2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

A Data-Driven Learning-from-Demonstration Framework for Robotic Grasping

Lars Pedersen, Erik D. Lindby, Jeppe Langaa, Leon Bodenhagen, Aljaz Kramberger

Abstract— **Autonomous grasp planning remains a key challenge in robotic manipulation, particularly in unstructured environments where object types, poses, and arrangements vary. This work presents a data-driven grasp planning method for a robotic manipulator tasked with clearing a table containing diverse objects. The method encodes human-demonstrated grasping strategies by representing Cartesian trajectories with Dynamic Movement Primitives (DMPs), whose parameters are predicted by a neural network from grasp-specific inputs. A second neural network estimates feasible grasp poses based on the object pose estimate data, which is used as the new goal parameter for the grasp trajectory generation.**

To reduce demonstration effort, synthetic datasets are generated via data augmentation of the recorded trajectories. The approach is implemented on a real system and evaluated on four objects with varying shapes and sizes. The experiments show a high success rate for grasping as well as the ability to incorporate new objects into the system through minimal additional effort.

I. INTRODUCTION

Autonomous robotic grasping remains a fundamental challenge in robotics, particularly in unstructured environments where object types, poses, and arrangements are unpredictable [1], [2]. Effective manipulation in such scenarios requires not only accurate perception but also the capability to plan and execute robust grasping trajectories for a wide variety of objects. Nowadays in robotic research, model based learning approaches are deployed in simulation, with reinforcement learning (RL) [3] and imitation learning [4] being deployed. The methods rely on high simulation capabilities and are computationally intensive. Systems may be trained via supervised learning (SL), where annotations are provided by humans or generated automatically in a self-supervised manner, or via reinforcement learning . Methods can operate as discriminative approaches, which sample and rank grasp candidates using a neural network [5], or as generative approaches, which directly produce grasp poses. Data-driven methods, on the other hand, have demonstrated significant promise by leveraging demonstrations and learning-based approaches to generalize grasp strategies to unseen objects [5], [6]. Training environments further differentiate methods. Some are trained exclusively in simulation, others in the real world, and some use a combination of both. The sensor modalities vary widely, including RGB images, depth images, RGB-D data, and point clouds, sometimes combining multiple sensor types. In order for learning to be successful, vision based systems are deployed for object segmentation and recognition in order to find the best grasp candidate [7].

Human demonstrations offer a rich source of information for teaching robots complex manipulation tasks. Encoding these demonstrations efficiently is crucial for reducing the effort required for training while preserving the variability inherent in human motions. Dynamic Movement Primitives (DMPs) provide a compact representation of demonstrated Cartesian trajectories, enabling generalization to new start and goal configurations [8]. Recent works have combined DMPs with neural networks to predict trajectory parameters from task-specific inputs, facilitating adaptive grasping in dynamic settings [9], [10].

Despite these advances, acquiring sufficiently large demonstration datasets remains a bottleneck. Data augmentation and synthetic trajectory generation have emerged as practical solutions to expand limited datasets without extensive additional human effort [11], [12]. By leveraging these techniques, robots can learn robust grasping policies capable of handling variations in object pose, orientation, and initial robot configuration.

In this work, we present a data-driven grasp planning method that combines neural networks to generate feasible grasp poses and corresponding trajectories for a robotic manipulator tasked for clearing a table of diverse objects. Synthetic datasets generated through trajectory augmentation reduce the reliance on extensive human demonstrations. The proposed approach is implemented on an real robotic platform and evaluated on four objects of varying shape and size, demonstrating robustness to pose and orientation variations. A key advantage of the method is the ease of integrating new objects into the system with minimal additional demonstrations.

II. METHOD OVERVIEW

The primary objective of the proposed approach is to simplify the deployment of robotic grasping tasks in unstructured environments, while minimizing the programming and calibration effort required on site. In particular, the system is designed so that even operators without prior robotics expertise can demonstrate grasping actions for previously unseen objects in a fast and intuitive manner. Instead of manually programming robot motion, the approach leverages *learning from demonstration* (LfD) to capture and encode human-guided grasping strategies that can be reproduced and adapted by the robot.

The overall method is composed of two core components: *grasp acquisition* and *robot execution*. In the first stage, the

The Maersk Mc-Kinney Moller Institute, SDU Robotics, University of Southern Denmark, Odense, Denmark, Email: alk@mmmi.sdu.dk

user physically demonstrates the desired grasping action by kinesthetically guiding the manipulator along an appropriate approach trajectory, while positioning the gripper to a suitable grasp pose. This demonstration is recorded in Cartesian space, with positions expressed in meters and orientations represented as quaternions. The approach trajectory is encoded using Cartesian space Dynamic Movement Primitives (CDMPs) [13], which reduce the complexity of the recorded data to a compact set of parameters while preserving the essential motion characteristics. In parallel, the final grasp pose is stored in a database, linked to the demonstrated object pose. Synthetic data augmentation is applied to expand the dataset, increasing robustness to variations in object pose.

