electronics

Article

DARC: Disturbance-Aware Redundant Control for Human–Robot Co-Transportation

Al Jaber Mahmud [1], Amir Hossain Raj [2], Duc M. Nguyen [2], Xuesu Xiao [2] and Xuan Wang [1,*]

[1] Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030, USA; amahmud5@gmu.edu
[2] Department of Computer Science, George Mason University, Fairfax, VA 22030, USA; araj20@gmu.edu (A.H.R.); mnguy21@gmu.edu (D.M.N.); xiao@gmu.edu (X.X.)
* Correspondence: xwang64@gmu.edu

Abstract: This paper introduces Disturbance-Aware Redundant Control (DARC), a control framework addressing the challenge of human–robot co-transportation under disturbances. Our method integrates a disturbance-aware Model Predictive Control (MPC) framework with a proactive pose optimization mechanism. The robotic system, comprising a mobile base and a manipulator arm, compensates for uncertain human behaviors and internal actuation noise through a two-step iterative process. At each planning horizon, a candidate set of feasible joint configurations is generated using a Conditional Variational Autoencoder (CVAE). From this set, one configuration is selected by minimizing an estimated control cost computed via a disturbance-aware Discrete Algebraic Riccati Equation (DARE), which also provides the optimal control inputs for both the mobile base and the manipulator arm. We derive the disturbance-aware DARE and validate DARC with simulated experiments with a Fetch robot. Evaluations across various trajectories and disturbance levels demonstrate that our proposed DARC framework outperforms baseline algorithms that lack disturbance modeling, pose optimization, or both.

Keywords: redundant robots; Disturbance-Aware Control; pose optimization; human–robot collaboration; Model Predictive Control

check for updates

Academic Editor: Enzo Pasquale Scilingo

Received: 29 April 2025
Revised: 9 June 2025
Accepted: 17 June 2025
Published: 18 June 2025

Citation: Mahmud, A.J.; Raj, A.H.; Nguyen, D.M.; Xiao, X.; Wang, X. DARC: Disturbance-Aware Redundant Control for Human–Robot Co-Transportation. *Electronics* **2025**, *14*, 2480. https://doi.org/10.3390/electronics14122480

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Human–robot collaboration is an emerging topic in research that helps to reduce humans' physical and cognitive workloads in various applications. For example, in industrial applications, collaborative robots are often used to perform repetitive tasks that enhance productivity and reduce human effort [1,2]. In such scenarios, robots generally operate behind safety fences, with physical separation from humans, to ensure safety [3]. Moreover, when humans and robots share the same workspace, the robots must be aware of human movements to ensure safety and achieve task goals [4]. This is important in both industrial and non-industrial settings. The application in non-industrial settings includes supporting elderly people with daily activities [5] and providing navigation assistance to individuals [6]. These applications require high-level planning with low-level control of the robot to achieve increased performance while maintaining safety. Increased performance refers to comparison with the performance level of either humans or robots operating independently [7].

Among these diverse applications, the co-transportation of objects by a human with a mobile manipulator (i.e., a robotic arm mounted on a mobile base) is a common scenario [8], as illustrated in Figure 1. In human–robot co-transportation tasks, there are challenges in

ensuring safe and efficient performance, especially when subjected to disturbances [9,10]. These disturbances include unpredictable human behavior, hardware imperfections such as sensor noise or actuator inaccuracies, etc. Even if the task has a predefined nominal path, humans often deviate from it due to fatigue, personal preferences, or external influences [11,12]. These deviations need to be compensated for by the robot to achieve the task goals. For a mobile manipulator, such adaptation toward disturbances is complex, as the robotic arm must dynamically reconfigure its joint angles to adapt to the human.

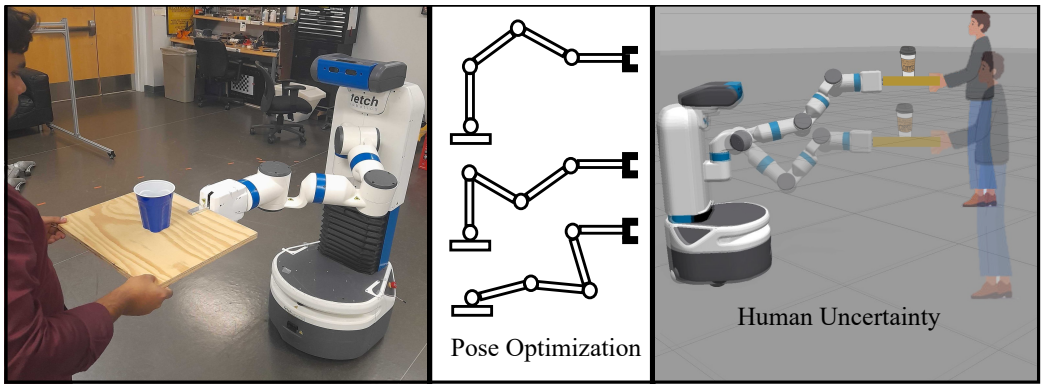


Figure 1. Human–robot co-transportation through disturbance-aware MPC and pose optimization (redundancy resolution for the same end-effector position).

The high-level planning of a mobile manipulator highly relies on the Model Predictive Control (MPC) framework, which has the predictive capabilities to enhance collaborative task performance [13]. However, this framework may lead to singularities or semi-singularities where the arm loses degrees of freedom or becomes stuck [14]. To address (semi-)singularities, proactive pose optimization can be implemented by exploiting the manipulator's kinematic redundancy [15,16]. Yet, limited existing works consider the integration of MPC and pose optimization, and they are not directly applicable to the co-transportation task considered in this paper [17,18]. Furthermore, the complexity rises in a mobile manipulator compared to a fixed-base robotic arm. Although using a mobile manipulator extends the system's workspace and maneuverability, such flexibility challenges the simultaneous coordination of the base's mobility with the arm's joint angle velocities [19].

To address these challenges, this work (a preprint of this paper is available in [20]) proposes a DARC framework by combining a disturbance-aware Model Predictive Control strategy as a high-level planner with a pose optimization mechanism at the low-level control. The MPC component plays the role of coordinating the movements of the mobile base and robotic arm by estimating the effects of disturbances on tracking accuracy and energy consumption. At the low-level control, we use a pose optimization mechanism that dynamically adjusts the robot's joint angles to compensate for disturbances. This also helps avoid singularities and enhances tracking accuracy under disturbances. This two-step interactive process, integrating the predictive capability of MPC and the proactive control of the robot with pose optimization, helps to reduce control effort and improve energy efficiency and tracking performance.

Statement of Contribution: The main contributions of this paper are as follows:

- We formulate a *high-level Model Predictive Control (MPC)* planner problem that incorporates disturbances along with a *low-level pose optimization* mechanism and the robot's whole-body kinematics. We propose a two-step iterative optimization approach to holistically solve the high-level and low-level problems.

- For the high-level MPC problem, we estimate control costs under disturbances and generate optimal control inputs. The initial state of the MPC controller depends on the low-level pose optimization.
- For the low-level pose optimization, we optimize the robot's pose selected from a joint configuration set generated using Conditional Variational Autoencoder (CVAE). The selection criteria are informed by the expected cost of the high-level MPC.
- We provide theoretical derivations and simulated experiments with a Fetch mobile manipulator to validate the DARC framework. Quantitative comparisons highlight the advantages of our method over different baselines.

2. Literature Review

Human–robot collaboration can be broadly classified into four levels based on their interaction modes: (i) *physical separation (No Coexistence)*, (ii) *shared space with no shared goals (Coexistence)*, (iii) *shared goals in shared space (Cooperation)*, and (iv) *simultaneous work on a shared object in shared space (Collaboration)* [21,22]. Co-transportation work falls into level (iv) as the human and the robot carry the same object in the same environment.

To enhance the performance of such collaborative tasks while ensuring safety by adapting to humans, integrating model-based control and reinforcement learning has been explored. Recent works use model-based reinforcement learning to assist humans in target activities [23], perform wood sawing and surface polishing [24], and execute Gaussian Process MPC in collaborative assembly tasks [13]. Furthermore, studies consider robust variable admittance control with unknown payload [25], sampling-based MPC for task and motion planning [26], and combined task/joint-space constraint handling [27]. Co-manipulation and transportation tasks are also addressed by considering whole-body kinematics of mobile manipulators [28–30] or employing human–humanoid collaboration strategies [31–33] to reduce human workload. While reinforcement learning methods are effective for handling unmodeled human uncertainties, they often lack transparency and rigorous guarantees of performance. For such issues, the control-theoretic framework is provided by robust MPC for addressing uncertainties, ensuring strict safety [34], adhering to physical constraints [35], and accommodating parameter variations [36]. However, these approaches do not integrate the disturbance-aware MPC formulation with pose optimization to enhance the overall performance further. This is one of the main differences from the problem addressed in our proposed framework. Additionally, they usually do not embed the disturbance component in the problem formulation itself; rather, they include it in the robot dynamics.

The idea of pose optimization is closely related to redundancy resolution, where multiple joint configurations can achieve a desired end-effector pose [16,37]. Redundancy resolution is one of the most important parts in controlling high-degree-of-freedom robotic systems, particularly for end-effector manipulation. Such kinematic redundancy helps to avoid singularities [38], improves dexterity in confined workspaces [39], and enhances precision under dynamic conditions [40]. Recent research has explored diverse methodologies to address these objectives—for instance, employing Monte Carlo tree search to optimize redundant configurations [41], introducing a sampling-based approach to navigate complex constraint spaces [42], and leveraging stiffness modeling to refine pose selection for task-specific compliance [43]. In soft robotics, [44] demonstrated redundancy resolution for continuum manipulators by adapting poses to flexible forms. Additionally, many researchers have used learning methods for redundant manipulators. These include utilizing the proximal policy optimization algorithm [45], deep deterministic policy gradient algorithm [46], and deep reinforcement learning [47]. In general, these works highlight the adaptability of pose optimization in overcoming environmental and operational constraints.

Although pose optimization has been investigated in many scenarios, these seldom account for disturbances [48]. In contrast to prior approaches, our formulation redefines pose optimization by coupling it with a disturbance-aware MPC framework. This strategy helps to select poses that not only satisfy kinematic goals but also minimize the impact of disturbances.

3. Preliminaries and Problem Formulation

In this section, we first define how disturbances impact the co-transportation process in terms of a nominal trajectory for the human and robot to complete the task. Next, we present the kinematic model of a mobile manipulator's end-effector, where we integrate the kinematics of its mobile base and arm. Finally, we formulate an MPC problem that incorporates pose optimization to address disturbances effectively.

Notations: Let I_r be the $r \times r$ identity matrix and $\text{diag}\{a_1, \cdots, a_r\}$ be a diagonal matrix whose entries are $a_1, \ldots, a_r$. For any vector x, let $\|x\|_2$ be the Euclidean norm. If M is a square matrix, $\text{Tr}(M)$ is the trace. Additionally, $M \succ 0$ ($M \succeq 0$) denotes that M is positive (semi-)definite We define $\|x\|_M^2 = x^\top M x$.

3.1. Trajectory with External Human Disturbances

Consider the co-transportation task illustrated in Figure 1, where a human and a mobile manipulator jointly move an object. Let

$$\boldsymbol{r}(k) = \left[r_x(k),\ r_y(k),\ r_z(k),\ r_\alpha(k),\ r_\beta(k),\ r_\gamma(k)\right]^\top \in \mathbb{R}^6$$

be the manipulator's end-effector pose, where (r_x, r_y, r_z) denotes the Cartesian position and $(r_\alpha, r_\beta, r_\gamma)$ the orientation in Euler angles, respectively. We define this pose over a finite horizon $k = 0, 1, \ldots, H$, where H is the planning horizon. To account for external human disturbances, we incorporate a disturbance term $\varpi(\tau)$ in the trajectory:

$$\boldsymbol{r}(k) = \tilde{\boldsymbol{r}}(k) + D \sum_{\tau=0}^{k} \varpi(\tau), \tag{1}$$

where $\tilde{\boldsymbol{r}}(k) \in \mathbb{R}^6$ is a nominal trajectory. If we assume that the task has a desired trajectory, this $\tilde{\boldsymbol{r}}(k)$ can be treated as known to the human and the robot. Otherwise, we may consider trajectory prediction of $\tilde{\boldsymbol{r}}(k)$ by the robot based on human behaviors. In existing works, such trajectory prediction can be achieved using geometry-based curve fitting [49], utility function frameworks [50], or learning methods like recurrent neural networks [51,52], LSTM [53–55]. We use LSTM (the detailed implementation will be described in the Appendix A) because it can capture temporal dependencies in sequential data. This makes it beneficial in predicting the continuous, non-linear evolution of human movement. The term $\varpi(\tau)$ represents the disturbance due to human movement, modeled as a Gaussian distribution—i.e., $\varpi(\tau) \sim \mathcal{N}(\alpha_\varpi, \Sigma_\varpi)$ in the x, y, and z directions where τ denotes the time step. Here, $\alpha_\varpi \in \mathbb{R}^3$ is the non-zero-mean and $\Sigma_\varpi \in \mathbb{R}^{3\times3}$ is the covariance matrix that captures individual variations or preferences of the human collaborator. $D \in \mathbb{R}^{6\times3}$ is a matrix that maps $\varpi(\tau)$ into the $6D$ pose space. To focus on the translational disturbances caused by the human, we assume the orientation $(r_\alpha, r_\beta, r_\gamma)$ remains unaffected by these disturbances. This is because the orientation is co-determined by the robot and the human. Thus, the disturbance towards orientation is not necessarily impacted only by the human, and the robot can compensate for this. Thus, we define $D = \begin{bmatrix} I_3 & \mathbf{0}_{3\times3} \end{bmatrix}^\top$.

