Stochastic Trajectory Optimization for Demonstration Imitation

Chenlin Ming[1], Zitong Wang[1], Boxuan Zhang[2], Xiaoming Duan[1] and Jianping He[1] .

Abstract—**Humans often learn new skills by imitating the experts and gradually developing their proficiency. In this work, we introduce Stochastic Trajectory Optimization for Demonstration Imitation (STODI), a trajectory optimization framework for robots to imitate the shape of demonstration trajectories with improved dynamic performance. Consistent with the human learning process, demonstration imitation serves as an initial step, while trajectory optimization aims to enhance robot motion performance. By generating random noise and constructing proper cost functions, the STODI effectively explores and exploits generated noisy trajectories while preserving the demonstration shape characteristics. We employ three metrics to measure the similarity of trajectories in both the time and frequency domains to help with demonstration imitation. Theoretical analysis reveals relationships among these metrics, emphasizing the benefits of frequency-domain analysis for specific tasks. Experiments on a 7-DOF robotic arm in the PyBullet simulator validate the efficacy of the STODI framework, showcasing the improved optimization performance and stability compared to previous methods. The source code can be found at ming-bot/Proud.**

Index Terms—**Motion and Path Planning, Optimization and Optimal Control, Learning from Demonstration.**

I. INTRODUCTION

IN the last decades, optimization-based planning methods have been widely used to optimize trajectory performance in high-dimensional configuration spaces. They gain great reputations for their explainability and ease of implementation [1]–[3]. Learning from demonstration (LfD) has also been extensively studied in recent years. Human teachers can easily transfer their knowledge to robots by providing high-quality demonstrations [4]. We focus on situations where robots are required to imitate the trajectories of rapid human movements. Given the high cost of the motion capture system, such demonstrations containing rapid movements are difficult to obtain. As to the human learning process, individuals frequently acquire new skills by imitating the experts and progressively developing their proficiency. Drawing inspiration from it, we divide the robot learning process into two main tasks: demonstration imitation and trajectory optimization. We relax the demand on the exact dynamic performance of the demonstration, merely requiring that the demonstration's shape encapsulates human prior knowledge. Demonstration imitation serves as an initial step for learning new skills, and trajectory optimization helps robots perform better. More specifically, we aim to get an optimized trajectory that inherits the demonstration's shape property but with better dynamic performance. While optimization-based planning methods mainly focus on searching for better-performing trajectories, rarely considering the resulting trajectory shape after optimization, LfD methods mainly rely on demonstrations, ignoring the optimization aspect. Nevertheless, when robots learn new skills, it is equally crucial to leverage prior human demonstrations fully and have the capability to optimize trajectory performance [5], [6].

In this paper, we introduce a framework named **S**tochastic **T**rajectory **O**ptimization for **D**emonstration **I**mitation (STODI). Adapted from STOMP [7], we categorize the iterative trajectories and reconstruct the iterative process to enhance the stability and exploration capability of our method. Compared with STOMP [7], our approach utilizes the same amount of random noise to explore and exploit more generated noisy trajectories. Since the stochastic trajectory optimization process does not require explicit gradient information, our method can effectively deal with more complex objectives, even in cases where gradients cannot be easily computed. To preserve the overall shape characteristic of the demonstration during optimization, we augment the cost function by introducing an additional term that captures the resemblance in the overall shape between the optimization trajectory and the demonstration. We further integrate time optimization into stochastic trajectory optimization, enabling users to modify the execution time of trajectories. Finally, we show the strong imitation and optimization capabilities of our framework through experiments using a 7-degree-of-freedom (DOF) robot in both simulated environments and real-world scenarios.