In the second stage, the encoded grasp information is reproduced autonomously by the robot. A vision-based perception system, estimates the object pose in the robot's workspace. This information is used by a neural network to predict feasible grasp poses for the detected object. For each grasp pose, a second neural network predicts the CDMP parameters that generate an appropriate approach trajectory. The motion is then executed on the manipulator, equipped with either a pneumatic or a parallel-jaw gripper, depending on the target object.

The modular design of the system allows new objects to be incorporated with minimal additional effort, and the LfD interface ensures that task programming can be performed quickly and reliably by operators without robotics expertise. This combination of intuitive skill acquisition, compact trajectory encoding, and data-driven grasp pose prediction enables robust execution of grasping tasks in diverse and unpredictable environments.

The main contribution of this paper is: *a LfD-based, DMP–neural network pipeline for grasp planning that enables rapid introduction to novel objects without complete retraining of the model.*

The remainder of this paper is structured as follows: Section III details the proposed pipeline, including data augmentation, CDMP trajectory encoding, and neural network architectures. Section IV presents experimental validation and analysis, and Section V concludes with discussion and future work.

III. METHODOLOGY

In the following section we give an in-depth analysis of the methods used to solve the problem outlined in the introduction.

1) Data Collection: A standalone data collection framework was implemented for capturing and analyzing demonstration data through the Universal Robot (UR) Real Time Data Exchange (RTDE) [14] interface. For demonstration purposes the operator exploits the unbuild free drive functionality of the robot, which in terms enables recording of robot motion with simply guiding the robot in space. In this work we record Cartesian space poses $\mathbf{p} = [x, y, z]$ recorded in $[m]$ and orientations represented as a unit quaternion $\mathbf{q} = v + \mathbf{u} \in \mathrm{S}$, where S is the unit sphere in $\mathbb{R}^4$. In this way two datasets are recorded: (i) approach trajectories toward the object and (ii) object poses with multiple grasp configurations. Trajectories and grasps are obtained via kinesthetic teaching and managed through a Graphical User Interface (GUI). In the GUI the user can change the mode of the robot, select the desired tool and post process the recorded data e.g., cut the trajectory, before saving in the local data base. The demonstrations are recorded with a fixed sampling frequency of 500 $[Hz]$ and relative to the robot base frame. For the grasp pose recording, a vision system is deployed, which can automatically classify and pose estimate the objects. Computer vision is not the main research of this paper but used as a tool, therefore for more information see the following work by Naik et al., [15].

A. Trajectory Representation

In this work, we primarily focus on Cartesian space trajectories demonstrated by the operator which are encoded with CDMPs [13]. A CDMP is defined by the following free parameters: weights $\mathbf{w}_k^p, \mathbf{w}_k^q \in \mathbb{R}^3$ representing the positional and orientational parts of the trajectory, respectively; the trajectory duration τ; and the final desired position $\mathbf{g}^p$ and orientation $\mathbf{g}^q$ of the motion. The proposed CDMP formulation for encoding position ($\mathbf{p}$) and orientation ($\mathbf{q}$) trajectories is given by:

$$\tau\dot{\mathbf{z}} = \alpha_z\big(\beta_z(\mathbf{g}^p - \mathbf{p}) - \mathbf{z}\big) + \mathbf{f}_p(s), \tag{1}$$

$$\tau\dot{\mathbf{p}} = \mathbf{z}, \tag{2}$$

$$\tau\dot{\boldsymbol{\eta}} = \alpha_z\big(\beta_z \cdot 2\log(\mathbf{g}^q * \overline{\mathbf{q}}) - \boldsymbol{\eta}\big) + \mathbf{f}_o(s), \tag{3}$$

$$\tau\dot{\mathbf{q}} = \frac{1}{2}\,\boldsymbol{\eta} * \mathbf{q}, \tag{4}$$

$$\tau\dot{s} = -\alpha s, \tag{5}$$

where $\mathbf{z}$ and $\boldsymbol{\eta}$ denote the scaled linear and angular velocities, and α_z and β_z positive defined DMP gains. Mathematical operations in quaternion space, such as quaternion product ($*$), conjugation ($\overline{\mathbf{q}}$), and logarithm $\log(\mathbf{q})$, defined in [13].