3.2. End-Effector Kinematics

We illustrate the modeling details using a Fetch Mobile Manipulator [56], as we used this robot for experiments. The Fetch robot has a differential-drive mobile base and a 7-DOF robotic arm. In the following, we consider simplified kinematics, focusing on velocity inputs.

Note that our proposed framework is applicable to either a dynamic or a kinematic model of the end-effector of any kind of mobile manipulator. A dynamic model with torque control requires the use of accurate inertial parameters, the modelling of non-linear Coriolis and gravity terms, and the capturing of friction and actuator dynamics. When using dynamic models instead of kinematic models, the computational and implementation complexity might increase, but the process remains unchanged. Additionally, the approach can be easily generalized to other types of mobile manipulators, such as robots with an omnidirectional base and a robotic arm mounted on top. For this, we only need to modify the corresponding kinematics equations accordingly.

Mobile Base. The kinematics model of the mobile base of the Fetch Mobile Manipulator can be approximated using a standard differential-drive model in the inertial frame (Figure 2a). Let

$$\boldsymbol{s}_{\text{base}}(k) \;=\; \begin{bmatrix} x_{\text{base}}(k) & y_{\text{base}}(k) & \phi(k) \end{bmatrix}^{\top} \in \mathbb{R}^3$$

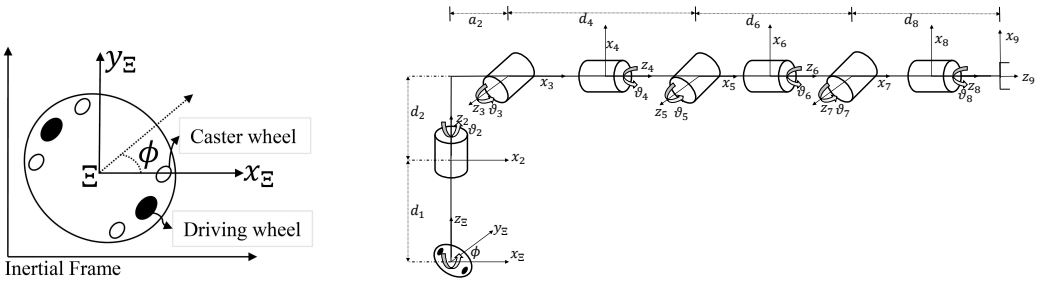
be the pose of the base. This has Cartesian coordinates $(x_{\text{base}}, y_{\text{base}})$ in the inertial frame and heading angle ϕ. The state update of s_{base} from step k to $k+1$ is given by

$$\boldsymbol{s}_{\text{base}}(k+1) \;=\; \boldsymbol{s}_{\text{base}}(k) \;+\; \Delta t \begin{bmatrix} \cos(\phi(k)) & 0 \\ \sin(\phi(k)) & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u}_{\text{base}}(k), \tag{2}$$

where

$$\boldsymbol{u}_{\text{base}}(k) \;=\; \begin{bmatrix} v(k) & \dot{\phi}(k) \end{bmatrix}^{\top} \in \mathbb{R}^2$$

where the base's linear velocity is v and the angular velocity is $\dot{\phi}$ at time k. We denote the time interval as Δt. Here, $(v, \dot{\phi})$ are in the robot's body frame Ξ, which translates and rotates with the base.



(**a**) (**b**)

Figure 2. (**a**,**b**) show the Fetch robot's mobile base and arm configuration. (**a**) Fetch robot's base in the inertial frame. A shifted frame Ξ centered at the robot's current position. (**b**) Denavit–Hartenberg (DH) representation with 8-DOF (manipulator's joint angles and base's heading angle) joint configurations in Ξ frame.

Robotic Manipulator. Fetch robot incorporates a 7-DOF manipulator mounted on top of its base. We denote each joint angle as ϑ_i, where $i \in \{2, \ldots, 8\}$ represent seven joint angles. In addition, we include the base's heading angle ϕ as an extra DOF that yields an 8-DOF system as illustrated in Figure 2b. This helps to merge the kinematics of the mobile base and the manipulator by enabling the transformation of the end-effector's pose in the

robot body frame from the inertial frame without any rotation and vice versa. Additionally, it allows the incorporation of the heading angle ϕ into the pose optimization along with the arm's joint angles. Let

$$\boldsymbol{\theta} = [\,\phi,\ \vartheta_2, \ldots, \vartheta_8\,]^\top \in \mathbb{R}^8$$

combine the base's heading angle and the arm's seven joint angles. We denote the corresponding angular velocities by

$$\boldsymbol{\omega} = [\,\dot{\phi},\ \dot{\vartheta}_2, \ldots, \dot{\vartheta}_8\,]^\top \in \mathbb{R}^8.$$

Let s_{arm} be the end-effector pose in the body frame Ξ. Specifically,

$$\boldsymbol{s}_{\text{arm}} \;=\; \begin{bmatrix} p_{\text{arm}} \\ \psi_{\text{arm}} \end{bmatrix} \;\in\; \mathbb{R}^6,$$

where $p_{\text{arm}} \in \mathbb{R}^3$ denotes the Cartesian position of the end-effector in Ξ frame, and $\psi_{\text{arm}} \in \mathbb{R}^3$ represents the orientation in Euler angles in the same frame. Note that our use of Euler angles is primarily for simplicity, although we are aware that Euler angles may potentially lead to singularities [57]. In our application, the target end-effector orientation is maintained horizontally relative to the ground, which helps mitigate the issue. Orientation may also be represented using quaternions or $\mathbb{SO}(3)$ with complex dynamic matrices. However, we claim that the choice of the representation method does not impact our results if we modify the equations related to Euler angles in Section 4.

Although the original model of forward kinematics is non-linear, it can be locally linearized around $\boldsymbol{\theta}(k)$ for small sampling intervals. We perform linearization to map the end-effector pose from joint angle velocities, which is a standard process in MPC to enhance computational tractability. Additionally, if the planning horizon of MPC is short, then the impact of this linearization is not significant. Letting $f(\cdot)$ be the forward-kinematics map, we have

$$\boldsymbol{s}_{\text{arm}}(k+1) \;=\; f\big(\boldsymbol{\theta}(k) + \Delta t \cdot \boldsymbol{\omega}(k)\big) \;\approx\; s_{\text{arm}}(k) \;+\; \Delta t \cdot \mathbf{J}\big(\boldsymbol{\theta}(k)\big)\,\boldsymbol{\omega}(k), \tag{3}$$

where $\mathbf{J}\big(\boldsymbol{\theta}(k)\big) \in \mathbb{R}^{6\times 8}$ is the Jacobian matrix, which is computed as $\mathbf{J}(\boldsymbol{\theta}(k)) = \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}(k)}$. The specific form of $\mathbf{J}$ is derived from the DH parameters of the Fetch arm [56].

End-Effector Kinematics. We integrate the kinematics of the base (2) and manipulator (3) to capture the end-effector kinematics of the manipulator in the inertial frame. We define

$$\boldsymbol{s} \in \mathbb{R}^6 \quad \text{where} \quad \boldsymbol{s} \;=\; \boldsymbol{s}_{\text{arm}} + \begin{bmatrix} x_{\text{base}} \\ y_{\text{base}} \\ 0_{4\times 1} \end{bmatrix}.$$

Because $\boldsymbol{s}_{\text{base}}$ is measured in the inertial frame, $\boldsymbol{s}_{\text{arm}}$ is in the frame Ξ, and ϕ is already included in $\boldsymbol{\theta}$, the heading angle change is effectively captured by the Jacobian matrix itself. Combining the control inputs into $\boldsymbol{u} = [\,v,\ \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^9$ and combining (2) and (3) into a single linearized state transition for $s(k)$ yields

$$\boldsymbol{s}(k+1) = \boldsymbol{s}(k) + B\big(\boldsymbol{\theta}(k)\big)\,\boldsymbol{u}(k), \tag{4}$$

where

$$B(\boldsymbol{\theta}(k)) = \Delta t \begin{bmatrix} \cos(\phi(k)) & \\ \sin(\phi(k)) & \mathbf{J}(\boldsymbol{\theta}(k)) \\ 0_{4\times 1} & \end{bmatrix} \in \mathbb{R}^{6\times 9}$$

is the input matrix. Here, the first column represents the contribution of the base's linear velocity, and the other columns reflect the effect of the angular velocities $\boldsymbol{\omega}(k)$ through the Jacobian $\mathbf{J}(\boldsymbol{\theta}(k))$.

3.3. MPC-Based Tracking with Pose Optimization

To achieve effective co-transportation, the robot mainly aims to precisely control its end-effector pose by adjusting its joint angles. This ensures seamless coordination with the human during the collaborative task. However, due to the inherent redundancy of high-degree-of-freedom manipulators, a given end-effector pose corresponds to infinitely many joint angle combinations. These different configurations can lead to different future costs. For example, some joint combinations may perform better in tracking trajectories along the y-axis, while some others may be better for the z-axis. Motivated by this observation, we introduce an approach that allows the robot to slightly adjust its joint configuration between each control input without changing the end-effector pose. Such adjustments enable the robot to proactively select a configuration from multiple solutions that potentially minimize future tracking costs. This leads to jointly optimizing both the control inputs and the robot's joint configuration, as we describe next.

Our goal is to track the trajectory $\boldsymbol{r}(k)$ while considering the robot's whole-body kinematics. To achieve this, we formulate a disturbance-aware Model Predictive Control (MPC) problem that incorporates *pose optimization*. This allows the robot to select a new joint configuration $\bar{\boldsymbol{\theta}}$, which may differ from the initial configuration $\boldsymbol{\theta}_0$. We formulate the following finite-horizon optimization problem:

$$\begin{aligned} \min_{\boldsymbol{u}(0:H-1),\, \bar{\boldsymbol{\theta}}\in\Theta} \quad & \mathcal{J}\left(\boldsymbol{u}(0:H-1), \bar{\boldsymbol{\theta}}\right) \\ \triangleq \; & \mathbb{E}_{\varpi}\Bigg[\sum_{k=0}^{H} \underbrace{\left(\boldsymbol{s}(k)-\boldsymbol{r}(k)\right)^{\top} Q \left(\boldsymbol{s}(k)-\boldsymbol{r}(k)\right)}_{\text{Tracking Error Cost}}\Bigg] \\ & + \sum_{k=0}^{H-1} \underbrace{\boldsymbol{u}(k)^{\top} R\, \boldsymbol{u}(k)}_{\text{Control Cost}} + \underbrace{\kappa \left\|\bar{\boldsymbol{\theta}}-\boldsymbol{\theta}_0\right\|_2^2}_{\text{Pose Optimization}} \\ & + \underbrace{w_\theta \exp\Big(-\left\|\boldsymbol{\theta}^{\lim}-\bar{\boldsymbol{\theta}}\right\|_2^2\Big)}_{\text{joint limit penalty}} \end{aligned} \tag{5}$$

subject to

$$\begin{aligned} \boldsymbol{s}(k+1) \;&=\; \boldsymbol{s}(k) \;+\; B\big(\bar{\boldsymbol{\theta}}\big)\, \boldsymbol{u}(k)\,, \quad \boldsymbol{s}(0) = \boldsymbol{s}_0, \\ \boldsymbol{r}(k) \;&=\; \tilde{\boldsymbol{r}}(k) + D \sum_{\tau=0}^{k} \varpi(\tau), \qquad \varpi(\tau) \sim \mathcal{N}(\alpha_\varpi, \Sigma_\varpi). \end{aligned}$$

Here, $\boldsymbol{u}(0:H-1) = \{\boldsymbol{u}(0), \boldsymbol{u}(1), \ldots, \boldsymbol{u}(H-1)\}$ denote the control input sequence over finite horizon H, and Θ is the candidate set of all feasible joint configurations. The tracking error cost term penalizes the deviations of the end-effector pose $\boldsymbol{s}(k)$ from the trajectory $\boldsymbol{r}(k)$ over the planning horizon. The matrices $Q \succeq 0$ and $R \succ 0$ weigh the tracking error and control efforts costs, respectively. $\kappa > 0$ adjusts how strongly the optimizer prefers to remain near the initial configuration $\boldsymbol{\theta}_0$. The joint limit penalty discourages the robot

from approaching its joint limits. This penalty term exponentially increases the cost as the pose $\bar{\theta}$ approaches the joint limit values $\boldsymbol{\theta}^{\text{lim}}$. This helps the robot operate within safety bounds and avoids any potential configuration that may lead to a stuck pose. The coefficient $w_\theta > 0$ is the weighting factor for the joint limit penalty. The expectation $\mathbb{E}_{\varpi}[\cdot]$ is computed over the disturbance $\varpi(\tau)$. Note that, for computational tractability, $B(\bar{\boldsymbol{\theta}})$ is treated as fixed over the planning horizon based on the new configuration $\bar{\boldsymbol{\theta}}$. This technique is common in MPC implementations with short horizons [58,59] since re-linearizing at each step would increase the computational burden. By allowing the robot to proactively adjust its joint configuration from $\boldsymbol{\theta}_0$ to $\bar{\boldsymbol{\theta}}$, we claim that our framework selects more favorable poses to better compensate for disturbances.