Our contributions mainly lie in that:

- We introduce a stochastic trajectory optimization framework named STODI, which categorizes the iterative trajectories and reconstructs the iterative process of STOMP [7], significantly improving both stability and optimization effectiveness.
- We employ three metrics—Dynamic Time Warping (DTW), Mean Squared Error (MSE) in the spectrum (MSES), and MSE in the power spectrum (MSEPS) to measure the similarity of trajectories in both time and frequency domains. Furthermore, we derive theoretical relationships among these metrics.
- We compare STODI with STOMP in the Pybullet simulator [8] to demonstrate the superiorities of our framework. We also provide specific denoising methods to enhance the demonstration's input quality. We implement STODI on a robotic arm in both simulated environments and real-

[1]The Department of Automation, Shanghai Jiao Tong University, and Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai, China. Email: {`mcl2019011457, wangzitong, xduan, jphe`}`@sjtu.edu.cn`.

[2] School of Computation, Information and Technology in Technical University of Munich (TUM). Email: `boxuan.zhang@tum.de`.

world scenarios, verifying its outstanding capability to perform imitation and optimization simultaneously.

Notation: We use slim symbols to denote scalars and bold symbols to denote vectors and matrices. Let $\boldsymbol{I}_N$ be an identity matrix with dimension $N \times N$. Let $\mathbf{0}_N$ and $\mathbf{1}_N$ represent all-zero and all-one column vectors with dimension N, respectively. The sets of real numbers and complex numbers are denoted by $\mathbb{R}$ and $\mathbb{C}$, respectively. Let $|\cdot|$ denote the modulus of a vector.

The remainder of this paper is organized as follows: Section II provides essential related work of trajectory optimization and imitation from demonstration. Section III presents the problem formulation and the details of the proposed STODI framework. Section IV shows the experiments and the results of the algorithm. Finally, Section V concludes the work.

II. Related Work

A. Optimization-based Methods

Optimization-based trajectory planners mainly address the planning problem by formulating and solving an optimization problem and have been developed for decades [7], [9]–[11]. CHOMP [9] presents a functional gradient approach to optimize motion planning by iteratively reducing the value of a cost function, which integrates both trajectory smoothness and collision avoidance into a single framework. TrajOpt [10] presents sequential convex optimization to solve motion planning problems, explicitly incorporating collision avoidance constraints into the optimization process. For optimization problems where computing the gradients of the cost function is challenging, STOMP [7] focuses on generating optimal robot trajectories through stochastic optimization, using noises to explore feasible trajectories effectively. Although we categorize STOMP as an optimization-based method, it leverages a sampling process and optimizes a trajectory in a stochastic manner.

B. Sampling-based Methods

Sampling-based trajectory planners rely on sampling configurations in order to quickly find feasible and optimal robot motions [12]–[16]. The Probabilistic Roadmap Method (PRM) [12] constructs a roadmap in the configuration space of robots by randomly sampling points and connecting them via feasible paths. This roadmap then serves as a guide to navigating between two points. The Rapidly-exploring Random Tree (RRT) [13] algorithm incrementally explores high-dimensional spaces by randomly sampling points and extending branches for efficient path planning. Jaillet [14] presented a PRM-based motion planning method for dynamically changing environments, addressing challenges such as real-time adaptation and path replanning. Sampling-based methods are frequently employed in the path planning of mobile robots due to their ability to deal with complex situations. However, these methods are computationally inefficient for motion planning of robotic manipulators.

C. Imitation from Demonstration

Imitation learning methods leverage trajectory models obtained from demonstrations to enhance performance. Behavioral Cloning from Observation (BCO) [17] allows autonomous agents to learn from state-only demonstrations with minimal environment interactions, achieving comparable performance to methods that require action information and more interactions. Inverse Reinforcement Learning (IRL) [18] methods aim to infer the reward function from observed expert behavior, enabling the model to generalize and perform well in unseen environments by optimizing the learned reward function. Ye [19] proposed a framework that combines learning from demonstrations and sampling-based motion planning. Similarly, Koert [20] and Osa [21] proposed methods combining optimization-based approaches and imitation learning-based approaches. However, these methods still have significant limitations in trajectory generalization and acquiring training datasets.