The nonlinear forcing terms $\mathbf{f}_p(s)$ and $\mathbf{f}_o(s)$ are defined:

$$\mathbf{f}_p(s) = \frac{\sum_{k=1}^{N} \mathbf{w}_k^p \Psi_k(s)}{\sum_{k=1}^{N} \Psi_k(s)} s, \tag{6}$$

$$\mathbf{f}_o(s) = \frac{\sum_{k=1}^{N} \mathbf{w}_k^q \Psi_k(s)}{\sum_{k=1}^{N} \Psi_k(s)} s, \tag{7}$$

where $\mathbf{w}_k^p$ and $\mathbf{w}_k^q$ are free parameters contained in the forcing terms. They approximate any given Cartesian trajectory (position and orientation) when integrated from (1)–(4). The radial basis functions (RBFs) are defined as $\Psi_k(s) = \exp(-h_k(s - c_k)^2)$.

Following [16], the RBF centers and widths are defined as: $c_k = \exp(-\alpha_x \frac{k-1}{N-1})$ and $h_k = \frac{1}{(c_{k+1} - c_k)^2}$, $h_N = h_{N-1}$, $k = 1, \ldots, N-1$. The time constant is $\tau = t_T - t_1$. The goal position and orientation are set to the final position and orientation of the demonstrated trajectory. For more in depth information about CDMP's and DMP's please read the following work by Severiano et al., [17].

1) Roto-Dilatation for N-Dimensional DMPs: In order to efficiently generalize a N-dimensional DMP, we add a roto-dilatation term to the forcing function, as described in [18]. This method applies when the positional part of the CDMP is used in Cartesian space. The roto-dilatation term rotates and scales the trajectory according to start and goal changes, improving generalization capabilities compared to simply changing the goal ter of the DMP. We modify the forcing term notation as follows:

$$\mathbf{f}'_p(x) = \mathbf{S}\mathbf{f}_p(x), \tag{8}$$

where $\mathbf{S}$ is the roto-dilatation term:

$$\mathbf{S} = \frac{\|\mathbf{g}'_p - \mathbf{y}'_{p,0}\|}{\|\mathbf{g}_p - \mathbf{y}_{p,0}\|}\mathbf{R}^{\widehat{\mathbf{g}'_p - \mathbf{y}'_{p,0}}}_{\widehat{\mathbf{g}_p - \mathbf{y}_{p,0}}}, \tag{9}$$

with $\mathbf{g}'_p$ and $\mathbf{y}'_{p,0}$ as the new goal and start positions, respectively. The operator $\widehat{\cdot}$ denotes normalization. The scalar term applies dilatation, while $\mathbf{R}$ applies rotation, where as the start and goal positions must differ to avoid a null vector. The rotation term is computed as in [19], but follows the implementation in [18].

2) Modulation for Quaternion-Based DMPs: In [20], a method is introduced to modulate the orientation trajectory when changing the goal orientation. This modulation alters the forcing term across the trajectory:

$$\mathbf{f}'_o(x) = \mathbf{R}_q\mathbf{f}_o(x), \tag{10}$$

where $\mathbf{R}_q$ is computed by first finding the relative transformation:

$$\mathbf{q}^q_T = \mathbf{g}^q_{o,new} * \overline{\mathbf{g}}^q_{o,train}, \tag{11}$$

and then the new start orientation:

$$\mathbf{q}^0_{new} = \mathbf{q}^q_T * \mathbf{q}_0. \tag{12}$$

The rotation is computed by converting the start and goal orientations to angular velocity form:

$$\begin{aligned}\mathbf{q}_{T,new} &= 2\log(\mathbf{g}^q_{o,new} * \overline{\mathbf{q}}^0_{new}),\\ \mathbf{q}_{T,train} &= 2\log(\mathbf{g}^q_{o,train} * \overline{\mathbf{q}}^0_{train}).\end{aligned} \tag{13}$$

The relative rotation is expressed in axis–angle form:

$$\begin{aligned}\gamma_{ang} &= \arccos\left(\frac{\mathbf{q}_{T,new}\cdot\mathbf{q}_{T,train}}{\|\mathbf{q}_{T,new}\|\|\mathbf{q}_{T,train}\|}\right),\\ \boldsymbol{\gamma}_{ax} &= \|\mathbf{q}_{T,new}\times\mathbf{q}_{T,train}\|.\end{aligned} \tag{14}$$

This representation is then converted to a rotation matrix [21], $\mathbf{R}_q$, which at the end redefines the forcing term in equation (10) and substitutes it into equation (3) to improve behavior.

