

RETRO: Reactive Trajectory Optimization for Real-Time Robot Motion Planning in Dynamic Environments

Apan Dastider¹, Hao Fang¹ and Mingjie Lin¹

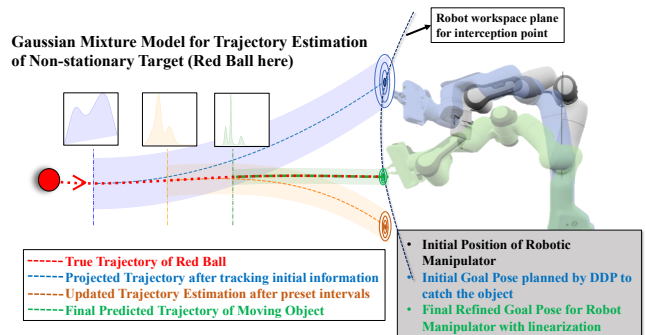
Abstract—Reactive trajectory optimization for robotics presents formidable challenges, demanding the rapid generation of purposeful robot motion in complex and swiftly changing dynamic environments. While much existing research predominantly addresses robotic motion planning with predefined objectives, emerging problems in robotic trajectory optimization frequently involve dynamically evolving objectives and stochastic motion dynamics. However, effectively addressing such reactive trajectory optimization challenges for robot manipulators proves difficult due to inefficient, high-dimensional trajectory representations and a lack of consideration for time optimization.

In response, we introduce a novel trajectory optimization framework called RETRO. RETRO employs adaptive optimization techniques that span both spatial and temporal dimensions. As a result, it achieves a remarkable computing complexity of $O(T^{2.4}) + O(Tn^2)$, a significant improvement over the traditional application of DDP, which leads to a complexity of $O(n^4)$ when reasonable time step sizes are used. To evaluate RETRO’s performance in terms of error, we conducted a comprehensive analysis of its regret bounds, comparing it to an Oracle value function obtained through an Oracle trajectory optimization algorithm. Our analytical findings demonstrate that RETRO’s total regret can be upper-bounded by a function of the chosen time step size. Moreover, our approach delivers smoothly optimized robot trajectories within the joint space, offering flexibility and adaptability for various tasks. It can seamlessly integrate task-specific requirements such as collision avoidance while maintaining real-time control rates. We validate the effectiveness of our framework through extensive simulations and real-world robot experiments in closed-loop manipulation scenarios.

For further details and supplementary materials, please visit: <https://sites.google.com/view/retro-optimal-control/home>

I. INTRODUCTION

Reactive Trajectory Optimization (RETRO) is a distinct paradigm in robotic control, characterized by dynamic changes in system dynamics and optimization/control objectives over time. In this approach, the full set of system state data is not available simultaneously. The progression of the control process is represented as a sequence of system control policies. The primary aim is to continuously and effectively refine the currently computed trajectory throughout this sequence while intelligently incorporating knowledge from past experiences. To be more specific, the overarching objective of RETRO is to minimize a predefined cost function over an entire epoch of robotic control. This remains true even when making multiple adjustments or refinements to the presently computed control policy. In essence, RETRO can be conceptualized as an online learning process that necessitates the fusion of knowledge from sequentially presented data streams over time. Practically, RETRO also seeks to optimize memory usage, computational resources, and



7 DoF Franka Emika Panda Arm Fixed on a Tabletop

Fig. 1. Overall Experimental Platform for RETRO. A Robotic Research-grade 7 DoF Robotic Manipulator has been used and a ball is thrown towards the manipulator to intercept it. As more information for ball tracking is available, the GMM model becomes more accurate to predict the future trajectory. Variance gets smaller as more data is available for trajectory prediction. With updated trajectory information, the action sequence also gets refined through computation-efficient action adjustment method.

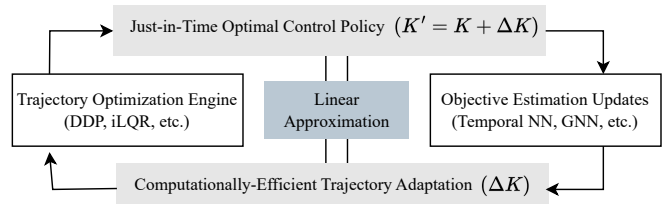


Fig. 2. Conceptual Diagram of RETRO methodology. execution speed during the robotic trajectory optimization process, making it a fast and efficient approach to robotic control in dynamic and evolving scenarios.

Practical applications of Reactive Trajectory Optimization (RETRO) are diverse and encompass a wide range of scenarios. For instance, consider a mobile robot tasked with reaching a goal pose that swiftly moves along an unpredictable yet discernible path. In another context, a navigating drone endeavors to safely explore a cluttered environment, equipped with sensors featuring severely limited detection ranges. Consequently, it can only compute its control policy incrementally. Taking a more imaginative perspective, envision a highly intelligent missile interceptor seeking to intercept a highly maneuverable hypersonic missile warhead. In this research paper, our primary focus is on the development of an efficient real-time control algorithm. This algorithm empowers a 7-degree-of-freedom (7-DOF) Franka robotic arm to consistently and successfully catch a flying subject that approaches it. Our approach distinguishes itself from several analogous problem settings [1]–[7] by a crucial assumption: the trajectory of the flying subject, while predictable, can only be estimated incrementally and with precision challenges. This limitation arises from the early

¹Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL, 32816, USA (E-mail: apan.dastider@ucf.edu)

