ in the unit quaternion space Q , which is formulated as follows:

$$d_O(\mathbf{q}_1, \mathbf{q}_2) = \|\mathbf{q}_1^H \mathbf{q}_2\|_{\text{mag}} \\ \|\mathbf{q}\|_{\text{mag}} = 1 - |\text{Re}\{\mathbf{q}\}|$$

We are going to prove now that this function suffices all required properties of a metric. These properties are valid for all quaternions in Q :

$$\mathbf{q}_1^H \mathbf{q}_2 = \|\mathbf{q}_1\|^2 \Leftrightarrow \mathbf{q}_1 = \mathbf{q}_2 \quad (6.1)$$

$$\|\mathbf{q}\| = 1 \quad (6.2)$$

Given (6.1) and (6.2) proving the identity of indiscernibles is straightforward:

$$0 = 1 - |\text{Re}\{1\}| = d_O(\mathbf{q}_1, \mathbf{q}_2) \Leftrightarrow \mathbf{q}_1 = \mathbf{q}_2$$

We use the antihomomorphism of conjugations and:

$$\|\mathbf{q}^H\|_{\text{mag}} = 1 - |\text{Re}\{\mathbf{q}^H\}| = 1 - |\text{Re}\{\mathbf{q}\}| = \|\mathbf{q}\|_{\text{mag}}$$

to show that our metric is symmetric:

$$d_O(\mathbf{q}_1, \mathbf{q}_2) = \|\mathbf{q}_1^H \mathbf{q}_2\|_{\text{mag}} = \|(\mathbf{q}_2 \mathbf{q}_1^H)^H\|_{\text{mag}} = \|\mathbf{q}_2 \mathbf{q}_1^H\|_{\text{mag}} = d_O(\mathbf{q}_2, \mathbf{q}_1)$$

For the proof of the triangle inequality we refer to this work [32] (Algorithm 5), which presents a mathematically equivalent metric.

6.2 Experiment Hyperparameters

6.2.1 Data Collection

Hyperparameter	Value
FPS Camera	15
Object Labels	R, T
Relevant Hand Landmarks	THUMB, INDEX, INDEX_MCP
Marker Length	5 mm
Grasp Detection Threshold	4 cm

6.2.2 Filtering and Interpolation

Hyperparameter	Value
Savgol Window Length	51
Savgol Polynomial Order	3
Savgol Interpolation Mode	Interp
Median Filter Kernel Size	19
Distance Temperature	1
Outlier Group Size	7
Maximum Mahalanobis Distance Factor	5
Robot Control Rate	100 Hz

6.2.3 Task Representation Modifications

Hyperparameter	Value
Pre-Grasp Range	40
Grasp Range	10
Lift Range	20
Pre-Release Range	30
Post-Release Range	20
Pre-Grasp Height	5 cm
Post-Release Height	5 cm

6.2.4 Cost and Control

Hyperparameter	Value
Spatial Cost Weight	15
Orientation Cost Weight	380
Grasp Cost	300
Floor Contact Cost	60
Minimal Height	5 cm
Phase Position Precision	1 cm
Maximum Endeffector Displacement	5 cm
Maximum Endeffector Rotation	5 °
Sample Size N	100
Horizon Length H	5
Distribution Variance Σ	0.03
MPC Temperature β	0.05
Minimum Gripper Width	0 cm
Maximum Gripper Width	8 cm