B. Introduction to the Neural Networks

Two Neural Networks (NNs) are utilized as function approximators. Although the two networks have different architectures, their fundamental design is the same. Both networks are multi layer perceptrons, which are fully connected feed-forward networks composed of multiple hidden layers. The activation function used for all hidden layers is the exponential linear unit, which transforms negative values according to a logarithmic curve. The output layer uses a linear activation function, as both tasks involve regression problems where negative values are possible.

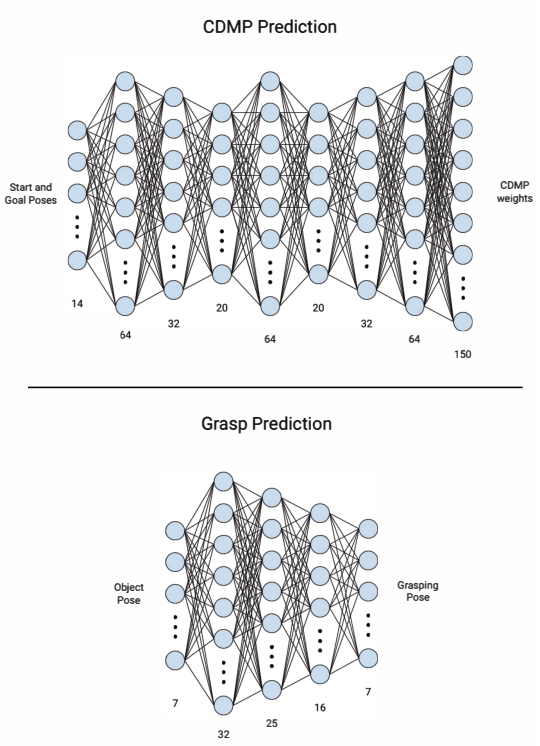


Fig. 1: Structure of the NN used to predict CDMP weights (top). The start and goal poses are input to seven hidden fully connected layers, with the output predicting the CDMP weights. As well as the structure of the NN used to predict the grasping pose (bottom). The object's pose represents the input to three hidden fully connected layers, with the output predicting the grasping pose.

1) Neural Network for Planning: The network for trajectory planning is shown in Figure 1 (top). The input consists of 14 values representing the start and grasping poses e.g., positions and orientations. The output dimension is determined by the number of weights used for each CDMP; 25 weights per DMP were found sufficient, resulting in 150 output values. The network architecture has seven hidden layers with varying numbers of neurons, inspired by [22], which compresses the input data for optimal learning and then decompresses it for prediction.

Preliminary tests showed that CDMP weights varied between -5000 and 2000, potentially reducing accuracy. Therefore, a standardization step is applied: outputs are rescaled to zero mean and unit variance, with the original mean and standard deviation stored for inverse transformation. The network uses mean squared error as its loss function:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2, \tag{15}$$

where Y_i and $\widehat{Y}_i$ denote the ground truth and predicted outputs, respectively.

2) Neural Network for Grasp Pose Estimation: The network for grasp estimation is illustrated in Figure 1 (bottom). It takes the seven dimensional input corresponding to the object pose provided by computer vision [15] and outputs a seven dimensional grasping pose, corresponding to a state vector of $\mathbf{X}_{grasp} = [\mathbf{g}^p_{new}, \mathbf{g}^q_{new}]$. The predicted grasping pose is used as the new goal for the trajectory generator. The network has three hidden layers with a descending number of neurons per layer. The loss function is designed specifically for 3D position and quaternion orientation outputs:

$$loss_{total} = loss_p + loss_q. \tag{16}$$

The positional loss is computed as the Euclidean distance between predicted $\widehat{\mathbf{Y}}_p$ and ground truth $\mathbf{Y}_p$ positions:

$$loss_p = \|\mathbf{Y}_p - \widehat{\mathbf{Y}}_p\| \tag{17}$$

For the quaternion orientation, the difference is first computed as:

$$\triangle \mathbf{Y}_q = \mathbf{Y}_q * \overline{\widehat{\mathbf{Y}}}_q, \tag{18}$$

where $\overline{\widehat{\mathbf{Y}}}_q$ is the conjugate of the predicted unit quaternion. The orientation loss is then calculated by averaging the absolute values of the scalar and vector components of the two quaternions:

$$loss_q = \frac{1}{4}\left(|1 - |\triangle Y_{qw}|| + |\triangle Y_{qx}| + |\triangle Y_{qy}| + |\triangle Y_{qz}|\right). \tag{19}$$

The scalar part is subtracted from one because when predicted and ground truth quaternions are identical, $\triangle Y_{qw} = 1$ and the vector part is zero.