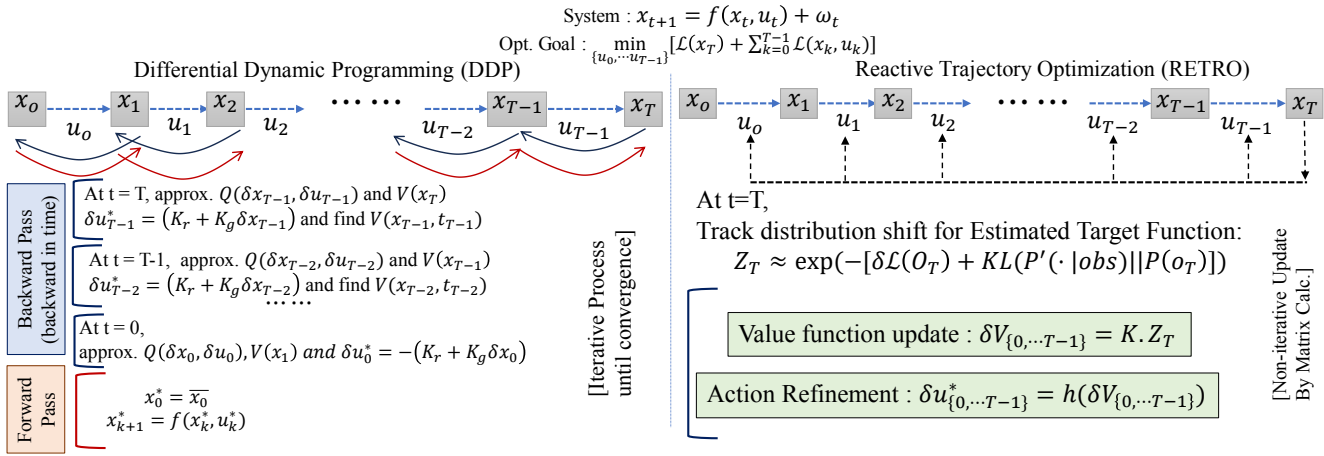


Fig. 3. A Comparative Analysis Between DDP-based Action Refinement and RETRO suggested Optimal Action Sequence Planning. The left panel shows that a multiple-run DDP is required for a non-stationary target. In the right panel, the proposed RETRO upgrades the actions based on the initially planned trajectory and value function update calculated through linearized calculation.

stages of robotic trajectory optimization, where the scarcity of sensory data renders the prediction of the flying subject's final impact point highly imprecise and inherently probabilistic in nature. Nonetheless, the trajectory optimization process must remain continuous throughout the entire operational epoch so that it can easily solve any non-stationary tracking problem.

II. RELATED WORK

Optimal control theory, which uses the dynamical system approach, has been widely used for solving modern trajectory and tracking problems [1], [2], [8]–[18]. In robotics, Differential Dynamic Programming (DDP) is well-studied for motion planning by solving nonlinear and time-varying optimization problems. Prior works [1], [2], [8], [9], [19] studied how to effectively apply DDP to control systems subject to nonlinear constraints, such as safety constraints [1] and feasibility constraints [2], [8], parametric uncertainty [10], as well as how to account for uncertainties and stochasticity in the system dynamics [9]. More recently, multiple studies attempted to improve the efficiency and applicability of DDP-based control methods. For example, study [11] explored the integration of data-driven approaches and Gaussian processes into DDP in order to enhance the adaptability and generalization of DDP-based controllers. Study [12] presented a general parameterized version of differential dynamic programming for solving problems with time-invariant parameters. Study [13] ensured better numerical stability while handling dynamics and path constraints through primal-dual proximal Lagrangian Approach and advanced integrators.

III. OPTIMAL CONTROL PROBLEM IN RETRO

We consider the following general nonlinear discrete time control dynamical system

$$x_{t+1} = f(x_t, u_t) + \omega_t, \quad (1)$$

where $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ are the system state and the control input at time-step t , correspondingly. The nonlinear system transition function $f(\cdot, \cdot) : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is smooth and assumed to have second order derivatives. ω_t models independent and identically distributed Gaussian noise.

Conventionally, the finite time optimal control problem

giving (1) is to find a control sequence $U^* = \{u_0^*, \dots, u_T^*\}$ such that it minimizes the following cost function.

$$J(u_0, \dots, u_T) = L(x_T) + \sum_{t=0}^{T-1} L(x_t, u_t), \quad (2)$$

where $L(x_t, u_t)$ and $L(x_T)$ are the running cost and final cost, respectively. Given the principle of optimality [20], one can use a dynamic programming algorithm such as DDP to minimize Eq. (2) by solving a single time-step control action backward in time [21],

$$V(x_t, t) = \min_{u_t} \{L(x_t, u_t) + \underbrace{V(x_{t+1}, t+1)}_{Q(x_t, u_t)}\}. \quad (3)$$