We also note that the state update (cf. Equation (4)) considers an ideal scenario without noise. However, in real-world applications, a robot's internal noise, such as sensor noise or actuator errors, often exists and introduces uncertainty. From a modeling perspective, both types of noise can be incorporated into the disturbance term $\varpi(\tau)$ without introducing additional variables. The impact of these noises will then be propagated to the predicted control cost to inform pose optimization. In terms of implementation, actuator errors can be compensated through closed-loop control. Sensor noise, however, is more difficult to address, as it corrupts the measurements and prevents the system from accurately observing its true state. Developing filtering techniques to mitigate the impact of sensor noise will be the focus of our future work.

4. Main Result

In this section, we first present the theoretical foundation of how our framework estimates the impact of disturbances with derivations of the optimal control law. Then, we detail the methodology of generating a candidate set of feasible joint configurations using a Conditional Variational Autoencoder (CVAE).

4.1. Optimal Control Law:

In the problem formulation (5), the control input sequence $\boldsymbol{u}(0:H-1)$ depends on the input matrix $B(\bar{\boldsymbol{\theta}})$, which itself is a function of the robot joint configuration $\bar{\boldsymbol{\theta}}$. A direct solution that jointly optimizes both $\boldsymbol{u}(0:H-1)$ and $\bar{\boldsymbol{\theta}}$ is often not tractable because of highly non-linear and non-analytic dependence on $\bar{\boldsymbol{\theta}}$. To handle this challenge, we adopt a two-step iterative approach:

1. We first generate a set of candidate joint angle configurations around $\boldsymbol{\theta}_0$. Then, for each candidate pose $\bar{\boldsymbol{\theta}}$ in this set, we compute the sequence of optimal control inputs $\boldsymbol{u}^*(0:H-1)$ and associated cost-to-go.
2. We compare the estimated cost-to-go $\mathcal{J}^*(\bar{\boldsymbol{\theta}})$ for each configuration in the candidate set and choose the one that yields the minimum cost-to-go.

To achieve our goal, we first need to derive an optimal control law, and for that, we need error dynamics. To obtain the error dynamics, we subtract $\boldsymbol{r}(k+1)$ from Equation (4):

$$\begin{aligned} \boldsymbol{s}(k+1) - \boldsymbol{r}(k+1) &= \boldsymbol{s}(k) - \boldsymbol{r}(k) + \boldsymbol{r}(k) - \boldsymbol{r}(k+1) + \bar{B}\boldsymbol{u}(k) \\ \Rightarrow \boldsymbol{e}(k+1) &= \boldsymbol{e}(k) + \bar{B}\boldsymbol{u}(k) + \tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\varpi(k+1) \end{aligned} \tag{6}$$

where we use $\bar{B} = B(\bar{\boldsymbol{\theta}})$, which is the input matrix evaluated at the chosen pose $\bar{\boldsymbol{\theta}}$. $\boldsymbol{e}(k) = \boldsymbol{s}(k) - \boldsymbol{r}(k)$ is the tracking error. We assume that optimal cost-to-go has the following form:

$$\mathcal{J}^*(\boldsymbol{e}(k), k) = \|\boldsymbol{e}(k)\|^2_{P(k)} + 2\boldsymbol{e}(k)^\top q(k) + c(k) \tag{7}$$

where $P(k) \in \mathbb{R}^{6\times 6}, q(k) \in \mathbb{R}^6$, and $c(k) \in \mathbb{R}$ are unknown quantities to be determined.

Theorem 1. *Suppose the optimal control input* $\boldsymbol{u}^*$ *for* (5) *yields a cost-to-go function in the quadratic form of* (7)*. Then for a given pose* $\bar{\theta}$ *and its corresponding input matrix* $\bar{B}$*, the quantities* $P(k)$*,* $q(k)$*, and* $c(k)$ *are obtained using a disturbance-aware Discrete Algebraic Riccati Equation (DARE). Letting* $M = R + \bar{B}^\top P(k+1)\,\bar{B}$*,*

$$P(k) = Q + P(k+1) \;-\; P(k+1)\,\bar{B}\,M^{-1}\,\bar{B}^\top\,P(k+1), \tag{8a}$$

$$\begin{aligned} q(k) = {} & q(k+1) + P(k+1)\,\big[\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\,\alpha_\varpi\big] \\ & -\; P(k+1)\,\bar{B}\,M^{-1}\,\bar{B}^\top\,\Big(P(k+1)\,\big[\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\,\alpha_\varpi\big] + q(k+1)\Big), \end{aligned} \tag{8b}$$

$$\begin{aligned} c(k) = {} & c(k+1) + \big\|\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1)\big\|^2_{P(k+1)} + Tr\Big(\Sigma_\varpi\,D^\top\,P(k+1)\,D\Big) \\ & -\; \big\|P(k+1)\,\big[\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\,\alpha_\varpi\big] + q(k+1)\big\|^2_{\bar{B}\,M^{-1}\,\bar{B}^\top} \\ & +\; 2\,\big(\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\,\alpha_\varpi\big)^\top\,\big(P(k+1)\,[-D\,\alpha_\varpi] + q(k+1)\big), \end{aligned} \tag{8c}$$

with the terminal conditions

$$P(H) \;=\; Q, \quad q(H) \;=\; 0, \quad c(H) \;=\; \kappa\,\|\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}_0\|_2^2 \;+\; w_\theta\,\exp\Big(-\,\big\|\boldsymbol{\theta}^{\lim} - \bar{\boldsymbol{\theta}}\big\|_2^2\Big).$$

The corresponding optimal control input has the form

$$\boldsymbol{u}^*(k) = -M^{-1}B^\top\Big(P(k+1)\big(\boldsymbol{e}(k) + \tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\alpha_\varpi\big) + q(k+1)\Big) \tag{9}$$

Proof of Theorem 1. Combining the Equations (6) and (7),

$$\begin{aligned} & \mathcal{J}^*(\boldsymbol{e}(k),k) = \min_{\boldsymbol{u}(k)} \mathcal{J}(\boldsymbol{e}(k),k)) \\ = {} & \min_{\boldsymbol{u}(k)}\Big[\mathbb{E}_\varpi\big[\|\boldsymbol{e}(k)\|^2_Q + \|\boldsymbol{u}(k)\|^2_R + \mathcal{J}^*(\boldsymbol{e}(k+1)),k+1)\big]\Big] \\ = {} & \min_{\boldsymbol{u}(k)}\Big[\big[\|\boldsymbol{e}(k)\|^2_Q + \|\boldsymbol{u}(k))\|^2_R \\ & + \|\boldsymbol{e}(k) + \bar{B}\boldsymbol{u}(k) + \tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1)\|^2_{P(k+1)} \\ & + \mathrm{Tr}(\Sigma_\varpi D^\top P(k+1)D) \\ & + 2(\boldsymbol{e}(k) + \bar{B}\boldsymbol{u}(k) + \tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\alpha_\varpi)^\top P(k+1)(-D\alpha_\varpi) \\ & + 2(\boldsymbol{e}(k) + \bar{B}\boldsymbol{u}(k) + \tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\alpha_\varpi)^\top q(k+1) + c(k+1)\big]\Big] \end{aligned} \tag{10}$$

where $\mathbb{E}(\varpi(k+1)) = \alpha_\varpi$. Because the control input is optimized to minimize the cost at each step, the optimality condition yields the following:

$$\frac{\partial \mathcal{J}^*(\boldsymbol{e}(k),k)}{\partial \boldsymbol{u}(k)} = R\boldsymbol{u}(k) + \bar{B}^\top P(k+1)(\boldsymbol{e}(k) + \bar{B}\boldsymbol{u}(k) + \tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1)) + \bar{B}^\top q(k+1) = 0. \tag{11}$$

By solving this, we obtain the optimal control law as

$$\boldsymbol{u}^*(k) = -(R + \bar{B}^\top P(k+1)\bar{B})^{-1}B^\top(P(k+1)(\boldsymbol{e}(k) + \tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\alpha_\varpi) + q(k+1))$$

Let

$$d = \tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\alpha_\varpi, \qquad M = R + \bar{B}^\top P(k+1)\bar{B},$$
$$\mathcal{M} = \bar{B}^\top(P(k+1)(\boldsymbol{e}(k) + d) + q(k+1)).$$

By expanding (10) and injecting $\boldsymbol{u}^*(k) = -M^{-1}\mathcal{M}$ into (10), one has

$$\begin{aligned}\mathcal{J}^*(\boldsymbol{e}(k),k) =& \boldsymbol{e}(k)^\top (Q + P(k+1))\boldsymbol{e}(k) \\ &+ 2\boldsymbol{e}^\top (P(k+1)d + q(k+1)) \\ &+ (M^{-1}\mathcal{M})^\top M(M^{-1}\mathcal{M}) \\ &- 2(M^{-1}\mathcal{M})^\top \mathcal{M} + \mathcal{Z} \\ =& \boldsymbol{e}(k)^\top (Q + P(k+1))\boldsymbol{e}(k) \\ &+ 2\boldsymbol{e}^\top (P(k+1)d + q(k+1)) \\ &- \mathcal{M}^\top M^{-1}\mathcal{M} + \mathcal{Z} \end{aligned} \tag{12}$$

where $\mathcal{Z} = \mathrm{Tr}(\Sigma_\omega D^\top P(k+1)D) + (\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1))^\top P(k+1)(\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1))$ $+ 2d^\top P(k+1)(-D\alpha_\omega) + 2d^\top q(k+1) + c(k+1)$. Expanding $\mathcal{M}^\top M^{-1}\mathcal{M}$ yields

$$\begin{aligned}&\left[P(k+1)(\boldsymbol{e}(k)+d) + q(k+1)\right]^\top \bar{B}M^{-1}\bar{B}^\top \left[P(k+1)(\boldsymbol{e}(k)+d) + q(k+1)\right] \\ &\quad = \boldsymbol{e}(k)^\top (P(k+1)\bar{B})M^{-1}(\bar{B}^\top P(k+1))\boldsymbol{e}(k) \\ &\quad\quad + 2\boldsymbol{e}(k)^\top (P(k+1)\bar{B})M^{-1}(\bar{B}^\top P(k+1)d + \bar{B}^\top q(k+1)) \\ &\quad\quad + (\bar{B}^\top P(k+1)d + \bar{B}^\top q(k+1))^\top M^{-1}(\bar{B}^\top P(k+1)d + \bar{B}^\top q(k+1)) \end{aligned} \tag{13}$$

Putting this back into (12) gives

$$\begin{aligned}\mathcal{J}^*(\boldsymbol{e}(k),k) =& \boldsymbol{e}(k)^\top (Q + P(k+1) - P(k+1)\bar{B}M^{-1}\bar{B}^\top P(k+1)))\boldsymbol{e}(k) \\ &+ 2\boldsymbol{e}(k)^\top (P(k+1)d + q(k+1) \\ &- (P(k+1)\bar{B})M^{-1}(\bar{B}^\top (P(k+1)d + \bar{B}^\top q(k+1))) \\ &- (\bar{B}^\top (P(k+1)d + \bar{B}^\top q(k+1))^\top M^{-1}(\bar{B}^\top (P(k+1)d + \bar{B}^\top q(k+1)) + \mathcal{Z} \end{aligned} \tag{14}$$

Comparing (14) with (7), we obtain

$$\begin{aligned} P(k) &= Q + P(k+1) - P(k+1)\bar{B}M^{-1}\bar{B}^\top P(k+1) \\ q(k) &= q(k+1) + P(k+1)(\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\alpha_\omega) \\ &\quad - P(k+1)\bar{B}M^{-1}\bar{B}^\top (P(k+1)(\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\alpha_\omega) + q(k+1)) \\ c(k) &= c(k+1) + \|\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1)\|^2_{P(k+1)} \\ &\quad + \mathrm{Tr}(\Sigma_\omega D^\top P(k+1)D) \\ &\quad - \|P(k+1)(\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\alpha_\omega) + q(k+1))\|^2_{\bar{B}M^{-1}\bar{B}^\top} \\ &\quad + 2(\tilde{\boldsymbol{r}}(k) - \tilde{\boldsymbol{r}}(k+1) - D\alpha_\omega)^\top (P(k+1) - D\alpha_\omega + q(k+1)) \end{aligned}$$

where $\mathcal{J}^*(\boldsymbol{e}(H), k = H)$, $P(H)$, $q(H)$, and $c(H)$ are obtained. □

From Theorem 1, we observe that $c(k)$ is not a part of the expression for $\boldsymbol{u}^*(k)$ in (9). However, the term $c(k)$ is essential in computing the optimal cost-to-go $\mathcal{J}^*$. Therefore, when we compare different pose candidates $\bar{\boldsymbol{\theta}} \in \Theta$, each candidate yields a different set of P, q, and c values. This results in a different cost $\mathcal{J}^*(\bar{\boldsymbol{\theta}})$ for each candidate. Among all, we select the pose with the lowest $\mathcal{J}^*$.