III. Methodology

This section mainly introduces the overall theoretical framework and the implementation details of the STODI algorithm. Firstly, we present the stochastic trajectory optimization problem. Then, we introduce the detailed process of the STODI algorithm and provide explanations for each module. To equip the STODI algorithm with the ability to imitate demonstrations, we introduce three metrics for measuring the overall similarity between trajectories. Specifically, we apply DTW to calculate the difference between two non-stationary trajectories in the time domain. Then, we introduce two frequency-domain metrics to complement the time-domain metric and theoretically derive their relationships with DTW.

A. Problem Formulation

We define all trajectories as discrete sequences. The trajectory can be represented as $\boldsymbol{\theta} \in \mathbb{R}^{N \times M}$, where N denotes the length of the discrete sequence and M represents the dimension of the trajectory's points. We assume that a robot has a fixed maximum control frequency f which determines the time interval of the discrete trajectory $\boldsymbol{\theta}$. Under this assumption, the discrete sequence's length N can reveal the motion duration by $T = (N-1)/f$. We then define $\tilde{\boldsymbol{\theta}}$ as a noisy trajectory following a normal distribution $\tilde{\boldsymbol{\theta}}_i \sim \mathcal{N}(\boldsymbol{\theta}_i, \boldsymbol{\Sigma}), i = 1, 2, \ldots, M$ with mean $\boldsymbol{\theta}_i$ and variance $\boldsymbol{\Sigma}$. The cost function of the noisy trajectory $\tilde{\boldsymbol{\theta}}$ is defined as:

$$Q(\tilde{\boldsymbol{\theta}}) = \sum_{i=1}^{N} q(\tilde{\boldsymbol{\theta}}_i^\top) + \frac{1}{2}(\tilde{\boldsymbol{\theta}}\mathbf{1}_M)^\top \boldsymbol{R}(\tilde{\boldsymbol{\theta}}\mathbf{1}_M), \tag{1}$$

where $\boldsymbol{R}$ is a positive semi-definite symmetric matrix. The $\sum_{i=1}^{N} q(\tilde{\boldsymbol{\theta}}_i^\top)$ term is a combination of arbitrary state-dependent cost terms which can include obstacle costs, expected velocity costs, and other specific costs. The $\frac{1}{2}(\tilde{\boldsymbol{\theta}}\mathbf{1}_M)^\top \boldsymbol{R}(\tilde{\boldsymbol{\theta}}\mathbf{1}_M)$ term presents as a control cost.

Following (1), the optimization problem is formulated as:

$$\min_{\boldsymbol{\theta}} \mathbb{E}\left[\sum_{i=1}^{N} q(\tilde{\boldsymbol{\theta}}_i^\top) + \frac{1}{2}(\tilde{\boldsymbol{\theta}}\mathbf{1}_M)^\top \boldsymbol{R}(\tilde{\boldsymbol{\theta}}\mathbf{1}_M)\right]. \tag{2}$$

Computing the gradient of the expectation in (2) with respect to $\boldsymbol{\theta}$, we have:

$$\begin{aligned}&\nabla_{\boldsymbol{\theta}}\left(\mathbb{E}\left[\sum_{i=1}^{N} q(\tilde{\boldsymbol{\theta}}_i^\top) + \frac{1}{2}(\tilde{\boldsymbol{\theta}}\mathbf{1}_M)^\top \boldsymbol{R}(\tilde{\boldsymbol{\theta}}\mathbf{1}_M)\right]\right) = 0 \\ &\Rightarrow \quad \mathbb{E}[\tilde{\boldsymbol{\theta}}] = -\boldsymbol{R}^{-1}\mathbb{E}\left[\nabla_{\boldsymbol{\theta}}\left(\sum_{i=1}^{N} q(\tilde{\boldsymbol{\theta}}_i^\top)\right)\right].\end{aligned} \tag{3}$$