Here, we consider a quadratic Taylor expansion of a sufficiently small perturbation of $Q(x_t + \delta x_t, u_t + \delta u_t)$ around a nominal point $[x_t, u_t]$,

$$\begin{aligned} Q(x_t + \delta x_t, u_t + \delta u_t) &\approx Q(x_t, u_t) + (Q_t^x)' \delta x_t + (Q_t^u)' \delta u_t \\ &+ \frac{1}{2} \begin{pmatrix} (\delta x_t)' \\ (\delta u_t)' \end{pmatrix} \begin{pmatrix} Q_t^{xx} & Q_t^{xu} \\ Q_t^{ux} & Q_t^{xx} \end{pmatrix} \begin{pmatrix} \delta x_t \\ \delta u_t \end{pmatrix} \end{aligned} \quad (4)$$

where the superscripts of Q are the derivatives that can be solved analytically [21] and $'$ denotes transpose of matrix. Using the above approximation Eq. (4), one can minimize $\delta Q_t = Q(x_t + \delta x_t, u_t + \delta u_t) - Q(x_t, u_t)$ with respect to δu_t and get the conventional DDP optimal control update backward in time,

$$\delta u_t^{\text{DDP}} = K_r + K_g \delta x_t, \quad (5)$$

where the feedforward gain $K_r = -(Q_t^{uu})^{-1} Q_t^{xu}$ and feedback gain $K_g = -(Q_t^{uu})^{-1} Q_t^{ux}$. Next, a line search-based forward process will be computed to update the nominal trajectory,

$$\begin{aligned} \hat{x}_0 &= x_0, \\ \hat{u}_t &= u_t + \varepsilon K_r + K_g (\hat{x}_t - x_t), \\ \hat{x}_{t+1} &= f(\hat{x}_t, \hat{u}_t), \end{aligned} \quad (6)$$

where $0 < \varepsilon < 1$ is a scalar. The above process repeats until the value function converges (see Fig. 3 left panel).

On the contrary, optimal control handling in Reactive Trajectory Optimization (RETRO) is versatile and dynamic since the target state x_T is non-stationary and probabilistic. Thus, the corresponding control sequence needs to be adjusted against the distribution shift according to the objective target

state x_T . Here, we defined the time-varying objective goal location at each discrete time step t as o_t , which can be modeled by the time-series forecasting techniques such as the Gaussian mixture model (GMM) [22] or Bayesian time series analysis [23]. Therefore, the prior value function in Eq. (3) becomes the following equation in the objective belief space, $V(o_t, t) = \min_{u_t} \{L(o_t, u_t) + V(o_{t+1}, t+1)\}$. As new observations appear from tracking non-stationary target locations, the objective belief space at each time step evolves which in turn updates the prior belief about the objective trajectory. Thus, we introduce the following Kullback-Leibler(KL) divergence to quantify the distribution shift between posterior belief $P'(o_t|obs)$ and prior belief $P(o_t)$,

$$D_{KL}(P'(o_t|obs)||P(o_t)) = E_{o_t \sim P'(\cdot)} \log \left[\frac{P'(o_t|obs)}{P(o_t)} \right]. \quad (7)$$

With (7), we modify the prior value function by adding the KL divergence and reformulate the optimal control problem in RETRO which indirectly helps to address non-stationary control problems in real-time,

$$\tilde{V}(o_t, t) = \min_{u_t} \{E_{o_t \sim P'(\cdot)} [\tilde{L}(o_t, u_t) + \tilde{V}(o_{t+1}, t+1)] + D_{KL}(P'(o_t|obs)||P(o_t))\}. \quad (8)$$

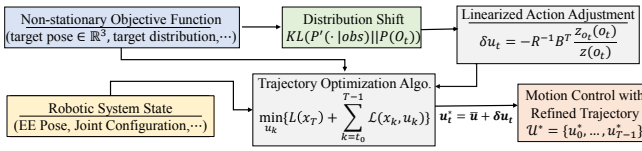


Fig. 4. Overall Algorithmic Blocks of RETRO

IV. METHODOLOGY OF REACTIVE TRAJECTORY OPTIMIZATION (RETRO)

Traditional DDP algorithms effectively solve the optimal control problem under the scenario where the tracking target is stationary and deterministic. However, when the evolution of the target trajectory is non-stationary and uncertain, it becomes challenging to run the DDP algorithm to achieve optimal control sequence as well as realize expected tracking performance. One naive solution to mitigate uncertainties is to repeat performing the DDP algorithm at certain time intervals as we have more information about future trajectories. However, running the DDP algorithm multiple times is very computation-heavy and there remains the risk of solving an online dynamic robotic task because of multiple forward-backward processes in control calculation [24].

To address the above challenges, we develop an efficient control adjustment algorithm while maintaining the utilization of the DDP solutions. Specifically, when we receive a belief-space trajectory $\{o_0, \dots, o_T\}$ for varying target objectives, the KL divergence enters into the value function Eq. (8) resulting in the prior DDP optimal solution becoming sub-optimal. Therefore, we perform control adjustments by exploiting linearly solvable optimal control formulation to fine-tune previously computed sub-optimal control sequences $\{\tilde{u}_0, \dots, \tilde{u}_T\}$. Intuitively, our method utilizes the introduced KL divergence term to guide the refinement of control actions. Fig. 4 depicts the overall methodology. We provide the detailed derivations of our algorithm in the following subsection. Although the derivations slightly follow the KL

control theory [25], [26], the control adjustment scenario is innovative in our work.