Specifically, our proposed two-step iterative process framework, described in Algorithm 1, addresses the challenge that arises from the strong non-linear and non-analytic dependence on $\bar{\boldsymbol{\theta}}$. For this reason, finding an optimized $\bar{\boldsymbol{\theta}}$ can be computationally infeasible. Hence, we use a sample-based approach for generating a set of candidate poses Θ. In our approach, we use a deep learning method that is discussed in the following Section 4.2 to

obtain each candidate pose $\bar{\boldsymbol{\theta}} \in \Theta$. A similar approach of using learning methods and a sampling-based approach have been used in [42,60], respectively.

Algorithm 1 Disturbance-Aware MPC Tracking with Pose Optimization

Require: Reference trajectory $\tilde{\boldsymbol{r}}(0:H)$, disturbance parameters $\alpha_\omega, \Sigma_\omega$, initial pose $\boldsymbol{\theta}_0$
Ensure: Sequence of optimal inputs $\boldsymbol{u}^*(0:H-1)$ and optimized pose control $\bar{\boldsymbol{u}}$
Trajectory Formulation
Formulate trajectory $\boldsymbol{r}(k=0:H)$ through (1).
Create set of candidate joint configurations Θ using CVAE (cf. Algorithm 2).
Step 1: Evaluation for Each Candidate Pose
for all $\bar{\boldsymbol{\theta}} \in \Theta$ **do**
  Compute the Jacobian matrix $J(\bar{\boldsymbol{\theta}}) = \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}}$.
  Compute input matrix $B(\bar{\boldsymbol{\theta}})$.
  Solve (5) by the disturbance-aware DARE solution (see *Theorem 1*).
  Compute the optimal cost $\mathcal{J}^*(\bar{\boldsymbol{\theta}})$ via (7).
end for
Step 2: Pose Optimization and Control Computation
Determine the optimal pose $\bar{\boldsymbol{\theta}}^* = \arg\min_{\bar{\boldsymbol{\theta}} \in \Theta} \mathcal{J}^*(\bar{\boldsymbol{\theta}})$.
Compute the pose control input as $\bar{\boldsymbol{u}} = \frac{\bar{\boldsymbol{\theta}}^* - \boldsymbol{\theta}_0}{\tau}$.
Reuse the DARE solution for $\bar{\boldsymbol{\theta}}^*$ to compute $\boldsymbol{u}^*(0:H-1)$ using Equation (9).
Apply the combined control input $\bar{\boldsymbol{u}} + \boldsymbol{u}^*(0)$ to the robot.

For each candidate pose $\bar{\boldsymbol{\theta}} \in \Theta$, we compute the associated Jacobian, input matrix $B(\bar{\boldsymbol{\theta}})$ and then evaluate the estimated optimal cost $\mathcal{J}^*(\bar{\boldsymbol{\theta}})$ using Theorem 1. Because each candidate in Θ is processed independently, this computation can be parallelized to reduce runtime. We can achieve this if the number of candidates in Θ is less than the number of computing threads of the test computer. Once the cost $\mathcal{J}^*(\bar{\boldsymbol{\theta}})$ is obtained for every candidate, we choose the $\bar{\boldsymbol{\theta}}$ that yields the smallest cost ($\bar{\boldsymbol{\theta}}^*$) and apply the corresponding optimal control sequence computed via Equation (9). The ideal implementation of the algorithm would be as follows: (i) move the robot pose to $\bar{\theta}^*$ and then (ii) apply the arm control inputs $\boldsymbol{u}^*$. However, such execution can be challenging in real-time settings. Because of this, we merge these two controls into a single step by adding the pose optimization command and the manipulator control command vectorially. Based on the triangle law of vector addition, this combined approach does not increase the total cost relative to the separated steps. Moreover, if the difference $(\bar{\boldsymbol{\theta}}^* - \boldsymbol{\theta}_0)$ is very small, the potential difference in cost is negligible in practice.

4.2. Generation of Candidate Set with Conditional Variational Autoencoder (CVAE):

As discussed in Section 4.1, directly finding $\bar{\boldsymbol{\theta}}^*$ is challenging because of the input matrix's non-linear and non-analytic dependence on $\bar{\boldsymbol{\theta}}$. To overcome this, we generate a set of candidate poses that yield the same end-effector pose. We want these configurations to be closer to the initial joint angle combination $\boldsymbol{\theta}_0$, since larger deviation from $\boldsymbol{\theta}_0$ incurs a higher cost as formulated in (5). However, finding multiple configurations for a given end-effector pose is challenging, as traditional inverse kinematics often yields a single configuration for a certain end-effector pose. To address this, we use a CVAE model to obtain multiple joint angle combinations where each of them gives the same end-effector pose and is close to the initial configuration $\boldsymbol{\theta}_0$.

Algorithm 2 Training Algorithm for Conditional Variational Autoencoder (CVAE)

Require: Joint angle data $\{\boldsymbol{\theta}_i\}_{i=1}^{N}$, end-effector pose data $\{\boldsymbol{s}\}_{i=1}^{N}$, initial configurations $\{\boldsymbol{\theta}_{0i}\}_{i=1}^{N}$, learning rate α, weight coefficients λ_{KL}, λ_{pose}, number of epochs E, and batch size $\mathbf{B}$.

Ensure: Trained CVAE model parameters.

for epoch = 1 to E **do**
  for each mini-batch of $\{\boldsymbol{\theta}, \boldsymbol{s}\}$ of size $\mathbf{B}$ **do**
    Forward Pass:
    Compute encoder input: $\mathbf{E}_{\text{in}} \leftarrow [\boldsymbol{\theta}, \boldsymbol{s}]$.
    Compute latent variables: $(\mu, \log \sigma^2) \leftarrow \text{Encoder}(\mathbf{E}_{\text{in}})$.
    Compute $\sigma \leftarrow \exp\left(\frac{1}{2} \log \sigma^2\right)$.
    Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
    Obtain latent vector: $\xi \leftarrow \mu + \sigma \odot \epsilon$.
    Compute decoder input: $\mathbf{D}_{\text{in}} \leftarrow [\xi, \boldsymbol{s}]$.
    Reconstruct joint angles: $\hat{\theta} \leftarrow \text{Decoder}(\mathbf{D}_{\text{in}})$.
    Loss Computation:
    Total loss function: $L_{\text{total}} \leftarrow L_{\text{recon}} + \lambda_{\text{KL}} L_{\text{KL}} + \lambda_{\text{pose}} L_{\text{pose}}$
    Backpropagation and Optimization:
    Backpropagate L_{total}.
    Update model parameters ρ_e and ρ_d using the NAdam optimizer
  end for
end for

The CVAE model has an encoder, a reparameterization step, and a decoder. The encoder maps an input joint angle vector $\boldsymbol{\theta}$ to a latent variable $\xi \in \mathbb{R}^l$. This is conditioned on the end-effector pose $\boldsymbol{s} \in \mathbb{R}^6$. We model this mapping as a multivariate Gaussian distribution:

$$\mathcal{Q}_{\rho_e}(\xi \mid \boldsymbol{\theta}, \boldsymbol{s}) = \mathcal{N}\left(\xi; \mu_{\rho_e}(\boldsymbol{\theta}, \boldsymbol{s}), \text{diag}(\sigma^2_{\rho_e}(\boldsymbol{\theta}, \boldsymbol{s}))\right) \tag{15}$$

where ρ_e denotes the encoder's trainable parameters, and μ_{ρ_e} and $\log \sigma^2_{\rho_e}$ are the mean and log-variance of the latent distribution, respectively. To sample ξ while enabling gradient-based optimization, we employ the reparameterization trick:

$$\xi \leftarrow \mu_{\rho_e} + \sigma_{\rho_e} \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{16}$$

where $\odot$ represents element-wise multiplication. The decoder then reconstructs a joint angle vector $\hat{\boldsymbol{\theta}} \in \mathbb{R}^8$ from ξ, conditioned on $\boldsymbol{s}$ and $\boldsymbol{\theta}$:

$$\mathcal{P}_{\rho_d}(\hat{\boldsymbol{\theta}} \mid \xi, \boldsymbol{s}, \boldsymbol{\theta}) = \mathcal{N}\left(\hat{\theta}; \mu_{\rho_d}(\xi, \boldsymbol{s}, \boldsymbol{\theta}), \text{diag}(\sigma^2_{\rho_d})\right) \tag{17}$$

where ρ_d represents the decoder's parameters. Each reconstructed $\hat{\boldsymbol{\theta}}$ is a candidate for inclusion in Θ.

Loss Function: We train the model by minimizing a composite loss function. This function comprises (i) *Reconstruction loss* (L_{recon}) that measures the discrepancy between the reconstructed joint configurations $\hat{\boldsymbol{\theta}}$ and the original combinations $\boldsymbol{\theta}$; (ii) *Kullback–Leibler (KL) Divergence loss* (L_{KL}) that regularizes the latent space to follow a standard normal distribution; and (iii) *Pose Similarity loss* (L_{pose}) that encourages consistency between the pose derived from the reconstructed angles and the desired end-effector pose. Putting all these terms together, the total loss is

$$L_{\text{total}} = L_{\text{recon}} + \lambda_{\text{KL}} L_{\text{KL}} + \lambda_{\text{pose}} L_{\text{pose}}$$
$$= \frac{1}{\mathbf{B}} \sum_{i=1}^{\mathbf{B}} \left[\|\hat{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_i\|^2 - \frac{\lambda_{\text{KL}}}{2} \sum_{j=1}^{l} \left(1 + \log \sigma_{ij}^2 - \mu_{ij}^2 - \sigma_{ij}^2\right) + \lambda_{\text{pose}} \|\hat{\boldsymbol{s}}_i - \boldsymbol{s}_i\|^2 \right] \tag{18}$$

where $\mathbf{B}$ is the batch size, and λ_{KL} and λ_{pose} are the hyperparameters controlling the relative importance of each term.

Network Architecture and Training: We generate the training data for CVAE by randomly taking joint angle combinations and use forward kinematics to obtain the end-effector pose with the Fetch's URDF. Since all data is generated using the kinematic model and satisfies the joint limit constraints, the output of the CVAE model is therefore trained to meet these constraints. The URDF of the Fetch Mobile Manipulator can be found at https://github.com/ZebraDevs/fetch_ros/blob/ros1/fetch_description/robots/fetch.urdf (accessed on 16 June 2025). In the rare case where infeasible candidates are generated, we will discard the output during selection for the candidate set. We train the encoder and decoder parameters (ρ_e, ρ_d) by minimizing L_{total} using the Nesterov-accelerated Adaptive Moment Estimation (NAdam) optimizer [61]:

$$\rho_e \leftarrow \rho_e - \alpha \nabla_{\rho_e} L_{\text{total}}, \quad \rho_d \leftarrow \rho_d - \alpha \nabla_{\rho_d} L_{\text{total}} \tag{19}$$

where α is the learning rate. We generate the dataset by performing forward kinematics on an 8-DoF configuration (7-DoF manipulator and 1-DoF mobile base, considering heading angle only) to obtain pairs of joint angle vectors $\boldsymbol{\theta} \in \mathbb{R}^8$ and the corresponding end-effector pose $\boldsymbol{s}$. Algorithm 2 summarizes our training procedure, and a block diagram is illustrated in Figure 3. For presentation simplicity, we use $(\phi, \vartheta_2, \ldots, \vartheta_8)$ to represent $\boldsymbol{\theta}$ and $(\hat{\phi}, \hat{\vartheta}_2, \ldots, \hat{\vartheta}_8)$ to represent $\hat{\boldsymbol{\theta}}$ in Figure 3.

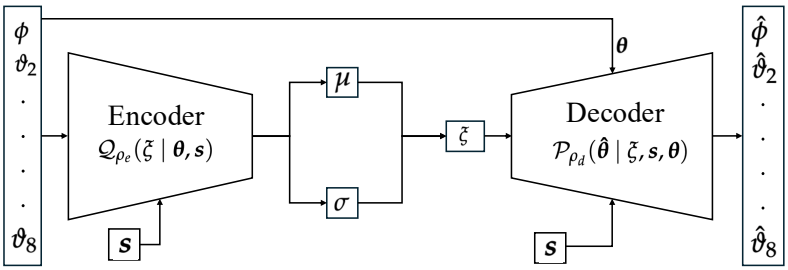


Figure 3. Block diagram model of CVAE for pose optimization.

5. Experiments

In this section, we first show the efficiency of the CVAE in generating multiple joint configurations while maintaining the same end-effector pose. We then evaluate our proposed algorithm *Pose Optimization with Human Uncertainty* (PO-HU), which represents the DARC framework, through simulation experiments in the Gazebo environment using a Fetch robot [56].