Note that the complex composition of the cost function may render challenges in calculating gradients. Consistent with [7], [22], the gradient is estimated by aggregating benefits from the introduced noise:

$$\begin{aligned}\mathbb{E}[\tilde{\boldsymbol{\theta}}] &= -\boldsymbol{R}^{-1}\left[\int \delta\tilde{\boldsymbol{\theta}} d\boldsymbol{P}\right] \\ &= -\boldsymbol{R}^{-1}\left[\int \exp\left(-\frac{1}{\lambda}\boldsymbol{S}(\tilde{\boldsymbol{\theta}})\right)\delta\tilde{\boldsymbol{\theta}} d(\delta\tilde{\boldsymbol{\theta}})\right],\end{aligned} \tag{4}$$

where the probability metric $\boldsymbol{P}$ represents $\exp\left(-\frac{1}{\lambda}\boldsymbol{S}(\tilde{\boldsymbol{\theta}})\right)$ and the state-dependent cost is defined as $\boldsymbol{S}(\tilde{\boldsymbol{\theta}}) = \sum_{i=1}^{N} q(\tilde{\boldsymbol{\theta}}_i^\top)$. Using the estimated gradient above, $\mathbb{E}\left[Q(\tilde{\boldsymbol{\theta}})\right]$ can be minimized by leveraging generated stochastic noisy trajectories [7].

B. The Optimization Algorithm of STODI

In this section, we detail the implementation of the STODI algorithm on a 7-DOF robotic arm. The trajectory is represented by a sequence of positions for the robotic joints $\boldsymbol{\theta} \in \mathbb{R}^{N\times 7}$. We design our framework based on STOMP [7], which generates K noise sequences $\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_K$, sampled from $\mathcal{N}(\mathbf{0}, \boldsymbol{R}^{-1})$ and adds them into the current iterative trajectory separately to explore trajectories with lower costs, as shown in line 5 to 16 of Algorithm 1. However, the performance of STOMP entirely depends on the quality of the generated noise. In other words, if all generated noisy trajectories score high costs, the algorithm's iterative process will make the current trajectory worse, even exerting a profound influence on subsequent iterations. Previous works focused on improving the format of the generated noise [23], by contrast, we reconstruct the iterative process to overcome these shortcomings.

We categorize the iterative trajectory into three types: the best trajectory $\boldsymbol{\theta}_b$, the distal exploration trajectory $\boldsymbol{\theta}_d$, and the proximal searching trajectory $\boldsymbol{\theta}_p$. The best trajectory $\boldsymbol{\theta}_b$ records the trajectory with the minimum cost encountered so far. The distal exploration trajectory $\boldsymbol{\theta}_d$ updates at all iterations regardless of whether the cost increases. This strategy enables the algorithm to explore the global optimal solution and jump out of the local optimal solution. The proximal searching trajectory $\boldsymbol{\theta}_p$ is responsible for exploring possible better trajectories near the current best trajectory. By employing $\boldsymbol{\theta}_p$, we can thoroughly search the solution space near the current best solution and have a better chance of generating better noisy trajectories. Furthermore, we periodically update $\boldsymbol{\theta}_p$ as shown in line 24 of Algorithm 1, thereby preventing potential degradation of $\boldsymbol{\theta}_p$ due to accumulated noise.

To enhance the stability of the STODI, we retain low-cost trajectories and reuse them to improve the quality of generated noisy trajectories. As a complement to line 12 of Algorithm 1, we replace high-cost noisy trajectories with reused trajectories and modify corresponding values in matrices $\boldsymbol{S}$ and $\boldsymbol{P}$, consistent with (4) and line 9 to 10. Note that the number of reused trajectories n should be strictly less than the number of noisy sequences K. Generating new noisy trajectories is crucial to ensure the estimated gradient retains its exploration. In line 17 to 20, we calculate the iterative trajectory's cost and record low-cost trajectories to improve the quality of reused trajectories. We initialize the reused trajectories with infinite cost values. During the iterative process, whenever the iterative trajectory's cost becomes less than the maximum cost value in the reused trajectories, a replacement operation is triggered. The details of the STODI algorithm can be seen in Algorithm 1.