A. Linearly Solvable Value Function Update

We start by calculating the difference of the minimized value functions,

$$\begin{aligned} \delta V(o_t, t) &= \tilde{V}(o_t, t) - V(o_t, t) \\ &= \delta L(t) + \delta V(o_{t+1}, t+1) + D_{KL}(P'(o_t|obs)||P(o_t)) \\ &= \delta L(t) + E_{o_t \sim P'(\cdot)} \log \left[\frac{P'(o_t|obs)}{P(o_t)} e^{-\delta V(o_{t+1}, t+1)} \right], \end{aligned} \quad (9)$$

where $\delta V(o_{t+1}, t+1) = E_{o_t \sim P'(\cdot)} [\tilde{V}(o_{t+1}, t+1)] - V(o_{t+1}, t+1)$ and $\delta L(t) = E_{o_t \sim P'(\cdot)} [\tilde{L}(o_t, u_t)] - L(o_t, u_t)$. Next, we define the desirability function,

$$z(o_{t+1}) = e^{-\delta V(o_{t+1}, t+1)}. \quad (10)$$

Intuitively, $z(o_t)$ contains information about how much control adjustment is required at time-step t . We next define a normalization term for the KL divergence as,

$$g[z](o_t) = \sum_t P(o_t) z(o_{t+1}). \quad (11)$$

As a result, the Eq.(9) can be simplified as,

$$\begin{aligned} \delta V(o_t, t) &= \delta L(t) + E_{o_t \sim P'(\cdot)} \log \left[\frac{P'(o_t|obs)}{P(o_t) z(o_{t+1}) \frac{g[z](o_t)}{g[z](o_t)}} \right] \\ &= \delta L(t) + E_{o_t \sim P'(\cdot)} \log \left[\frac{1}{g[z](o_t)} \right] + D_{KL}(P'(o_t|obs)|| \frac{P(o_t) z(o_{t+1})}{g[z](o_t)}). \end{aligned} \quad (12)$$

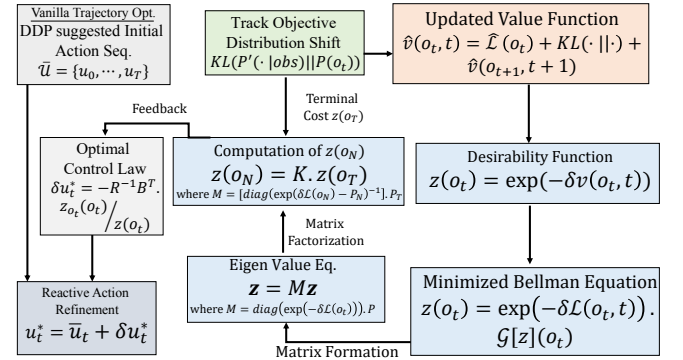


Fig. 5. RETRO diagram for control adjustment

To minimize the additional KL divergence introduced term $D_{KL}(P'(o_t|obs)|| \frac{P(o_t) z(o_{t+1})}{g[z](o_t)})$ in (12), one can simplify and restrict the following termination conditions,

$$z(o_{t+1}) = \frac{P'(o_t|obs)}{P(o_t)} g[z](o_t), \quad (13)$$

which results in the vanishing of KL divergence term since (13) ensures two identical distributions. Thus, we obtain the minimized value function update $\delta V(o_t, t)$ formulation as $\delta V(o_t, t) = \delta L(t) + \log \left[\frac{1}{g[z](o_t)} \right]$. Using the definition of the desirability function (10), the minimized equation (12) can further be simplified under some algebraic operations (see Appendix for details),

$$z(o_t) = e^{-\delta L(t)} g[z](o_t). \quad (14)$$

The main advantage of the above relationship (14) is that the exponentiated minimized value function $z(o_t)$ becomes linear in z as the normalization term $g[z](o_t)$ can be treated

as a linear operator. Essentially, it can be written in a more compact matrix form (see Appendix for details).

$$z = Mz, \quad (15)$$

where $M = \text{diag}(e^{-\delta L(1:T)})P(O_{1:T}) \in \mathbb{R}^{(T \times T)}$. Therefore, Eq. (15) is greatly simplified compared to (14) as in the vector form. To solve Eq. (15) efficiently, we define index sets N and T as a non-terminal trajectory index and terminal trajectory index, respectively. Thus, Eq. (15) can be partitioned as $\text{diag}(e^{\delta L(N)} - P(O_N))z(O_N) = P(O_T)z(O_T)$ via matrix partitioning, which gives the solution as

$$z(O_N) = (\text{diag}(e^{\delta L(N)} - P(O_N))^{-1}P(O_T)z(O_T)). \quad (16)$$

Once we calculate all z values, i.e., $z(O_{1:T})$ for the pre-computed trajectory from DDP, we can associate the change of minimized value function $\delta V(o_t, t)$ with the desirability function $z(o_t, t)$ using the chain rule,

$$\frac{\partial \delta V(o_t, t)}{\partial o_t} = \frac{\partial z(o_t, t)}{\partial o_t} \frac{1}{z(o_t)}. \quad (17)$$

In the next section, We provide an example of fine-tuning the control sequence according to the adjustment of the value function.

B. Adjustment of sub-optimal control sequence

Considering the following control dynamics of our robotic arms with quadratic cost $L(x_t, u_t, t) = \frac{1}{2}u_t^\top R_t u_t + x_t^\top Q_t x_t$, the closed-form optimal control law solved from the Hamilton-Jacobi-Bellman(HJB) equation is given by,

$$\begin{aligned} dx &= Axdt + Budt + \sigma d\omega, \\ u_t^{\text{optimal}} &= -R_t^{-1}B^\top \frac{\partial V(x,t)}{\partial x}. \end{aligned} \quad (18)$$

Together combining (17) and (18) we calculate each step control adjustment as

$$\delta u_t^* = -R_t^{-1}B^\top \frac{\partial \delta V(o_t, t)}{\partial o_t} = -R_t^{-1}B^\top \frac{\partial z(o_t, t)}{\partial o_t} \frac{1}{z(o_t)}. \quad (19)$$

As a result, we can finetune the previous sub-optimal control law as $u_t^* = \bar{u}_t + \delta u_t^*$. To this end, we complete our RETRO algorithm design. A summary of the proposed algorithm can be found in the algorithm 1 and figure 5.