C. Grasp generation for new objects

The system can roughly be described in four main parts: data collection, data generalization, NN training and execution on the real system.

1) Data Generalization for Grasping Poses: Training a NN to predict grasping poses requires a large dataset, which is impractical to demonstrate manually due to the vast variability in object and grasp configurations as well as susceptibility to human errors. To address this, a generalization method was developed that uses a small set of recorded object–grasp pose pairs to generate additional examples.

For each recorded pair, a transformation between the object and the corresponding grasp pose, T^{grasp}_{obj}, is computed as

$$T^{grasp}_{obj} = {T^{object}_{origin}}^{-1} T^{grasp}_{origin}. \tag{20}$$

By sampling new object poses within a defined workspace (uniformly for position, and using quaternion sampling on a sphere for orientation), corresponding grasp poses are generated via T^{grasp}_{obj}. Candidate poses are validated to ensure reachability and to avoid configurations such as the gripper pointing toward the robot base ($\theta < 0.4\pi$) or upwards (positive z-component in the grasp frame's z-axis vector). This process yields a large, diverse set of object–grasp pairs for NN training. At run time, this method proposes valid grasp candidates based on the pose estimate of the vision system.

2) Generation of synthetic trajectories based on human demonstration: The demonstrated Cartesian trajectories are generalized using DMPs, with both standard and rotationally modulated variants for position and quaternion orientation. Raw demonstrations are first processed by removing static segments at the beginning and end, identified via user input e.g., start and end of the recording. The processed trajectories are encoded with CDMP's, then decoded with randomized start and goal poses sampled from a predefined workspaces. Orientations are generated under similar angular constraints as in grasp pose generalization ($\theta_{start} < 0.4\pi$, $\theta_{goal} < 0.5\pi$). The result is a set of synthetically generated trajectories that maintain realistic kinematics while increasing dataset diversity. These trajectories, along with their start and goal poses, are stored for NN training.

3) Network Training: The two NNs described in the previous section are trained using the generated dataset to obtain object-specific models. The NNs were implemented in TensorFlow where the training employed a batch size of 32 and utilized early stopping to automatically determine the number of epochs. Model performance was empirically validated, with results presented in Section IV. For each new object introduced, the data generalization and NN training steps have to be executed, essentially creating a grasping model defining how an objects should be grasped.

4) Execution on the physical system: The trained grasp and trajectory neural network models were deployed on an ER-Flex robot system [23], a platform consisting of a six-DoF UR5e robot arm mounted on a MiR100 robot mobile base. The system is equipped with a Robotiq 2F-85 parallel gripper for grasping multiple objects. The vision system is implemented using an Intel RealSense D435 RGB-D camera, mounted near the tool center point of the robot arm to capture images for the object pose estimation. Training data was collected by kinesthetic demonstrations on the same hardware, where an operator physically guided the robot to perform example grasps and motions. These demonstrations were recorded and generalized to produce a training set for each of the two models.

A ROS pipeline connects the offline trained models, vision system, online inference, trajectory decoding and the ER-Flex motion interface, which can be seen in Fig. 2, along with a state machine that shows the control loop which follows a detect–plan–execute cycle: detect the target, plan the grasp trajectory using inference and approach, execute the motion and grasp, then proceed to the next item. When the Robotiq gripper is active, grasp confirmation is read from its built-in sensors. Camera extrinsic are calibrated to the robot base so that all poses and motions are expressed in a consistent frame.

The grasp model outputs a target grasp pose; together with the current TCP pose, this is passed to the trajectory model, which predicts DMP weights. A standard DMP decoder then generates a Cartesian path. Commands are streamed to the

ER-Flex controller as Cartesian setpoints with fixed velocity and acceleration, since the ER interface does not accept time-stamped trajectories. Before grasping the robot moves through predefined observation way-point poses to ensure a clear view. After grasping the object is placed at a fixed drop-off pose.