Bibliography

- [1] K. Kim, Y. Gu, J. Song, S. Zhao, and S. Ermon, “Domain adaptive imitation learning,” 2019.
- [2] M. Sieb, “Visual imitation learning for robot manipulation,” Master’s thesis, Carnegie Mellon University, Pittsburgh, PA, May 2019.
- [3] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, “One-shot imitation from observing humans via domain-adaptive meta-learning,” 2018.
- [4] M. V. Balakuntala, L. N. V. Venkatesh, J. P. Bindu, and R. M. Voyles, “Self-evaluation in one-shot learning from demonstration of contact-intensive tasks,” 2019.
- [5] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, “Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective,” in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 391–398, 2012.
- [6] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, “Online movement adaptation based on previous sensor experiences,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 365–371, 2011.
- [7] F. Liu, Z. Ling, T. Mu, and H. Su, “State alignment-based imitation learning,” 2019.
- [8] J. Colombo and D. W. Mitchell, “Infant visual habituation,” *Neurobiology of Learning and Memory*, vol. 92, p. 225–234, Sep 2009.
- [9] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, “Learning complex dexterous manipulation with deep reinforcement learning and demonstrations,” 2017.
- [10] S. Yang, W. Zhang, W. Lu, H. Wang, and Y. Li, “Cross-context visual imitation learning from demonstrations,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5467–5473, 2020.

-
-
- [11] K. H. Lee, J. Lee, A. L. Thomaz, and A. F. Bobick, “Effective robot task learning by focusing on task-relevant objects,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2551–2556, 2009.
- [12] S. Nguyen, O. Oguz, V. Hartmann, and M. Toussaint, “Self-supervised learning of scene-graph representations for robotic sequential manipulation planning,” in *Proceedings of the 2020 Conference on Robot Learning* (J. Kober, F. Ramos, and C. Tomlin, eds.), vol. 155 of *Proceedings of Machine Learning Research*, pp. 2104–2119, PMLR, 16–18 Nov 2021.
- [13] J. Fu, S. Levine, and P. Abbeel, “One-shot learning of manipulation skills with online dynamics adaptation and neural network priors,” 2015.
- [14] T. Yu, P. Abbeel, S. Levine, and C. Finn, “One-shot hierarchical imitation learning of compound visuomotor tasks,” 2018.
- [15] S. Yang, W. Zhang, W. Lu, H. Wang, and Y. Li, “Learning actions from human demonstration video for robotic manipulation,” 2019.
- [16] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. Ratliff, D. Fox, F. Ramos, and B. Boots, “Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation,” 2021.
- [17] B. Kouvaritakis and M. Cannon, *Model predictive control classical, robust and stochastic*. Springer, 2016.
- [18] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, “Mediapipe hands: On-device real-time hand tracking,” 2020.
- [19] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, *Automatic generation and detection of highly reliable fiducial markers under occlusion*. Pergamon, Jan 2014.
- [20] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [21] J. Furtado, H. Liu, G. Lai, H. Lacheray, J. Desouza-Coelho, and Quanser, “Comparative analysis of optitrack motion capture systems,” 05 2018.
- [22] V. Petrik, M. Tapaswi, I. Laptev, and J. Sivic, “Learning object manipulation skills via approximate state estimation from real videos,” 2020.
- [23] M. Zhong, M. Johnson, Y. Tassa, T. Erez, and E. Todorov, “Value function approximation and model predictive control,” in *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 100–107, 2013.

-
-
- [24] B. Piot, M. Geist, and O. Pietquin, “Bridging the gap between imitation learning and inverse reinforcement learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 8, pp. 1814–1826, 2017.
- [25] UM-ARM-Lab, “Um-arm-lab/pytorch_mppi: Model predictive path integral (mppi) with approximate dynamics implemented in pytorch.”
https://github.com/UM-ARM-Lab/pytorch_mppi.
- [26] G. Williams, A. Aldrich, and E. Theodorou, “Model predictive path integral control using covariance variable importance sampling,” 2015.
- [27] P. Florence, L. Manuelli, and R. Tedrake, “Self-supervised correspondence in visuomotor policy learning,” 2019.
- [28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [29] K. Shoemake, “Animating rotation with quaternion curves,” *SIGGRAPH Comput. Graph.*, vol. 19, p. 245–254, jul 1985.
- [30] B. Ellenberger, “Pybullet gymperium.”
<https://github.com/benelot/pybullet-gym>, 2018–2019.
- [31] B. Wen, C. Mitash, B. Ren, and K. E. Bekris, “se(3)-TrackNet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, oct 2020.
- [32] J. Kuffner, “Effective sampling and distance metrics for 3d rigid body path planning,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 4, pp. 3993–3998 Vol.4, 2004.