Algorithm 1 RETRO Algorithm

Input:

Initial Control Seq. $\bar{U} = \{\bar{u}_0, \dots, \bar{u}_T\}_{\text{DDP}}$

Prior Belief Traj. $B = \{P(o_0), \dots, P(o_T)\}$

Shift Threshold, λ_{thres}

Output:

Optimal Control Sequence, U^*

Function *Online trajectory optimization*($\bar{U}, B, \lambda_{\text{thres}}$):

```

for  $t \leftarrow 1$  to  $T$  do
  GET :  $\{obs\}$  of target function
  UPDATE : Calc. Posterior  $P'(o_t|obs)$ 
  CALCULATE :  $D_{KL}(P'(o_t|obs)||P(o_t))$ 
  if  $D_{KL}(\cdot||\cdot) > \lambda_{\text{thres}}$  then
    CALCULATE :  $z(o_t), \delta u_t^*$ 
    FINETUNE :  $u_t^* \leftarrow \bar{u}_t + \delta u_t^*$ 
  EXECUTE :  $u_t^*$ 

```

C. Regret Bound Analysis

In this section, we conduct theoretical analyses to investigate the error performance of the proposed RETRO

algorithm. We provide a regret-bound analysis compared with an Oracle value function solved by an Oracle trajectory optimization algorithm that completely knows the future evolution of the target function. First, we provide an intuition and observation. From Eq. (12), we aim to cancel the additional KL divergence cost but end with another normalization term $g[z](o_t)$. It should be understood that this term will accumulate leading to $V > V^*$, where V^* denotes the optimal value function achieved by the Oracle trajectory optimization algorithm. On the other hand, $g[z](o_t)$ should be bounded because of the following assumption and observation.

Assumption 4.1: Let $P(o_t)$ be the prior distribution of the object and $P'(o_t|obs)$ be the updated posterior distribution based on new observations. We assume the distribution shift measured by $D_{KL}(P'(o_t|obs)||P(o_t))$ is smooth and the upper bound can be estimated by $\alpha(\frac{1}{T})$,

$$0 \leq D_{KL}(P'(o_t|obs)||P(o_t)) \leq \alpha(\frac{1}{T}). \quad (20)$$

where $\alpha(\cdot)$ is an increasing function. Essentially, the above assumption excludes the rapid change in the objective trajectory o_t (see Appendix for details).

Lemma 4.2: The normalization term $g[z]o(t)$ defined in Eq.(11) by is bounded, i.e.,

$$g[z](o_t) \leq Te^{-\delta V_m}, \quad (21)$$

where T is the total time-step in the horizon of the control action sequence and δV_m is the minimum value function deviated from the optimal value function V^* across the total time-step.

Proof: From the definition of

$$\begin{aligned} g[z](o_t) &= \sum_t P(o_t)z(o_{t+1}) \\ &\leq \sum_t z(o_{t+1}) [\text{since } (p(o_t)) \leq 1] \\ &= e^{-\delta V_1} + e^{-\delta V_2} + \dots + e^{-\delta V_T} [\text{From def. of } z] \\ &\leq Te^{-\delta V_m} [\text{From def. of } \delta V_m] \end{aligned} \quad (22)$$

where $m \in [1, T]$ such that $e^{-\delta V_m}$ achieves the maximum. ■ Next, we describe the most essential theorem in stating the upper error bound. The proof of this theorem will use the above assumption 4.1 and lemma 4.2.

Theorem 4.3: The regret function R_t defined as the expected deviation of the current value function \tilde{V}_t from the optimal value function V_t^* is upper bounded,

$$\forall t \in [1, T] \quad R_t \leq \alpha(\frac{1}{T}) + \log(T) \quad (23)$$

Proof: We consider the proof by starting at time-step m , suggested by the previous lemma 4.2. From the definition,

$$\begin{aligned} R_m &= ||E_{o_m \sim P'(\cdot)}[\tilde{V}_m - V_m^*]| = E_{o_m \sim P'(\cdot)}||\delta V_m|| \\ &= E_{o_m \sim P'(\cdot)}\log(z(o_m)) \\ &= E_{o_m \sim P'(\cdot)}\left[\log\left[\frac{P'(o_m|obs)}{P(o_m)}g[z](o_m)\right]\right] \\ &\leq D_{KL}(P'(o_m|obs)||P(o_m)) + E_{o_m \sim P'(\cdot)}\log(Te^{-\delta V_m}) \\ &\leq \alpha(\frac{1}{T}) + \log(T) - R_m \\ &\leq \frac{\alpha(\frac{1}{T}) + \log(T)}{2}. \end{aligned} \quad (24)$$

On the other hand, the lower bound of R_m is 0 if there is no distribution shift, i.e., $D_{KL} = 0$. Together, we have the following

$$0 \leq R_m \leq \frac{\alpha(\frac{1}{T}) + \log(T)}{2}. \quad (25)$$

Thus, for a general time-step $t \in [1, T]$, $R_t \leq \alpha(\frac{1}{T}) + \log(T) - R_m$, which gives us the final regret upper bound

$$\forall t \in [1, T] \quad R_t \leq \alpha(\frac{1}{T}) + \log(T). \quad (26)$$

The above analysis provides an essential trade-off of our proposed algorithm. The regret error bound consists of two counterparts. The first term measures how large the distribution shift computed from the KL divergence. The larger T we have, the smaller $\alpha(\frac{1}{T})$ is. On the contrary, the second part is related to the error bound across all time step T . The larger T we have, the larger $\log(T)$ has. Our empirical simulation evidence also indicates the above theoretical trade-off evidence (see Fig. 8). In practice, researchers can decide the horizon T depending on the problem.