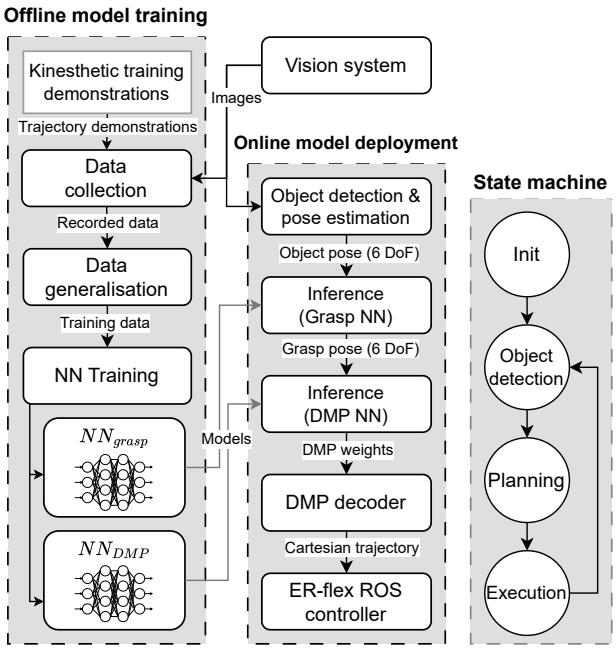


Fig. 2: Overall system overview. The left column shows how demonstrations are used to train the two models. The middle column shows how these models are used to generate trajectories for grasping objects. The right column describes the state machine.

IV. Experimental Evaluation

In this section we present the performance of the implemented methods described in the afore mentioned Section. We selected four test objects e.g., mug, bowl, cracker box and the bleach bottle, from the standard YCB benchmark [24]. For each of the four objects, grasping poses and trajectories were recorded and stored. Trajectories were captured at a frequency of 500 [Hz]. The stored data were then generalized, with each trajectory used 150 times and each grasping pose 1200 times. Table I summarizes the total generated data.

TABLE I: Overview of recorded and generated training data for each object.

	Mug	Bowl	Bleach bottle	Cracker box
Recorded trajectories	5	5	5	5
Generated trajectories	755	755	755	755
Recorded grasping poses	4	6	8	9
Generated grasping poses	4800	7200	9600	10800

The dataset is split into 85% for training, 10% for validation, and 5% for testing. The NN models for each object were trained and saved. For illustration purposes, the results presented in this section are for the mug object, similar figures can be created for the rest of the object respectively.

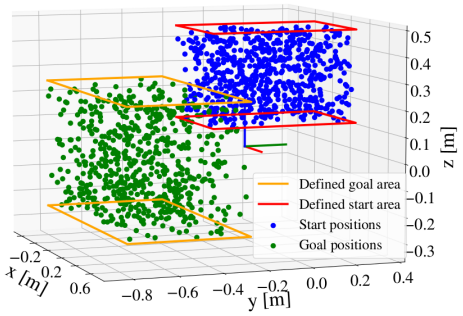


Fig. 4: Generated starting positions within the defined area. The x, y, and z axes are represented by red, blue, and green lines at the origin, respectively.

A. Trajectory Analysis

Figure 3 visualizes the five demonstrated trajectories for the mug. The starting poses are positioned such that the robot's camera faces the object, simulating a real grasping scenario. All trajectories follow a similar path toward the object while maintaining an approximately constant z value, ensuring a top-down approach for grasping.

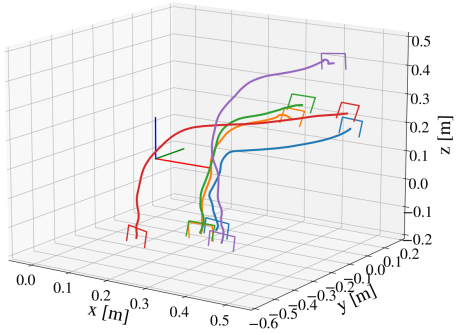


Fig. 3: Recorded trajectories for the mug. Each line represents a trajectory, with the gripper plotted at the start and goal poses. The x, y, and z axes are represented by red, blue, and green lines at the origin, respectively.

Figure 4 illustrates the generated start and goal positions within their defined limits. The start position limits were empirically defined in an area of $x = [0.20, 0.85]$[m], $y = [-0.4, 0.25]$[m], $z = [0.15, 0.5]$[m], and the goal area as $x = [-0.4, 0.7]$[m], $y = [-0.8, -0.25]$[m], $z = [-0.25, 0.25]$[m].

All start and goal poses were used to generate trajectories using CDMPs with generalization. The generated trajectories maintain the same top-down characteristic as the recorded trajectories, converging straight down above the goal position. The generated data was used to train NN for the mug object. In Figure 5 we show the comparison between the predicted trajectory generated from the NN model with a recorded trajectory. Both trajectories start and end at the same pose. The predicted trajectory closely follows the recorded trajectory with minor deviations, converging smoothly to the goal.