5.1. Evaluation of CVAE Method

As discussed earlier, there are three parts in CVAE: encoder network, reparameterization trick, and decoder network. In the encoder network, we set up the model as a 14-dimensional input layer ($\boldsymbol{\theta} \in \mathbb{R}^8$ and $\boldsymbol{s} \in \mathbb{R}^6$) that maps to a hidden layer of 128 neurons. This 14-dimension consists of an 8-dimensional $\boldsymbol{\theta}$ and a 6-dimensional end-effector pose $\boldsymbol{s}$. The second layer maps the hidden layer to 64 neurons. This outputs the mean

and log-variance of the latent distribution and maps from 64 to the $l = 20$ dimensional latent space. We then use the reparameterization trick part to allow gradient backpropagation through the sampling process. In the decoder network, the first layer maps the $(20 + 14)$-dimensional input to 64 neurons. Layer 2 maps the 64-dimensional hidden layer to 128 neurons, and this outputs the reconstructed joint angle with a dimension of 8. We set $\mathbf{B} = 64$ and $\alpha = 10^{-4}$.

Figure 4a shows the training and validation loss curves. We can observe the convergence of the loss from this plot. Figure 4b depicts the training and validation accuracy of the model. We evaluate the trained CVAE by feeding it a specific end-effector pose $\boldsymbol{s}$ along with its associated joint angles $\boldsymbol{\theta}$, represented by the row "*Given*" in Table 1. Using CVAE, we first generate 100 samples and then choose 20 samples for the candidate set. To choose these 20 samples, we consider the lowest reconstruction error, pose deviation, satisfaction of joint angle limits, and an error tolerance setting of the Mean Squared Error (MSE) being within 0.02 for the end-effector pose. If any of the generated joint configurations yields an MSE higher than 0.02 or violates the joint angle limits, that configuration is dropped from the candidate set. We observe that in sample 13 of Table 1, ϑ_8 is 1.696 rad, but the given ϑ_8 is 2.1 rad. However, the generated end-effector pose is very close to the given pose, with an MSE of only 0.00831. For that reason, it is still a valid candidate. Yet, these higher differences in generated joint configurations may result in higher estimated costs due to the "Pose Optimization" and "Joint Limit Penalty" terms in Equation (5). Thus, these configurations have a higher chance of being discarded when we select a configuration based on the lowest estimated costs for pose optimization.

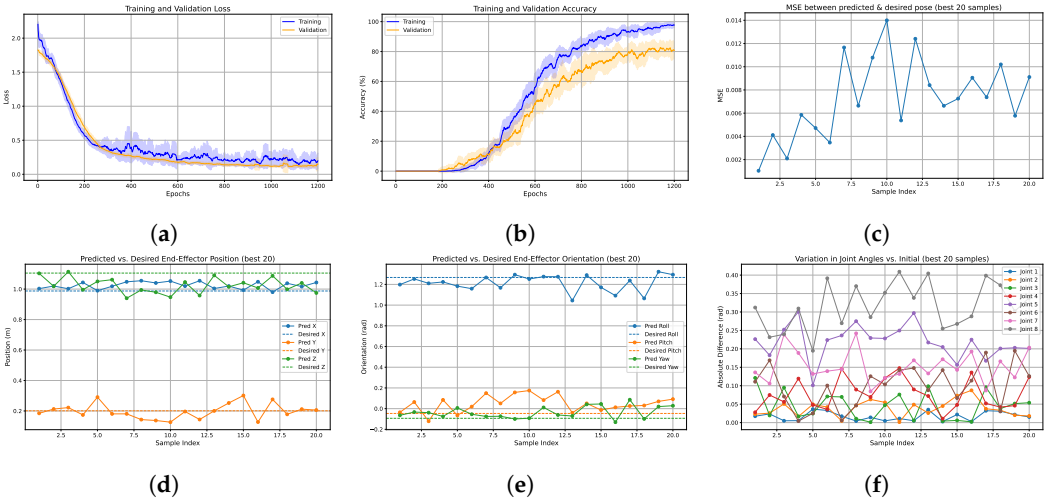


Figure 4. Visualization of CVAE model training metrics and outputs: (**a**–**f**) show various training and evaluation plots including loss, accuracy, MSE, predicted vs. desired outputs, and joint variations. (**a**) Training and validation loss of the CVAE model. (**b**) Training and validation accuracy of the CVAE model. (**c**) MSE between predicted and desired end-effector pose. (**d**) Predicted vs. desired end-effector position. (**e**) Predicted vs. desired end-effector orientation. (**f**) Variation in joint angles from the given joint angle.

Overall, Table 1 illustrates that, for nearly the same end-effector pose, the CVAE can produce different feasible joint angle combinations.

Figure 4c shows the MSE between each predicted end-effector pose and the desired pose. It is observed that the error is very small and within a range of 0.002 to 0.014. Figure 4d,e depict the end-effector position and orientation results, respectively. These figures show that the generated angles yield very similar end-effector poses. Finally, Figure 4f visualizes how the generated angles differ from the given joint angles. These results demonstrate that the proposed approach effectively produces diverse joint angle

solutions. Each of them satisfies the end-effector pose condition, along with exploring alternative configurations close to the initial one. In addition, we compare the averaged execution time and MSE with variations over 100 trials for generating the samples in Table 2. We observe that, by increasing the number of samples, we reduce the MSE. Although this increases the computation time, it remains within an acceptable range for real-time implementation. The low latency stems from using a neural network model that has already been trained offline, and we only perform the forward pass to produce the joint configurations.

Table 1. Joint angles (8 DOF), end-effector positions, orientations, and MSE between generated and given joint angles for 20 samples.

Sample	Joint Angles (rad)								Position (m)			Orientation (rad)			MSE
	ϕ	ϑ_2	ϑ_3	ϑ_4	ϑ_5	ϑ_6	ϑ_7	ϑ_8	s_x	s_y	s_z	s_{roll}	s_{pitch}	s_{yaw}	
Given	**0.0**	**0.2**	**−0.18**	**−0.5**	**−0.68**	**−0.24**	**0.95**	**2.1**	**0.987**	**0.201**	**1.105**	**1.266**	**−0.046**	**−0.093**	**-**
1	0.017	0.177	−0.301	−0.528	−0.454	−0.129	0.814	1.788	1.003	0.185	1.103	1.198	−0.036	−0.064	0.00101
2	0.023	0.174	−0.199	−0.575	−0.497	−0.071	0.844	1.868	1.020	0.213	1.018	1.253	0.064	−0.033	0.00411
3	0.005	0.251	−0.275	−0.444	−0.428	−0.311	0.711	1.860	1.002	0.222	1.114	1.210	−0.122	−0.039	0.00586
4	−0.005	0.217	−0.197	−0.381	−0.379	−0.236	0.761	1.791	1.044	0.174	0.996	1.224	0.084	−0.075	0.00581
5	0.037	0.250	−0.155	−0.548	−0.579	−0.265	0.818	1.905	0.990	0.291	1.050	1.183	−0.065	0.006	0.00492
6	0.033	0.158	−0.251	−0.464	−0.456	−0.140	0.810	1.709	1.018	0.181	1.062	1.159	0.019	−0.054	0.00340
7	0.017	0.195	−0.110	−0.355	−0.444	−0.246	0.805	1.830	1.048	0.181	0.940	1.267	0.149	−0.075	0.01162
8	0.004	0.153	−0.169	−0.410	−0.405	−0.193	0.708	1.730	1.055	0.143	0.994	1.167	0.050	−0.074	0.00661
9	0.014	0.137	−0.181	−0.431	−0.451	−0.114	0.865	1.814	1.040	0.137	0.979	1.294	0.157	−0.100	0.01077
10	−0.005	0.145	−0.133	−0.379	−0.452	−0.136	0.829	1.748	1.053	0.127	0.947	1.252	0.175	−0.092	0.01397
11	0.011	0.199	−0.256	−0.352	−0.430	−0.098	0.818	1.691	1.020	0.196	1.046	1.275	0.082	0.011	0.00531
12	0.005	0.151	−0.175	−0.410	−0.383	−0.092	0.781	1.762	1.055	0.144	0.958	1.274	0.163	−0.061	0.01237
13	0.035	0.174	−0.279	−0.572	−0.463	−0.152	0.817	1.696	1.004	0.201	1.090	1.045	−0.040	−0.071	0.00831
14	−0.003	0.245	−0.184	−0.489	−0.475	−0.098	0.778	1.845	1.020	0.252	1.015	1.289	0.051	0.041	0.00661
15	0.022	0.273	−0.186	−0.548	−0.523	−0.174	0.807	1.832	0.993	0.301	1.041	1.171	−0.013	0.045	0.00721
16	0.003	0.112	−0.178	−0.636	−0.455	−0.126	0.757	1.812	1.048	0.127	1.007	1.092	0.013	−0.131	0.00900
17	0.032	0.238	−0.276	−0.448	−0.512	−0.050	0.863	1.701	0.980	0.276	1.087	1.237	0.026	0.086	0.00734
18	0.031	0.166	−0.142	−0.458	−0.479	−0.270	0.784	1.727	1.038	0.178	0.999	1.065	0.030	−0.099	0.01009
19	0.022	0.180	−0.232	−0.454	−0.477	−0.045	0.827	1.798	1.017	0.212	1.040	1.322	0.070	0.020	0.00576
20	0.015	0.182	−0.126	−0.376	−0.479	−0.114	0.747	1.762	1.043	0.206	0.975	1.294	0.092	0.028	0.00908

Table 2. Comparison of averaged execution time and MSE with standard deviation under varying numbers of generated samples.

No. of Samples	Computation Time (s)	MSE
10	0.0011 ± 0.0001	0.0215 ± 0.0057
30	0.0027 ± 0.0004	0.0177 ± 0.0049
60	0.0047 ± 0.0011	0.0125 ± 0.0032
100	0.0061 ± 0.0019	0.0103 ± 0.0015
200	0.0112 ± 0.0031	0.0099 ± 0.0010

5.2. Evaluation of DARC Framework

We aim to validate the effectiveness of DARC under various levels of human disturbances and trajectories. We assumed that human uncertainties are mathematically equivalent to robot actuation disturbances, as discussed in Section 3.2. To model human uncertainties, we use $\varpi(k) \sim \mathcal{N}(\alpha_\varpi, \Sigma_\varpi)$, where $\alpha_\varpi = [0.0035, 0.0055, 0.0027]^\top$ (in meters) and $\Sigma_\varpi = \eta \cdot \text{diag}(0.011, 0.02, 0.013)$ (in meters) and $\eta \in \{0.3,\ 0.6\}$ represents the strength of the disturbances. This reflects the tendency for disturbances along the y-axis to be more pronounced than those in the x and z directions.

To evaluate the algorithm, we employ two representative trajectories, which are trajectory $\mathcal{A}$ and $\mathcal{B}$. Each of them is discretized with 500 waypoints and a time-step interval of $\Delta t = 0.1$ s. In one set of experiments, we assume that the future human poses

are unknown to the robot. To address this, we incorporate an LSTM network for pose prediction (cf. Appendix A). The algorithm predicts the next six poses based on the ten previous poses. The experiments leveraging the LSTM-based prediction are denoted by $\mathcal{A}_{\mathcal{P}}$ and $\mathcal{B}_{\mathcal{P}}$. We consider a second set of experiments where we assume the future human pose to be fully known a priori, though still subject to disturbances. The corresponding experiments are denoted by $\mathcal{A}_{\mathcal{N}}$ and $\mathcal{B}_{\mathcal{N}}$.

We adopt an MPC strategy as described in Section 3.3 and in Algorithm 1, where the planning is repeated at every discrete time step. Within each planning horizon, we sample twenty candidate robot poses, i.e., $|\Theta| = 20$. To solve the optimization problem (5), we set $H = 4$ for predictive experiments $\mathcal{A}_{\mathcal{P}}$ and $\mathcal{B}_{\mathcal{P}}$, and $H = 8$ for the experiments $\mathcal{A}_{\mathcal{N}}$ and $\mathcal{B}_{\mathcal{N}}$. In the predictive cases, we forecast only six future steps. We chose $H = 4$ (< 6) to keep the planning horizon shorter than the prediction window. For $\mathcal{A}_{\mathcal{N}}$ and $\mathcal{B}_{\mathcal{N}}$, although the robot knows the full reference trajectory, setting a very long horizon would increase computational time and slow the solver. Thus, we select a moderate value $H = 8$, which is double the predictive horizon. We examine two distinct settings for the tracking-error weight Q—i.e., $Q = 1500 \cdot I_6$ and $Q = 1100 \cdot I_6$. These configurations allow us to investigate the impact of reweighting the tracking error cost on the overall performance. A larger Q emphasizes pose tracking by rescaling the pose error to a magnitude comparable to the control inputs (unit-balancing). Moreover, it ensures the MPC actively suppresses the pose errors that might twist the object. We set $R = I_9$ to penalize all nine actuators evenly, which avoids over-regularizing of any specific control input. We set $\kappa = w_\theta = 1$, which in our case, is sufficient to penalize changes in joint configurations.

Although the algorithm is designed to minimize an expected cost, we quantify the true cumulative cost over the full trajectory as follows:

$$\mathcal{C}_{\text{total}} = \sum_{t=1}^{T} \Big[\boldsymbol{e}(t)^\top Q\, \boldsymbol{e}(t) + \boldsymbol{u}(t)^\top R\, \boldsymbol{u}(t) + \kappa \left\| \tilde{\boldsymbol{\theta}}^*(t) - \boldsymbol{\theta}(t) \right\|_2^2 \Big], \tag{20}$$

where the first term penalizes the end-effector tracking error, the second term penalizes the control effort, and the third term is related to pose optimization. $T = 500$ denotes the total number of steps.