D. Complexity Analysis

We next analyze the computational complexity. The key ingredient of the RETRO algorithm is solving a linear equation (15) only once to get the fine-tuned control adjustment of all time steps, which clearly distinguishes the multiple-run DDP algorithm. We provide the following comparison between the multiple-run DDP and the RETRO algorithm.

The operation complexity of a single backward-forward DDP algorithm is given by $T(2n^3 + \frac{7}{2}n^3m + 2nm^3 + \frac{1}{3}m^3)$ [21]. In most cases, the dimension of the state variable n is much higher than the dimension of the control variable m , i.e., $n \gg m$. Therefore, we can simplify the single backward-forward operation complexity as $O(Tn^3)$. Considering the quadratic convergence rate and termination bounds $O(\frac{1}{n^2T^2})$, we require at least $O(Tn)$ iterations before satisfying termination bounds (see Appendix for details). Together, the total iterations of the DDP algorithm can be estimated as $O(Tn \times Tn^3) = O(T^2n^4)$.

On the other hand, our proposed method directly computes adjustment without any iterations. We solve Eq. (15) by inverting $T \times T$ matrix, which can be estimated as $O(T^{2.4})$. After that, the fine-tuned control adjustment is computed by Eq. (16) and Eq. (19), where the total operation complexity is by $O(Tn^2)$. Together, the total running complexity is $O(T^{2.4}) + O(Tn^2)$. Compared with the DDP algorithm $O(T^2n^4)$, our methods slightly sacrifice $O(T^{0.4})$ for the first term because of the ambitious goal of finding fine-tuning control sequence without any iterations. However, our proposed method does not require computations $O(n^4)$ on state dimensions for the first term and improves the second term by $O(Tn^2)$ which together hugely improves the running time. The above comparison illustrates the superior complexity performance of our RETRO algorithm.

V. EXPERIMENTAL PLATFORM

We utilized the 7-DoFs Franka Emika Panda robotic manipulator as a simulation testbed to validate our algorithm. Each state sample $x_t \in \mathbb{R}^{13}$ consists of 13 numerical floating values, including 7 joint angles, end-effector tip position coordinates in \mathbb{R}^3 and the most likely target coordinates in \mathbb{R}^3 estimated from GMM model [22]. We initialized the manipulator from a default *home* position for all experiments, but the dynamic target trajectory was initiated from different

locations with varying velocities for validating the generalizability of the proposed method. To meet the requirements of derivative computations by using an automatic differentiation tool and smooth calculation of system dynamics, we have modeled the entire simulation platform in the open-source robot simulator software Drake [27].

For real-hardware demonstrations, we used the same 7 DoFs Franka Emika robot manipulator fixed on a table-top as shown in Fig. 6. Our algorithm for object tracking and pose estimation, as well as our adaptive trajectory optimization pipeline, ran on the QUAD GPU server equipped with an Intel Core-i9-9820X processor. We incorporated the Intel RealSense Depth Camera D435i for obstacle detection and localizing dynamic targets in \mathbb{R}^3 . We tracked the target trajectory information through depth sensing in real-hardware experimentation and transformed the most probable intercepting point into the robot's coordinate system to catch the dynamic object smoothly in real time.

VI. RESULTS AND DISCUSSIONS

Through our experiments and simulation results, we wanted to investigate the following three questions,

- Can RETRO produce an optimized control sequence in real-time for intercepting the dynamic object?
- Can RETRO perform computation-efficient control adjustment in dynamic environments compared to the multiple-run iterative trajectory optimization process?
- Can RETRO exhibit trade-off performance as analysed by our theoretical proof and how does the control cost performance change related to time-horizon T ?

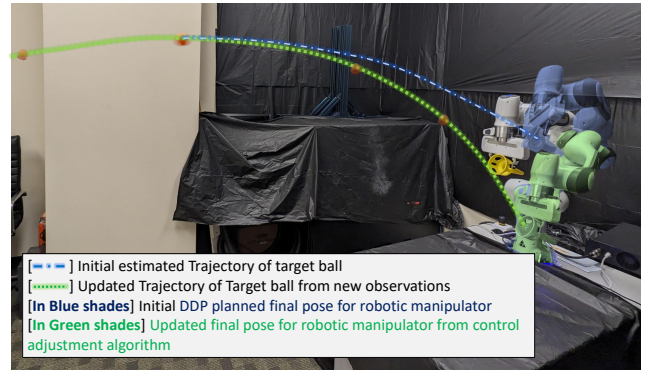


Fig. 6. Real Hardware demonstration for robotic Manipulator for target catching task.

A. Successful Task Completion

Our study aimed at leveraging the computation-efficient control adjustment technique for intercepting a flying target thrown toward the robotic manipulator. The object is thrown with different but bounded velocities from different initial locations. As discussed earlier, the object trajectory is tracked through a Realsense Depth sensor and the tracked trajectory points are given as time series inputs to our GMM model. The GMM model produces the probabilistic estimation for the most likely intercepting point. In Fig. 6, we have shown how in a real-hardware setting, the robotic manipulator followed the control sequence provided by RETRO. In the blue-shaded robot figure, we illustrate the sub-optimal final robot pose which is suggested by initial DDP-optimized control

inputs. Since the probabilistic estimation gets updated with new target locations, the obsolete control inputs need to be upgraded. The in-queue control sequence gets refined in the direction given by the feedback accruing from KL divergence cost at the terminal state. In green shades, we show how the optimal final pose looks for the robotic manipulator and the robotic manipulator intercepts the target object (a low-weight ball here) with smooth trajectory execution.