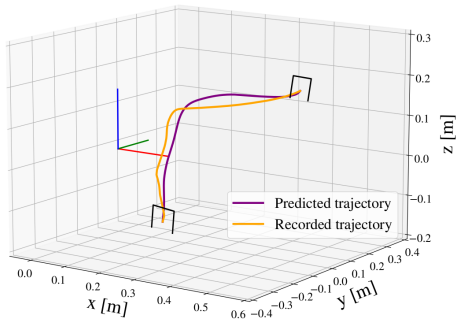


Fig. 5: Comparison of the recorded and predicted trajectories.

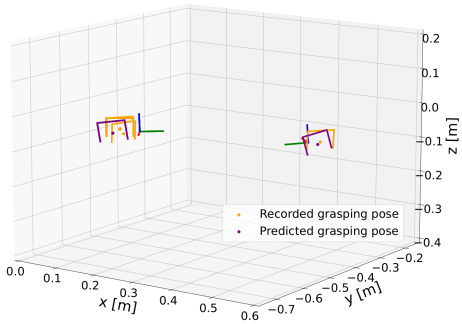


Fig. 6: Comparison of recorded and predicted grasping poses for the mug. Small frames represent object poses.

B. Grasping Pose Analysis

For verification purposes four grasping poses for the mug were acquired based on two object pose estimates.

These poses were used to generate a dataset, where the grasping position limits were defined as $x = [-0.568, 0.893]$[m] $y = [-0.875, -0.125]$[m], and $z = [-0.30, 0.30]$[m], extending beyond the robot's reach.

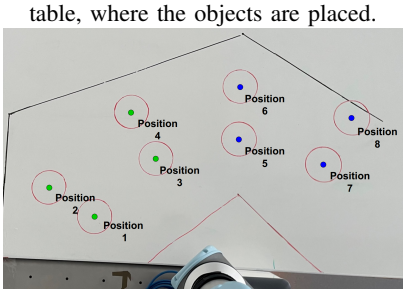
The generated grasping poses were used to train Grasping pose prediction NN. Figure 6 shows that predicted poses closely match the recorded ones in both position and orientation. Combining the both NN models allows the robot to approach objects correctly and end at a pose suitable for grasping.

C. Testing of the trained grasping models

To evaluate the performance of the system, several tests were performed in a real-world scenario. The tests combine the two trained NNs to execute a complete grasping sequence. These tests are designed to evaluate the system's performance under variations in the object's height, orientation, and the robot's starting pose. Figure 7 shows images of the experimental setup.

The robot is positioned against the table and can reach all objects. For repeatable tests eight object positions are defined in the robot's workspace. To avoid the inner workspace limit of the UR robot, the eight positions are split into two groups, with each group using a different starting pose. The starting poses are chosen so that the robot's camera can view the four positions of each group, as visualized in Figure 7b. For each test the vision system generates a pose

(a) The robot positioned against the table, where the objects are placed.



(b) The different positions where the objects are placed. A total of eight positions are used for the test.

Fig. 7: The pipeline test setup.

estimate corresponding to the centroid of the object. This pose is then used to generate the corresponding grasping poses and trajectories. Object are then manually placed at the corresponding positions on the table and the pipeline is repeated.

Validation is performed via visual inspection and gripper feedback. A grasping sequence is considered successful if the robot can grasp and lift the object for more than one second. If the grasping fails, the trajectory and grasping pose are evaluated independently. A grasp trajectory is deemed successful if the robot does not collide with the object. A grasping pose fails if the robot cannot be manually positioned to execute the grasp. To confirm a successful grasp, the robot is moved up in global Z direction and held for one second; if the object remains grasped, the trial is successful. The system's overall performance is shown based on the data from all the executed tests combined. The success rate for the entire grasp sequence is used to evaluate the system. A total of 56 grasping sequences are performed, and the results are shown in table II.

Success rate	Mug	Cracker box	Bowl	Bleach bottle
Grasping sequence	92.9%	92.9%	58.9%	42.9%

TABLE II: The resulting success rate for the grasping sequence for all tests.

In the table II, the grasping sequence for the mug and the cracker box achieves the highest success rate at 92.9%. The lowest success rate at 42.9% is obtained for the bleach bottle. The reason the bleach bottle obtains the lowest score is mainly due to NN inconsistencies generating a proper grasping pose furthermore, the object has a smooth surface, which does not allow for much friction, meaning that the object is easily dropped. The problem with the bowl is mainly caused by the trajectory. The robot must grasp the bowl at an angle, which causes the robot to move close to the base and thereby collide with itself or with the table.