Algorithm 1: The Optimization Algorithm of STODI

```
   Input: An initial trajectory vector $\boldsymbol{\theta}_{init} \in \mathbb{R}^{N\times 7}$;
   Output: Optimized trajectory $\boldsymbol{\theta}_b \in \mathbb{R}^{N\times 7}$;
 1 Predefine: A state-dependent cost function $q : \mathbb{R}^7 \to \mathbb{R}$,
     a given matrix $\boldsymbol{R} \in \mathbb{R}^{N\times N}$;
 2 Initialize iterative trajectories $\boldsymbol{\theta}_b, \boldsymbol{\theta}_d, \boldsymbol{\theta}_p = \boldsymbol{\theta}_{init}$;
 3 Initialize reused trajectory set: $\{\boldsymbol{\theta}_{reused,i}\}_{i=1}^{n}$;
 4 while Convergence of $Q(\boldsymbol{\theta}_b)$ in (1) is not achieved do
 5     Create $K$ noisy sequences $\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_K$, where
         $\boldsymbol{\epsilon}_k \in \mathbb{R}^{N\times 7}, [\boldsymbol{\epsilon}_k]_j \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}^{-1}), k = 1, \ldots, K, j = 1, \ldots, 7$;
 6     for $\boldsymbol{\theta}$ in $\{\boldsymbol{\theta}_d, \boldsymbol{\theta}_p\}$ do
 7         Create $K$ noisy trajectories $\tilde{\boldsymbol{\theta}}^1, \ldots, \tilde{\boldsymbol{\theta}}^K$ with
             each $\tilde{\boldsymbol{\theta}}^k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$;
 8         for $k = 1, \ldots, K$ do
 9             Compute $\boldsymbol{S}(\tilde{\boldsymbol{\theta}})_{k,j} = q([\tilde{\boldsymbol{\theta}}^k]_j^\top), \boldsymbol{S} \in \mathbb{R}^{K\times N}$;
10             Compute
                 $\boldsymbol{P}(\tilde{\boldsymbol{\theta}})_{k,j} = \frac{e^{-\frac{1}{\lambda}S(\tilde{\boldsymbol{\theta}})_{k,j}}}{\sum_{k=1}^{K}[e^{-\frac{1}{\lambda}S(\tilde{\boldsymbol{\theta}})_{k,j}}]}, \boldsymbol{P} \in \mathbb{R}^{K\times N}$;
11         end
12         Replace high-cost noisy trajectories with reused
             trajectories;
13         for $j = 2, \ldots, N-1$ do
14             Compute $\delta\boldsymbol{\theta}_j^\top = \sum_{k=1}^{K} \boldsymbol{P}(\tilde{\boldsymbol{\theta}})_{k,j}[\boldsymbol{\epsilon}_k]_j^\top$;
15         end
16         Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{R}^{-1}\delta\boldsymbol{\theta} \in \mathbb{R}^{N\times 7}$;
17         Compute $Q(\boldsymbol{\theta})$;
18         Compute $m = \arg\max_i Q(\boldsymbol{\theta}_{reused,i})$;
19         if $Q(\boldsymbol{\theta}) < Q(\boldsymbol{\theta}_{reused,m})$ then
20             $\boldsymbol{\theta}_{reused,m} \leftarrow \boldsymbol{\theta}$;
21     end
22     Update $Q(\boldsymbol{\theta}_b) \leftarrow \min\{Q(\boldsymbol{\theta}_d), Q(\boldsymbol{\theta}_p)\}$;
23     Update $\boldsymbol{\theta}_b \leftarrow \arg\min_{\boldsymbol{\theta}}\{Q(\boldsymbol{\theta}_d), Q(\boldsymbol{\theta}_p)\}$;
24     Update $\boldsymbol{\theta}_p \leftarrow \boldsymbol{\theta}_b$ at a scheduled frequency.
25 end
```