To evaluate the effectiveness of our proposed PO-HU algorithm with $|\Theta| = 20$, we compare it against five baseline methods: (i) *PO-HU* ($|\Theta| = 1$): single candidate pose optimization, which corresponds to using a traditional inverse kinematics approach that yields only one joint configuration; (ii) *pPO-HU*: periodic pose optimization every five time steps, which reduces computation load while still considering human uncertainty; (iii) *NPO-HU*: no pose optimization but considers human uncertainty while computing the control inputs; (iv) *PO-NHU*: perform pose optimization without considering the impact of human uncertainties; and (v) *NPO-NHU*: neither pose optimization nor human uncertainty is considered in the computation of the control inputs.

Table 3 shows the mean with standard deviation of the total cost $\mathcal{C}_{\text{total}}$ over 100 trials for each combination of trajectory type ($\mathcal{A}_{\mathcal{P}}$, $\mathcal{B}_{\mathcal{P}}$, $\mathcal{A}_{\mathcal{N}}$, $\mathcal{B}_{\mathcal{N}}$), disturbance strength η, and weighting matrix Q. Here, we observe that whenever pose optimization is performed (i.e., PO-HU, PO-NHU, pPO-HU), it achieves a lower mean cost and often reduced standard deviation compared to methods lacking pose optimization (NPO-HU and NPO-NHU). Notably, NPO-NHU yields the highest mean cost with the largest variance as it neither performs pose optimization nor considers human uncertainties. This supports the importance of selecting a better pose for improved performance.

Within pose optimization methods, PO-HU outperforms PO-NHU. This reflects how explicitly accounting for human uncertainty yields advantages in all scenarios. While comparing PO-HU ($|\Theta| = 1$) and PO-HU ($|\Theta| = 20$), we observe that even a single-

candidate version of pose optimization can achieve better results than not performing pose optimization at all, i.e., NPO-HU. However, when more candidate poses are considered ($|\Theta| = 20$), the algorithm better explores alternative joint configurations and achieves the lowest average total cost in all cases. Moreover, the pPO-HU shows that pose optimization at every five steps is still better than never performing pose optimization. Still, there exists a slight performance drop compared to optimizing at every time step.

Table 3. Comparison of C_{total} across different algorithms (mean ± standard deviation). $^{\dagger}, ^{\ddagger}, ^{\S}$ represent larger differences for NPO-HU, PO-NHU, and NPO-NHU, respectively, from PO-HU($|\Theta| = 20$).

Traj	Q	η	PO-HU ($\|\Theta\| = 20$)	PO-HU ($\|\Theta\| = 1$)	pPO-HU ($\|\Theta\| = 20$)	NPO-HU	PO-NHU	NPO-NHU
$\mathcal{A}_{\mathcal{P}}$	$1500I_6$	0.3	879.02 ± 32.1	1012.59 ± 43.2	993.12 ± 38.7	2467.55 ± 158.4	1289.69 ± 52.1	8421.79 ± 402.1 §
		0.6	1008.45 ± 42.5	1289.51 ± 60.1	1217.44 ± 54.8	4664.03 ± 279.3 †	1855.11 ± 83.7 ‡	9107.12 ± 445.6 §
	$1100I_6$	0.3	911.56 ± 29.8	986.14 ± 38.5	957.71 ± 35.2	2395.09 ± 142.1	1051.24 ± 63.2	7916.18 ± 365.9 §
		0.6	994.99 ± 38.11	1079.52 ± 51.4	1042.36 ± 46.3	3674.88 ± 201.1 †	1102.45 ± 58.4	9363.44 ± 481.2 §
$\mathcal{A}_{\mathcal{N}}$	$1500I_6$	0.3	799.66 ± 26.4	891.38 ± 36.5	851.45 ± 31.8	1271.54 ± 81.2	983.87 ± 56.2	1591.42 ± 102.4
		0.6	871.97 ± 36.1	968.91 ± 46.3	924.18 ± 42.6	1564.94 ± 104.8	1092.47 ± 64.9	3206.33 ± 189.7
	$1100I_6$	0.3	735.71 ± 30.1	809.48 ± 39.1	786.72 ± 35.6	1165.65 ± 75.4	869.09 ± 50.8	1778.19 ± 107.8
		0.6	861.07 ± 36.2	953.16 ± 45.8	902.42 ± 42.9	1334.66 ± 88.2	1013.35 ± 59.4	1819.98 ± 119.2
$\mathcal{B}_{\mathcal{P}}$	$1500I_6$	0.3	852.84 ± 33.6	1091.48 ± 57.1	983.47 ± 43.4	1891.23 ± 127.4	1431.73 ± 71.3	5721.94 ± 314.5 §
		0.6	976.77 ± 43.1	1186.40 ± 61.7	1098.75 ± 54.3	2880.69 ± 162.2 †	1569.67 ± 77.8 ‡	6452.61 ± 338.4 §
	$1100I_6$	0.3	861.766 ± 30.2	1019.72 ± 49.1	971.33 ± 44.5	6407.73 ± 285.1 †	1260.28 ± 61.8	10531.81 ± 490.9 §
		0.6	1001.92 ± 43.4	1218.77 ± 59.1	1179.25 ± 53.7	7243.09 ± 327.8 †	1461.87 ± 70.5	11355.08 ± 517.6 §
$\mathcal{B}_{\mathcal{N}}$	$1500I_6$	0.3	615.81 ± 26.1	701.63 ± 36.3	679.55 ± 33.7	1259.24 ± 92.1	764.59 ± 49.2	2558.78 ± 175.5
		0.6	828.87 ± 38.3	897.28 ± 52.6	866.41 ± 48.1	2483.88 ± 150.7	927.36 ± 62.1	3495.15 ± 198.8
	$1100I_6$	0.3	651.76 ± 32.1	762.99 ± 48.2	711.81 ± 39.5	2214.90 ± 12.3	969.85 ± 59.4	5278.04 ± 261.9 §
		0.6	778.93 ± 40.2	937.81 ± 59.4	894.11 ± 48.9	2758.48 ± 142.6 †	1172.75 ± 60.7	5661.65 ± 297.6 §

In addition, NPO-HU shows better performance than NPO-NHU. This highlights that even without performing pose optimization, and only considering human uncertainty, it can still increase the overall performance. Here, we use specific markers to highlight significant differences with respect to PO-HU. We use "†" to mark large differences for NPO-HU, "‡" for PO-NHU, and "§" for NPO-NHU. The fewer highlighted markers in PO-NHU than in NPO-HU and NPO-NHU imply that pose optimization contributes more to cost reduction than the characterization of human uncertainty.

Overall, the robot achieves better performance in the cases where future human poses are known a priori to the robot (i.e., $\mathcal{A}_{\mathcal{N}}$ and $\mathcal{B}_{\mathcal{N}}$) compared to the cases where the robot uses LSTM to predict the human future poses (i.e., $\mathcal{A}_{\mathcal{P}}$ and $\mathcal{B}_{\mathcal{P}}$). This stems from having more accurate knowledge of human movements than predicting human poses.

Figure 5 visualizes the tracking performance along the x-, y-, and z-axes for trajectories $\mathcal{A}_{\mathcal{N}}$, $\mathcal{B}_{\mathcal{N}}$, $\mathcal{A}_{\mathcal{P}}$, and $\mathcal{B}_{\mathcal{P}}$, with the weighting matrix $Q = 1500 \cdot I_6$ and $\eta = 0.3$ for 10 representative trials. We observe that PO-HU tracks the human trajectory with disturbances more closely, whereas methods lacking pose optimization comparatively deviate more. Empirically, the performance of LSTM is good, as suggested by this figure. We observe that the human trajectories with disturbances in Figure 5c,d are almost similar to those in Figure 5a,b. We validated our trained LSTM model with trajectories $\mathcal{A}_{\mathcal{N}}$ and $\mathcal{B}_{\mathcal{N}}$. The validation accuracy of the model is 90.84% with an MSE loss of 0.0512. If the human follows

a smooth trajectory, it is expected that the model can predict the future movements with high accuracy. Theoretically, the accuracy of the LSTM-based model is not guaranteed. If the human takes sharp turns, then the trajectory prediction accuracy will be lower. Consequently, the control cost obtained from the MPC can be inaccurate, which further impacts the pose selection. However, when the control input is implemented, the tracking accuracy will not be significantly impacted, as we have a closed-loop control. Specifically, while the MPC can plan for six future time steps, the execution applies to only one step.

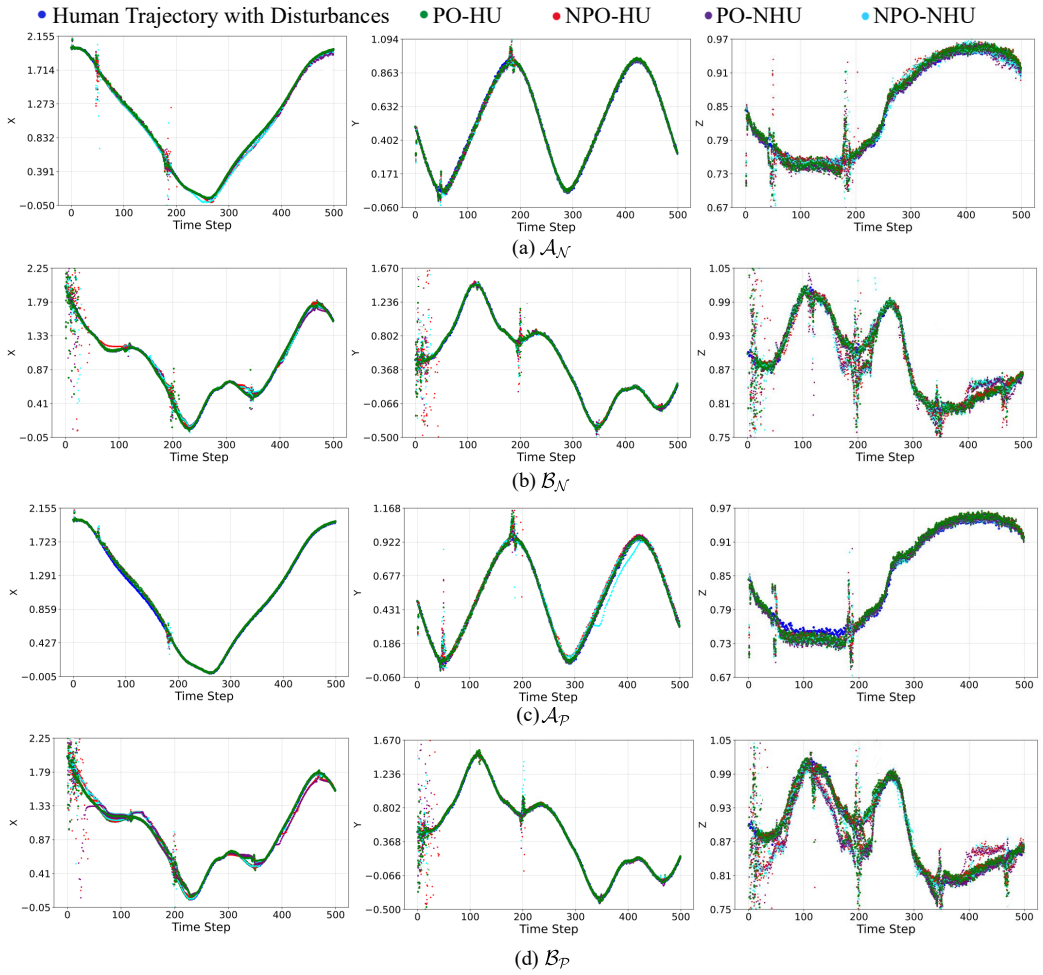


Figure 5. Tracking across trajectories (**a**) $\mathcal{A}_\mathcal{N}$, (**b**) $\mathcal{B}_\mathcal{N}$, (**c**) $\mathcal{A}_\mathcal{P}$, and (**d**) $\mathcal{B}_\mathcal{P}$ on the x-, y-, and z-axes, where $Q = 1500 \cdot I_6$, $R = I_9$, and $\eta = 0.3$ for 10 representative trials.

Figure 6 depicts the instantaneous (left) and accumulated cost (right) for trajectory $\mathcal{A}_\mathcal{N}$ with $Q = 1500 \cdot I_6$, $R = I_9$, and $\eta = 0.3$. We observe that, at each time step, the cost associated with PO-HU remains the lowest, while NPO-NHU consistently shows the highest cost. This reflects that incorporating pose optimization performs better than ignoring it, both in terms of instantaneous and accumulated cost.

Table 4 represents the comparison of average tracking error (mean $\pm$ standard deviation) across different planning horizons for the proposed PO-HU method. This tracking error is averaged over 100 trials. We observe that a longer planning horizon may increase tracking error if we execute the entire control input sequence. For example, when $H = 8$, executing all eight control inputs may incur a higher tracking error. In our method, although we generate the eight control inputs for the future eight time steps, we only execute the control input for the first time step, and then re-plan again for the next planning horizon.