V. Conclusions and future work

We developed a pipeline that generates Cartesian trajectories for a robotic arm to grasp specific objects in dynamic environments. The system employs two neural networks: one predicts CDMP weights from the demonstrated motion, while another estimates feasible grasp poses from object poses. Training data was collected via demonstrations, with roto-dilation and quaternion modulation introduced to improve generalization capabilities. The models were tested on the real system, enabling prediction of both grasp pose and trajectory for a given object.

System robustness was evaluated on four objects (bowl, mug, bleach bottle, cracker box) under variations in object height, orientation, and robot start pose. The approach was robust to height changes, with some executions achieving nearly 100% success. However, object orientation significantly affected performance, especially for the grasp-pose network. Overall grasping success rates reached 92.9 % (mug, cracker box) but dropped to 58.9 % (bowl) and 42.9 % (bleach bottle), leaving room for improvements to the method.

In future work the improvements include integrating obstacle avoidance into DMPs, better coupling of position and orientation adaptation, and exploring mixture density networks to handle objects with multiple valid grasp poses.

ACKNOWLEDGMENT

This work has been funded by the EU project Fluently (Grant agreement ID: 101058680), Bilateral Novo Nordisk – SDU Robotics collaboration project and supported by the Industry 4.0 lab at the University of Southern Denmark.

References

[1] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[2] Z. Xie, X. Liang, and C. Roberto, "Learning-based robotic grasping: A review," *Frontiers in Robotics and AI*, vol. 10, p. 1038658, 2023.

[3] R. S. Sutton, A. G. Barto, and R. J. Williams, "Reinforcement learning is direct adaptive optimal control," *IEEE control systems magazine*, vol. 12, no. 2, pp. 19–22, 2002.

[4] Y. Li, H. He, J. Chai, G. Bai, and E. Dong, "Grasping unknown objects with only one demonstration," *IEEE Robotics and Automation Letters*, 2024.

[5] J. Mahler, J. Liang, S. Niyaz, M. Laskey, A. Doan, X. Liu, J. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems (RSS)*, 2017.

[6] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," in *Robotics: Science and Systems (RSS)*, 2015.

[7] R. L. Haugaard, F. Hagelskjær, and T. M. Iversen, "Spyropose: Se (3) pyramids for object pose distribution estimation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2082–2091, 2023.

[8] S. Schaal, A. Ijspeert, and A. Billard, "Dynamic movement primitives—a framework for motor control in humans and humanoid robotics," in *Adaptive Motion of Animals and Machines*, pp. 261–280, Springer, 2006.

[9] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3828–3834, IEEE, 2011.

[10] O. Khatib, L. Sentis, J. Park, and J. Warren, "Robot learning from human demonstration: From human motion capture to robot motion generation," in *Springer Handbook of Robotics*, pp. 1993–2024, Springer, 2016.

[11] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[12] U. Viereck, A. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using easily simulated depth images," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 761–768, IEEE, 2017.

[13] A. Ude, B. Nemec, T. Petrič, and J. Morimoto, "Orientation in cartesian space dynamic movement primitives," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2997–3004, 2014.

[14] S. Robotics, "Universal robots rtde c++ interface." `sdurobotics.gitlab.io/ur_rtde/index.html`, August 2025.

[15] L. Naik, T. M. Iversen, A. Kramberger, J. Wilm, and N. Krüger, "Multi-view object pose distribution tracking for pre-grasp planning on mobile robots," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 1554–1561, IEEE, 2022.

[16] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[17] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.

[18] M. Ginesi, N. Sansonetto, and P. Fiorini, "Overcoming some drawbacks of dynamic movement primitives," *Robotics and Autonomous Systems*, vol. 144, p. 103844, 2021.

[19] O. I. Zhelezov, "N-dimensional rotation matrix generation algorithm," *American Journal of Computational and Applied Mathematics*, vol. 7, pp. 51–57, 2017.

[20] A. Kramberger, A. Kunic, I. Iturrate, C. Sloth, R. Naboni, and C. Schlette, "Robotic assembly of timber structures in a human-robot collaboration setup," *Frontiers in Robotics and AI*, vol. 8, 2022.

[21] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007.

[22] R. Pahič, Z. Lončarević, A. Gams, and A. Ude, "Robot skill learning in latent space of a deep autoencoder neural network," *Robotics and Autonomous Systems*, vol. 135, p. 103690, jan 2021.

[23] Enabled Robotics, "Er-flex: One robot—endless possibilities." www.enabled-robotics.com/. Accessed August 2025.

[24] B. Calli and A. Walsman, "Ycb benchmarks – object and model set." `www.ycbbenchmarks.com/`, 2015.