Note that the time interval of discrete trajectories remains constant, and the length of the discrete sequence, denoted by N, represents the duration of the robot's motion. By adjusting the value of N in the initial trajectory, we can control the overall velocity of the robot's execution. In conclusion, the

Algorithm 2: DTW Algorithm

```
Input: Trajectory 1 $A \in \mathbb{R}^{N\times 3}$, Trajectory 2
       $\hat{A} \in \mathbb{R}^{\hat{N}\times 3}$;
Output: similarity measures $\mathrm{DTW}(A, \hat{A})$;
1  Initialization DTW array $S \in \mathbb{R}^{(N+1)\times(\hat{N}+1)}$;
2  for $i = 0, ..., N$ do
3      for $j = 0, ..., \hat{N}$ do
4          $S_{i,j} = \inf$;
5      end
6  end
7  Assign $S_{0,0} = 0$;
8  for $i = 1, ..., N$ do
9      for $j = 1, ..., \hat{N}$ do
10         $cost = d(A_i^\top, A_j^\top)$;
11         $S_{i,j} = cost + \min(S_{i-1,j}, S_{i,j-1}, S_{i-1,j-1})$;
12     end
13 end
14 $\mathrm{DTW}(A, \hat{A}) = S_{N,\hat{N}}$.
```

optimization algorithm of the STODI framework effectively merges spatial and temporal trajectory optimization, augmenting its exploration efficiency and optimization performance.

C. Measuring Similarity using DTW Algorithm

To equip the STODI with the ability to imitate demonstrations, we expand (1) by introducing an extra term that computes the resemblance between the current trajectory and the demonstration. We apply appropriate penalties to encourage the trajectory under optimization to imitate the demonstration. By balancing optimization and imitation components, we can leverage the STODI to address demonstration imitation problems effectively.

Dynamic Time Warping (DTW) is a dynamic programming algorithm applied in language processing [24], [25] and matching unaligned data [21], [26]. We leverage the DTW algorithm to characterize the degree of similarity in trajectories with different durations. Since the quality of imitation relies on the resemblance between the executor's trajectory and the demonstration in the 3-dimensional Euclidean space, we use $A \in \mathbb{R}^{N\times 3}, \hat{A} \in \mathbb{R}^{\hat{N}\times 3}$ to denote two motion trajectories of the end-effector in Euclidean space. Notably, these trajectories may not be temporally aligned:

$$
\begin{aligned}
A &= \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{bmatrix} = \begin{bmatrix} p_1^\top \\ \vdots \\ p_N^\top \end{bmatrix}, \\
\hat{A} &= \begin{bmatrix} \hat{x}_1 & \hat{y}_1 & \hat{z}_1 \\ \hat{x}_2 & \hat{y}_2 & \hat{z}_2 \\ \vdots & \vdots & \vdots \\ \hat{x}_{\hat{N}} & \hat{y}_{\hat{N}} & \hat{z}_{\hat{N}} \end{bmatrix} = \begin{bmatrix} \hat{p}_1^\top \\ \vdots \\ \hat{p}_{\hat{N}}^\top \end{bmatrix},
\end{aligned} \tag{5}
$$

where $p_i = (x_i, y_i, z_i)$ represents the i-th coordinate of end-effector trajectory in Euclidean space. We fix the start and end points as $(x_1, y_1, z_1) = (\hat{x}_1, \hat{y}_1, \hat{z}_1)$ and $(x_N, y_N, z_N) = (\hat{x}_{\hat{N}}, \hat{y}_{\hat{N}}, \hat{z}_{\hat{N}})$, respectively.