B. Efficient Control Computation

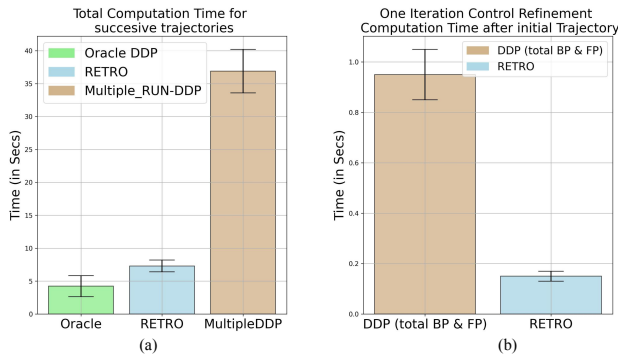


Fig. 7. (a) Total Computation Time for Final Control Trajectories against an evolving target location. Total runtime is the highest for Multiple-Run DDP because for each distribution shift the controller need to perform multiple forward pass and backward pass before convergence. (b) Computation time for one single control adjustment iteration after tracking target distribution changes. Computation time for new control sequences is higher for DDP iteration as it computes new control sequences from scratch. In RETRO, we only adjust the pre-computed initial control sequence.

Having established the successful task tracking performance, we next investigated how much computation efficiency the proposed method provides for control adjustment in a dynamic environment. To enable a fair comparison, we chose two baseline studies here: A) We chose an Oracle single-iteration DDP algorithm to which the full trajectory information and the accurate final landing point of the dynamic target are known beforehand. Simply, this Oracle DDP does not need to do any control input adjustment rather it gives an optimal control input sequence after meeting certain preset convergence criteria. B) We chose a computation-heavy multiple-run DDP optimization. The dynamic problem setting is as same as RETRO, but the method of re-optimization for handling non-stationary target locations is very different. For baselines, we completely discard the control sequence computed from a previous DDP run and start the optimization process from scratch for an updated target location. If the distribution shift is larger than a certain preset threshold, we run DDP again to find a new control sequence. In Fig. 7(a), we compare the computation time required for the mentioned baselines and the proposed method. As expected, the computation time is lowest for the Oracle method and highest for the multiple-run DDP method. Therefore, our method is computationally efficient with respect to the multiple-run DDP baseline since it performs very simple and efficient control input refinement for the already computed sub-optimal control sequence. On the other hand, the performance of our methods is close to the Oracle method which again confirms the superior performance. The above figure data are collected from Drake and ball movement

simulations in a smoothly evolving trajectory.

C. Performance Evaluation for Varying Horizon

Last, we investigated the effects of different preset horizons T for total regret values and control costs between the RETRO and the Oracle algorithms. Our theoretical analyses assume the smoothness of the distribution shift of the computed KL divergence and envision the trade-off regret performance of control horizons. The KL divergence increases as we have a smaller horizon. Thus, the RETRO adjusted the control input irregularly and the prior sub-optimal control inputs were executed by the robotic system for larger intervals leading to the value of regret increased and larger control error cost differences (see Fig. 8 left tail). On the contrary, as the horizon increases, the KL divergence becomes smoother leading to the difference in control cost shrinks. Therefore, the RETRO was only required to perform a very tiny control adjustment (see Fig. 8 right tail). Further, We also observed that as the horizon increases, the total value of regret increases as consistent with our theoretical proof where the term $\log T$ dominated the error upper bound. In between, We observed a relatively flat area where both the regret and control cost differences were small. In practice, how to select optimal times-step T that balances the above trade-off will be a future research direction.

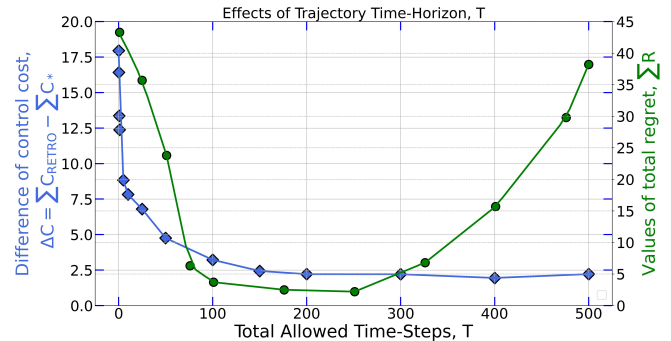


Fig. 8. Relationship of planning horizon T with the control cost difference and the total regret value. The control cost difference is in blue and the value of total regret is in green. Notice that the control cost difference becomes very tiny as T increases but does not converge to 0 as the linearized control refinement cost always exists.

VII. CONCLUSIONS

While much of the existing research has predominantly focused on optimal control with predefined objectives, the emerging problems in robotic trajectory optimization often involve objectives that evolve dynamically and motion dynamics that exhibit stochastic behavior. Unfortunately, effectively addressing such challenges for robot manipulators has proven difficult due to issues related to high-dimensional trajectory representations and a lack of consideration for time optimization. In response to these challenges, this paper has introduced a novel optimization framework, RETRO, to adaptively optimize trajectories across both spatial and temporal dimensions, resulting in smooth and optimally timed robot trajectories. Importantly, the RETRO framework is task-agnostic, allowing it to seamlessly incorporate additional task-specific requirements, such as collision avoidance, while maintaining real-time control rates.