This strategy yields a lower tracking error, as shown in the case of a planning horizon of $H = 1$.

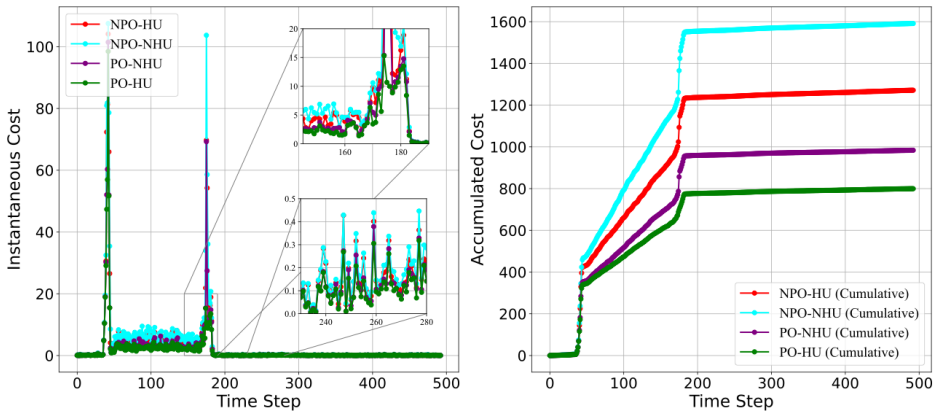


Figure 6. Instantaneous and accumulated cost for trajectory $\mathcal{A}_{\mathcal{N}}$ with $Q = 1500 \cdot I_6$, $R = I_9$, and $\eta = 0.3$.

Table 4. Average tracking error with standard deviation of trajectories for different planning horizons with PO-HU.

Trajectory	H = 1	H = 4	H = 6	H = 8
$\mathcal{A}_{\mathcal{P}}$	**0.049 ± 0.027**	0.072 ± 0.027	0.091 ± 0.031	0.104 ± 0.039
$\mathcal{A}_{\mathcal{N}}$	**0.034 ± 0.019**	0.056 ± 0.034	0.072 ± 0.038	0.082 ± 0.043
$\mathcal{B}_{\mathcal{P}}$	**0.041 ± 0.024**	0.062 ± 0.039	0.077 ± 0.045	0.085 ± 0.049
$\mathcal{B}_{\mathcal{N}}$	**0.031 ± 0.017**	0.057 ± 0.031	0.065 ± 0.035	0.075 ± 0.038

Note: Bold entries represent the lowest average tracking error for each trajectory across planning horizons.

Table 5 presents the average execution time with standard deviation in evaluating each candidate for different planning horizons. The time is averaged over 500 trials. In our approach, we can parallelize steps 4–9 of Algorithm 1. As a result, the computation times for methods with and without pose optimization differ slightly. This benefits from the multiple computing threads of the AMD 5975WX on our test computer. The majority of the computation time is spent on the one-time calculation of the Jacobian matrix. The minor increase is due to the increasing number of planning horizons and thus more iterations when solving the DARE (cf. Equation (8)).

Table 5. Average execution time with standard deviation in evaluating each candidate for different planning horizons with $|\Theta| = 20$ (in seconds).

Planning Horizon	PO-HU	NPO-HU	PO-NHU
4	0.013286 ± 0.00116	0.010474 ± 0.00094	0.013164 ± 0.00102
8	0.013633 ± 0.00121	0.011300 ± 0.00109	0.013295 ± 0.00107
12	0.013866 ± 0.00119	0.012736 ± 0.00113	0.013668 ± 0.00115
16	0.014126 ± 0.00124	0.012807 ± 0.00119	0.014003 ± 0.00119
20	0.014951 ± 0.00128	0.013287 ± 0.00116	0.014205 ± 0.00124
24	0.015287 ± 0.00131	0.013888 ± 0.00125	0.014257 ± 0.00129

5.3. Hardware Demonstration

Besides Gazebo simulations, we implemented our proposed algorithm in real hardware as a demo. A video of the hardware demo is available at https://www.youtube.com/

watch?v=QyR2NcSXhvY (accessed on 16 June 2025). For this hardware demo, we used an AprilTag marker on a board and placed a cup on top. We allowed the robot to capture the marker's pose with its built-in camera for end-effector tracking. We positioned the cup so that the marker remained visible to the camera. The captured pose was first obtained in the camera frame, and then it was transformed to the robot's base frame using the `tf` package. The human leads the robot by pulling the board along the x, y, and z axes. The robot predicts six future poses using LSTM, and uses a planning horizon of $H = 3$. It then plans three steps with MPC but executes only the first control input before re-planning to adapt to the human. We used $Q = 1100 \cdot I_6$, $R = I_9$, and $\kappa = w_\theta = 1$ in the demonstration. As discussed in Section 3, each control input consists of the linear and angular velocity of the base and seven joint angle velocities. The executions of these control inputs are driven by an ROS publisher and then sent to the corresponding ROS topic on Fetch. From the demonstration video, we observe that tracking is slower. This is primarily due to the physical execution time required for the robot to move its base and arm. The computation time does not significantly impact the slowness of the demonstration, as the generation of the candidate set is achieved by the pre-trained CVAE model, and the evaluation of each candidate was fully parallelized throughout the demonstration.

6. Conclusions and Future Works

We proposed and validated the Disturbance-Aware Redundant Control (DARC) framework, which integrates disturbance-aware MPC with a proactive pose optimization strategy for efficient human–robot co-transportation. The developed two-step iterative approach effectively addresses the inherent uncertainties of human behavior and internal robot disturbances by proactively generating and selecting robot joint configurations. To facilitate this process, we utilized a Conditional Variational Autoencoder (CVAE) to generate candidate sets of joint configurations for pose optimization, while a Long Short-Term Memory (LSTM) network was employed to predict future human poses. Theoretical derivations and simulated experiments with a Fetch mobile manipulator demonstrate the enhanced performance of our proposed framework compared to control methodologies that lack disturbance characterization or pose optimization. Future work will aim to refine the modeling of human behavior and decision-making processes and extend the proposed DARC framework to complex scenarios involving multi-robot multi-human collaborative tasks.

Author Contributions: Conceptualization, A.J.M., X.X. and X.W.; Methodology, A.J.M. and X.W.; Software, A.J.M., A.H.R., D.M.N. and X.W.; Validation, A.J.M., A.H.R., D.M.N. and X.W.; Formal analysis, A.J.M. and X.W.; Investigation, A.J.M., A.H.R. and X.W.; Resources, X.X. and X.W.; Data curation, A.J.M.; Writing—original draft, A.J.M. and X.W.; Writing—review & editing, A.J.M. and X.W.; Visualization, A.J.M.; Supervision, X.X. and X.W.; Funding acquisition, X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by U.S. National Science Foundation: ECCS Award 2332210.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Human Trajectory Prediction Using LSTM: As mentioned in Section 3.1, we need to predict human future poses during the co-transportation task. This is essential when we consider that the reference trajectory is unknown. For the prediction of human future movements based on previous poses, we use an LSTM-based model because of its ability to capture the temporal dependencies in sequential data.

Standard LSTM Equations: We denote the input to our model by $\{\boldsymbol{r}(k)\}_{k=1}^{K}$, which is a time-indexed sequence of observed human poses. Here, each $\boldsymbol{r}(k) \in \mathbb{R}^6$ represents the manipulator's end-effector pose to be tracked. The goal is to predict a sequence of future poses based on historic poses. This enables the manipulator to predict and adapt to human movements. At each discrete time step k, the LSTM maintains a hidden state $g_6(k) \in \mathbb{R}^{\varrho}$ and a cell state $g_5(k) \in \mathbb{R}^{\varrho}$, where ϱ is the dimensionality of the state vectors. The LSTM updates its states from time $k-1$ to k through a series of gated operations defined by the following standard equations:

$$g_1(k) = \sigma(W_{1,r}\,\boldsymbol{r}(k) + W_{1,6}\,g_6(k-1) + b_1), \tag{A1a}$$
$$g_2(k) = \sigma(W_{2,r}\,\boldsymbol{r}(k) + W_{2,6}\,g_6(k-1) + b_2), \tag{A1b}$$
$$g_3(k) = \sigma(W_{3,r}\,\boldsymbol{r}(k) + W_{3,6}\,g_6(k-1) + b_3), \tag{A1c}$$
$$g_4(k) = \tanh(W_{5,r}\,\boldsymbol{r}(k) + W_{5,6}\,g_6(k-1) + b_6), \tag{A1d}$$
$$g_5(k) = g_2(k) \odot g_5(k-1) \;+\; g_1(k) \odot g_4(k), \tag{A1e}$$
$$g_6(k) = g_3(k) \odot \tanh(g_5(k)). \tag{A1f}$$

where $\sigma(\cdot)$ is a sigmoid function constraining gate activations into $[0, 1]$, $\tanh(\cdot)$ introduces non-linearity by mapping inputs to $[-1, 1]$, and $\odot$ denotes element-wise multiplication. Input gate $g_1(k)$ regulates the contribution of the candidate cell state $g_4(k)$ to the current cell state $g_5(k)$, forget gate $g_2(k)$ controls the extent to which the previous cell state $g_5(k-1)$ is preserved, and output gate $g_3(k)$ determines the exposure of $g_5(k)$ to the hidden state $g_6(k)$. Trainable parameters include weight matrices $W_{(\cdot),r} \in \mathbb{R}^{\varrho \times 6}$, $W_{(\cdot),6} \in \mathbb{R}^{\varrho \times \varrho}$, and bias vectors $b_{(\cdot)} \in \mathbb{R}^{p}$, indexed by $(1, 2, 3, 5)$ representing (g_1, g_2, g_3, g_5), respectively.

Network Structure: Our network architecture employs a single-layer LSTM. We use $\varrho = 128$ hidden units to capture complex temporal dependencies in human movements. The LSTM transforms each input pose $\boldsymbol{r}(k) \in \mathbb{R}^6$ into hidden and cell states $(g_6(k), g_5(k))$:

$$(g_6(k), g_5(k)) = \mathrm{LSTM}(\boldsymbol{r}(k), g_6(k-1), g_5(k-1)). \tag{A2}$$

After processing the final input, the last hidden state is passed to a dense output layer. This output layer has $6 \times 6 = 36$ units, which are then reshaped into six predicted future poses:

$$[\tilde{\boldsymbol{r}}(k+1), \tilde{\boldsymbol{r}}(k+2), \ldots, \tilde{\boldsymbol{r}}(k+6)] \in \mathbb{R}^{6\times 6}. \tag{A3}$$

Here, each $\tilde{\boldsymbol{r}}(k+j) \in \mathbb{R}^6$ represents position and orientation at future time $k+j$. By generating multiple steps in a single forward pass, the model avoids iterative unrolling at inference time. This still leverages the LSTM's recurrent dynamics during training.

Input and Output Specification: The input to the model consists of a sequence of 10 consecutive past human poses, $\{\boldsymbol{r}(k-9), \boldsymbol{r}(k-8), \ldots, \boldsymbol{r}(k)\}$. We fed these poses chronologically into the LSTM layer. We designed the model in a way such that the network outputs six future poses, i.e., $\{\tilde{\boldsymbol{r}}(k+1), \tilde{\boldsymbol{r}}(k+2), \ldots, \tilde{\boldsymbol{r}}(k+6)\}$.

Data Collection and Normalization: We take the training data from different synthetic human movement trajectories through simulation. We use this trajectory data as a ground truth. Each trajectory is segmented into overlapping windows, where each sample consists of an input sequence $\{\boldsymbol{r}(k-9), \ldots, \boldsymbol{r}(k)\}$ and a corresponding label sequence $\{\boldsymbol{r}(k+1), \ldots, \boldsymbol{r}(k+6)\}$. We normalize each dimension of $\boldsymbol{r}(t)$ (i.e., $r_x, r_y, r_z, r_\alpha, r_\beta, r_\gamma$) using a min–max scaling to the range [0, 1]. This is because it ensures numerical stability, accelerates convergence, and mitigates scale differences between the position and orientation components.

Loss Function and Optimization: We define the loss function as the MSE over the 6-step prediction horizon, and it is averaged across all training samples:

$$\mathcal{L} = \frac{1}{6N} \sum_{i=1}^{N} \sum_{j=1}^{6} \|\boldsymbol{r}_i(k+j) - \tilde{\boldsymbol{r}}_i(k+j)\|_2^2, \tag{A4}$$

where N denotes total number of training samples, $\boldsymbol{r}_i(k+j)$ is the ground truth pose at time $k+j$ for sample i, and $\tilde{\boldsymbol{r}}_i(k+j)$ is its corresponding prediction. The model learns to align its predicted trajectories closely with the ground truth by penalizing the deviations in both position and orientation. All network parameters, weight matrices, and biases of the LSTM and output layers are optimized with the Adam optimizer using a learning rate of 10^{-3}.