REFERENCES

- [1] G. Alcan and V. Kyrki, "Differential dynamic programming with nonlinear safety constraints under system uncertainties," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1760–1767, 2022.
- [2] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 695–702.
- [3] D. Carneiro, F. Silva, and P. Georgieva, "Robot anticipation learning system for ball catching," *Robotics*, vol. 10, no. 4, 2021. [Online]. Available: <https://www.mdpi.com/2218-6581/10/4/113>
- [4] A. A. Oliva, E. Aertbeliën, J. De Schutter, P. R. Giordano, and F. Chaumette, "Towards dynamic visual servoing for interaction control and moving targets," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 150–156.
- [5] A. Dastider and M. Lin, "Non-parametric stochastic policy gradient with strategic retreat for non-stationary environment," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, 2022, pp. 1377–1384.
- [6] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1049–1065, 2014.
- [7] P. Zhao, Y.-F. Xie, L. Zhang, and Z.-H. Zhou, "Efficient methods for non-stationary online learning," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 11 573–11 585. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/4b70484ebef62484e0c8cdd269e482fd-Paper-Conference.pdf
- [8] C. Mastalli, W. Merkt, J. Marti-Saumell, H. Ferrolho, J. Solà, N. Mansard, and S. Vijayakumar, "A feasibility-driven approach to control-limited ddp," *Autonomous Robots*, vol. 46, no. 8, pp. 985–1005, Dec. 2022.
- [9] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014.
- [10] Y. Aoyama, A. D. Saravanos, and E. A. Theodorou, "Receding horizon differential dynamic programming under parametric uncertainty," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 3761–3767.
- [11] Y. Pan and E. A. Theodorou, "Data-driven differential dynamic programming using gaussian processes," in *2015 American Control Conference (ACC)*, 2015, pp. 4467–4472.
- [12] A. Oshin, M. D. Houghton, M. J. Acheson, I. M. Gregory, and E. A. Theodorou, "Parameterized differential dynamic programming," *arXiv preprint arXiv:2204.03727*, 2022.
- [13] W. Jallet, N. Mansard, and J. Carpentier, "Implicit differential dynamic programming," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1455–1461.
- [14] H. Fang, S. Berman, Y. Wang, and Y. Yang, "Robust adaptive deep brain stimulation control of non-stationary cortex-thalamus network models in parkinson's disease," *bioRxiv*, pp. 2023–08, 2023.
- [15] H. Fang and Y. Yang, "Predictive neuromodulation of cingulo-frontal neural dynamics in major depressive disorder using a brain-computer interface system: A simulation study," *Frontiers in Computational Neuroscience*, vol. 17, p. 1119685, 2023.
- [16] —, "A robust and adaptive control algorithm for closed-loop brain stimulation," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2021, pp. 6049–6052.
- [17] —, "Designing and validating a robust adaptive neuromodulation algorithm for closed-loop control of brain states," *Journal of Neural Engineering*, vol. 19, no. 3, p. 036018, 2022.
- [18] M. Vu, H. Fang, and S. Zeng, "Nonlinear optimal control synthesis using basis functions: Algorithms and examples," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2021, pp. 493–498.
- [19] Y. Huang, W. He, and S. Zeng, "A differential dynamic programming-based approach for balancing energy and time optimality in motion planning," in *2023 59th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2023, pp. 1–7.
- [20] B. Light, "The Principle of Optimality in Dynamic Programming: A Pedagogical Note," *arXiv e-prints*, p. arXiv:2302.08467, Feb. 2023.
- [21] L.-Z. Liao and C. A. Shoemaker, "Convergence in unconstrained discrete-time differential dynamic programming," *IEEE Transactions on Automatic Control*, vol. 36, no. 6, pp. 692–706, 1991.
- [22] E. Eirola and A. Lendasse, "Gaussian mixture models for time series modelling, forecasting, and interpolation," in *Proceedings of the 12th International Symposium on Advances in Intelligent Data Analysis XII - Volume 8207*, ser. IDA 2013. Berlin, Heidelberg: Springer-Verlag, 2013, p. 162–173. [Online]. Available: https://doi.org/10.1007/978-3-642-41398-8_15
- [23] M. F. Steel, "Bayesian time series analysis," in *Macroeconometrics and time series analysis*. Springer, 2010, pp. 35–45.
- [24] B. Plancher, "Parallel and constrained differential dynamic programming for model predictive control." Master's thesis, Harvard University, Cambridge, MA, USA, May. 2018.
- [25] E. Todorov, "Efficient computation of optimal actions," *Proceedings of the National Academy of Sciences*, vol. 106, no. 28, pp. 11 478–11 483, July 2009. [Online]. Available: <https://doi.org/10.1073/pnas.0710743106>
- [26] G. Neu and V. Gómez, "Fast rates for online learning in Linearly Solvable Markov Decision Processes," in *Proceedings of the 2017 Conference on Learning Theory*, ser. Proceedings of Machine Learning Research, S. Kale and O. Shamir, Eds., vol. 65. PMLR, 07–10 Jul 2017, pp. 1567–1588. [Online]. Available: <https://proceedings.mlr.press/v65/neu17a.html>
- [27] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019. [Online]. Available: <https://drake.mit.edu>