References

1. Prewett, M.S.; Johnson, R.C.; Saboe, K.N.; Elliott, L.R.; Coovert, M.D. Managing workload in human–robot interaction: A review of empirical studies. *Comput. Hum. Behav.* **2010**, *26*, 840–856. [CrossRef]
2. Liu, L.; Schoen, A.J.; Henrichs, C.; Li, J.; Mutlu, B.; Zhang, Y.; Radwin, R.G. Human robot collaboration for enhancing work activities. *Hum. Factors* **2024**, *66*, 158–179. [CrossRef] [PubMed]
3. Bischoff, R.; Kurth, J.; Schreiber, G.; Koeppe, R.; Albu-Schäffer, A.; Beyer, A.; Eiberger, O.; Haddadin, S.; Stemmer, A.; Grunwald, G.; et al. The KUKA-DLR Lightweight Robot arm—A new reference platform for robotics research and manufacturing. In Proceedings of the ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), Munich, Germany, 7–9 June 2010; VDE: Berlin, Germany; pp. 1–8.
4. Pedrocchi, N.; Vicentini, F.; Matteo, M.; Tosatti, L.M. Safe human-robot cooperation in an industrial environment. *Int. J. Adv. Robot. Syst.* **2013**, *10*, 27. [CrossRef]
5. Broekens, J.; Heerink, M.; Rosendal, H.; et al. Assistive social robots in elderly care: A review. *Gerontechnology* **2009**, *8*, 94–103. [CrossRef]
6. Cai, S.; Ram, A.; Gou, Z.; Shaikh, M.A.W.; Chen, Y.A.; Wan, Y.; Hara, K.; Zhao, S.; Hsu, D. Navigating real-world challenges: A quadruped robot guiding system for visually impaired people in diverse environments. In Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems, Honolulu, HI, USA, 11–16 May 2024; pp. 1–18.
7. Ajoudani, A.; Zanchettin, A.M.; Ivaldi, S.; Albu-Schäffer, A.; Kosuge, K.; Khatib, O. Progress and prospects of the human–robot collaboration. *Auton. Robot.* **2018**, *42*, 957–975. [CrossRef]
8. Sharkawy, A.N.; Koustoumpardis, P.N. Human–robot interaction: A review and analysis on variable admittance control, safety, and perspectives. *Machines* **2022**, *10*, 591. [CrossRef]
9. Hayes, B.; Scassellati, B. Challenges in shared-environment human-robot collaboration. *Learning* **2013**, *8*.
10. Brosque, C.; Galbally, E.; Khatib, O.; Fischer, M. Human-robot collaboration in construction: Opportunities and challenges. In Proceedings of the 2020 IEEE International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 26–28 June 2020; pp. 1–8.
11. Cursi, F.; Modugno, V.; Lanari, L.; Oriolo, G.; Kormushev, P. Bayesian neural network modeling and hierarchical MPC for a tendon-driven surgical robot with uncertainty minimization. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2642–2649. [CrossRef]
12. Mahmud, A.J.; Li, W.; Wang, X. Mutual Adaptation in Human-Robot Co-Transportation with Human Preference Uncertainty. *arXiv* **2025**, arxiv:2503.08895.
13. Haninger, K.; Hegeler, C.; Peternel, L. Model predictive control with gaussian processes for flexible multi-modal physical human robot interaction. In Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 6948–6955.
14. Donelan, P. Singularities of robot manipulators. In *Singularity Theory: Dedicated to Jean-Paul Brasselet on His 60th Birthday*; World Scientific: Singapore, 2007; pp. 189–217.
15. Chiaverini, S. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Trans. Robot. Autom.* **1997**, *13*, 398–410. [CrossRef]
16. Chiacchio, P.; Chiaverini, S.; Sciavicco, L.; Siciliano, B. Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. *Int. J. Robot. Res.* **1991**, *10*, 410–425. [CrossRef]
17. Wittmann, J.; Kist, A.; Rixen, D.J. Real-Time Predictive Kinematics Control of Redundancy: A Benchmark of Optimal Control Approaches. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 11759–11766.

18. Faroni, M.; Beschi, M.; Tosatti, L.M.; Visioli, A. A predictive approach to redundancy resolution for robot manipulators. *IFAC-Pap. Online* **2017**, *50*, 8975–8980. [CrossRef]
19. Erhart, S.; Sieber, D.; Hirche, S. An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 315–322.
20. Mahmud, A.J.; Raj, A.H.; Nguyen, D.M.; Xiao, X.; Wang, X. Human-Robot Co-Transportation with Human Uncertainty-Aware MPC and Pose Optimization. *arXiv* **2024**, arxiv:2404.00514.
21. Aaltonen, I.; Salmi, T.; Marstio, I. Refining levels of collaboration to support the design and evaluation of human-robot interaction in the manufacturing industry. *Procedia CIRP* **2018**, *72*, 93–98. [CrossRef]
22. Magrini, E.; Ferraguti, F.; Ronga, A.J.; Pini, F.; De Luca, A.; Leali, F. Human-robot coexistence and interaction in open industrial cells. *Robot. Comput.-Integr. Manuf.* **2020**, *61*, 101846. [CrossRef]
23. Roveda, L.; Maskani, J.; Franceschi, P.; Abdi, A.; Braghin, F.; Molinari Tosatti, L.; Pedrocchi, N. Model-based reinforcement learning variable impedance control for human-robot collaboration. *J. Intell. Robot. Syst.* **2020**, *100*, 417–433. [CrossRef]
24. Peternel, L.; Tsagarakis, N.; Caldwell, D.; Ajoudani, A. Robot adaptation to human physical fatigue in human–robot co-manipulation. *Auton. Robot.* **2018**, *42*, 1011–1021. [CrossRef]
25. Mujica, M.; Crespo, M.; Benoussaad, M.; Junco, S.; Fourquet, J.Y. Robust variable admittance control for human–robot co-manipulation of objects with unknown load. *Robot. Comput.-Integr. Manuf.* **2023**, *79*, 102408. [CrossRef]
26. Zhang, Y.; Pezzato, C.; Trevisan, E.; Salmi, C.; Corbato, C.H.; Alonso-Mora, J. Multi-Modal MPPI and Active Inference for Reactive Task and Motion Planning. *IEEE Robot. Autom. Lett.* **2024**, *9*, 7461–7468. [CrossRef]
27. Bhardwaj, M.; Sundaralingam, B.; Mousavian, A.; Ratliff, N.D.; Fox, D.; Ramos, F.; Boots, B. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In Proceedings of the Conference on Robot Learning, Auckland, New Zealand, 14–18 December 2022; pp. 750–759.
28. De Schepper, D.; Schouterden, G.; Kellens, K.; Demeester, E. Human-robot mobile co-manipulation of flexible objects by fusing wrench and skeleton tracking data. *Int. J. Comput. Integr. Manuf.* **2023**, *36*, 30–50. [CrossRef]
29. Sirintuna, D.; Giammarino, A.; Ajoudani, A. An Object Deformation-Agnostic Framework for Human–Robot Collaborative Transportation. *IEEE Trans. Autom. Sci. Eng.* **2023**, *21*, 1986–1999. [CrossRef]
30. Sirintuna, D.; Ozdamar, I.; Gandarias, J.M.; Ajoudani, A. Enhancing Human-Robot Collaboration Transportation through Obstacle-Aware Vibrotactile Feedback. *arXiv* **2023**, arxiv:2302.02881.
31. Bussy, A.; Gergondet, P.; Kheddar, A.; Keith, F.; Crosnier, A. Proactive behavior of a humanoid robot in a haptic transportation task with a human partner. In Proceedings of the 2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication, Paris, France, 9–13 September 2012; pp. 962–967.
32. Agravante, D.J.; Cherubini, A.; Sherikov, A.; Wieber, P.B.; Kheddar, A. Human-humanoid collaborative carrying. *IEEE Trans. Robot.* **2019**, *35*, 833–846. [CrossRef]
33. Berger, E.; Vogt, D.; Haji-Ghassemi, N.; Jung, B.; Amor, H.B. Inferring guidance information in cooperative human-robot tasks. In Proceedings of the 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Atlanta, GA, USA, 15–17 October 2013; pp. 124–129.
34. Zanon, M.; Gros, S. Safe reinforcement learning using robust MPC. *IEEE Trans. Autom. Control* **2020**, *66*, 3638–3652. [CrossRef]
35. Saltık, M.B.; Özkan, L.; Ludlage, J.H.; Weiland, S.; Van den Hof, P.M. An outlook on robust model predictive control algorithms: Reflections on performance and computational aspects. *J. Process Control* **2018**, *61*, 77–102. [CrossRef]
36. Mohammadpour, J.; Scherer, C.W. *Control of Linear Parameter Varying Systems with Applications*; Springer Science & Business Media: Berlin, Germany, 2012.
37. Patel, S.; Sobh, T. Manipulator performance measures-a comprehensive literature survey. *J. Intell. Robot. Syst.* **2015**, *77*, 547–570. [CrossRef]
38. Yoshikawa, T. Manipulability of robotic mechanisms. *Int. J. Robot. Res.* **1985**, *4*, 3–9. [CrossRef]
39. Cheong, H.; Ebrahimi, M.; Duggan, T. Optimal design of continuum robots with reachability constraints. *IEEE Robot. Autom. Lett.* **2021**, *6*, 3902–3909. [CrossRef]
40. Mohammed, A.; Schmidt, B.; Wang, L.; Gao, L. Minimizing energy consumption for robot arm movement. *Procedia Cirp* **2014**, *25*, 400–405. [CrossRef]
41. Zhang, M.M.; Atanasov, N.; Daniilidis, K. Active end-effector pose selection for tactile object recognition through monte carlo tree search. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 3258–3265.
42. Wang, J.; Zhang, T.; Wang, Y.; Luo, D. Optimizing Robot Arm Reaching Ability with Different Joints Functionality. In Proceedings of the 2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Busan, Republic of Korea, 28–31 August 2023; pp. 1778–1785.

43. Schneider, U.; Posada, J.R.D.; Verl, A. Automatic pose optimization for robotic processes. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 2054–2059.
44. George Thuruthel, T.; Ansari, Y.; Falotico, E.; Laschi, C. Control strategies for soft robotic manipulators: A survey. *Soft Robot.* **2018**, *5*, 149–163. [CrossRef]
45. Kumar, V.; Hoeller, D.; Sundaralingam, B.; Tremblay, J.; Birchfield, S. Joint space control via deep reinforcement learning. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 3619–3626.
46. Hua, X.; Wang, G.; Xu, J.; Chen, K. Reinforcement learning-based collision-free path planner for redundant robot in narrow duct. *J. Intell. Manuf.* **2021**, *32*, 471–482. [CrossRef]
47. Shen, Y.; Jia, Q.; Huang, Z.; Wang, R.; Fei, J.; Chen, G. Reinforcement learning-based reactive obstacle avoidance method for redundant manipulators. *Entropy* **2022**, *24*, 279. [CrossRef] [PubMed]
48. Soleimani Amiri, M.; Ramli, R. Intelligent trajectory tracking behavior of a multi-joint robotic arm via genetic–swarm optimization for the inverse kinematic solution. *Sensors* **2021**, *21*, 3171. [CrossRef] [PubMed]
49. Parque, V.; Miyashita, T. Smooth curve fitting of mobile robot trajectories using differential evolution. *IEEE Access* **2020**, *8*, 82855–82866. [CrossRef]
50. Sun, L.; Zhan, W.; Hu, Y.; Tomizuka, M. Interpretable modelling of driving behaviors in interactive driving scenarios based on cumulative prospect theory. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 4329–4335.
51. Zhang, J.; Liu, H.; Chang, Q.; Wang, L.; Gao, R.X. Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly. *CIRP Ann.* **2020**, *69*, 9–12. [CrossRef]
52. Kratzer, P.; Toussaint, M.; Mainprice, J. Prediction of human full-body movements with motion optimization and recurrent neural networks. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 1792–1798.
53. Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; Savarese, S. Social lstm: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 961–971.
54. Manh, H.; Alaghband, G. Scene-lstm: A model for human trajectory prediction. *arXiv* **2018**, arxiv:1808.04018.
55. Rossi, L.; Paolanti, M.; Pierdicca, R.; Frontoni, E. Human trajectory prediction and generation using LSTM models and GANs. *Pattern Recognit.* **2021**, *120*, 108136. [CrossRef]
56. Fetch Robotics. *Fetch & Freight Research Edition Documentation*. Available online: https://github.com/fetchrobotics/docs (accessed on 16 June 2025).
57. Campa, R.; De La Torre, H. Pose control of robot manipulators using different orientation representations: A comparative review. In Proceedings of the 2009 IEEE American Control Conference, St. Louis, MO, USA, 10–12 June 2009; pp. 2855–2860.
58. Zheng, H.; Negenborn, R.R.; Lodewijks, G. Trajectory tracking of autonomous vessels using model predictive control. *IFAC Proc. Vol.* **2014**, *47*, 8812–8818. [CrossRef]
59. Shi, Q.; Zhao, J.; El Kamel, A.; Lopez-Juarez, I. MPC based vehicular trajectory planning in structured environment. *IEEE Access* **2021**, *9*, 21998–22013. [CrossRef]
60. Ho, C.K.; Chan, L.W.; King, C.T.; Yen, T.Y. A deep learning approach to navigating the joint solution space of redundant inverse kinematics and its applications to numerical IK computations. *IEEE Access* **2023**, *11*, 2274–2290. [CrossRef]
61. Dozat, T. Incorporating Nesterov Momentum into Adam. In Proceedings of the International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016; pp. 1–4.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.