Details of the DTW can be seen in Algorithm 2. As a special note, $d(\cdot,\cdot)$ represents an arbitrary function that measures the distance between two vectors, e.g., Euclidean distance. This step is crucial for determining the similarity between two trajectories. Due to the dynamic programming property of the basic DTW algorithm, its time complexity is $O(N\hat{N})$. In practice, some improved versions can be implemented.

Additionally, employing forward and inverse kinematics in robotics allows for seamless conversions between joint space and task space coordinates, i.e. $\theta = IK(A), A = FK(\theta)$. By using DTW to measure the similarity between the trajectory $\tilde{\theta}$ and the demonstration θ_{demo}, we can add an extra term $q_d(\tilde{\theta}, \theta_{demo}) = \mathrm{DTW}\left(FK(\tilde{\theta}), FK(\theta_{demo})\right)$ into $Q(\tilde{\theta})$ to encourage imitation of the demonstration. The extended loss function can be formulated as:

$$
\begin{aligned}
Q(\tilde{\theta}) &= \sum_{i=1}^{N} q(\tilde{\theta}_i^\top) + \mathrm{DTW}\left(FK(\tilde{\theta}), FK(\theta_{demo})\right) \\
&+ \frac{1}{2}(\tilde{\theta}\mathbf{1}_M)^\top R(\tilde{\theta}\mathbf{1}_M) \\
&= \sum_{i=1}^{N}\left(q(\tilde{\theta}_i^\top) + \frac{1}{N} q_d(\tilde{\theta}, \theta_{demo})\right) + \frac{1}{2}(\tilde{\theta}\mathbf{1}_M)^\top R(\tilde{\theta}\mathbf{1}_M).
\end{aligned} \tag{6}
$$

D. Measuring Similarity with Spectrum Analysis

In some cases, the trajectory may be periodic,making it more important to capture the overall trend of the demonstration. Analyzing the trajectory in the time domain may ignore its periodic characteristics. Inspired by the fact that the frequency domain can extract more fundamental information, such as periodicity, we introduce spectrum analysis to measure the similarity of two trajectories in the frequency domain.

Firstly, we use Discrete Fourier Transform (DFT) to convert a finite sequence in the time domain $x \in \mathbb{R}^{M\times N}$ into a sequence in the frequency domain $X \in \mathbb{C}^{M\times N}$:

$$
X_{u,v} = \sum_{i=0}^{M-1}\left[\sum_{k=0}^{N-1} x_{i,k} e^{-j2\pi\frac{vk}{N}}\right] e^{-j2\pi\frac{ui}{M}}. \tag{7}
$$

We apply zero-padding to construct same-length sequences from (5). Zero-padding is a classic technique used in DFT to ensure that the input signal has a length that is equal to a power of two. Appending zeroes to the end of (5) does not alter the frequency content, it merely interpolates the output spectrum to the same length. Then, we have:

$$
\begin{aligned}
\mathcal{F} &= \begin{bmatrix} f_{0,0} & f_{0,1} & f_{0,2} \\ f_{1,0} & f_{1,1} & f_{1,2} \\ \vdots & \vdots & \vdots \\ f_{(N-1),0} & f_{(N-1),1} & f_{(N-1),2} \end{bmatrix} \in \mathbb{C}^{N\times 3}, \\
\hat{\mathcal{F}} &= \begin{bmatrix} \hat{f}_{0,0} & \hat{f}_{0,1} & \hat{f}_{0,2} \\ \hat{f}_{1,0} & \hat{f}_{1,1} & \hat{f}_{1,2} \\ \vdots & \vdots & \vdots \\ \hat{f}_{(N-1),0} & \hat{f}_{(N-1),1} & \hat{f}_{(N-1),2} \end{bmatrix} \in \mathbb{C}^{N\times 3}.
\end{aligned} \tag{8}
$$

The following result explains the relationship between signals in the time and frequency domains. By employing

Parseval's theorem, we establish the theoretical connections among our metrics.

Lemma 1 (Parseval's theorem [27])**.** *For the 2-dimensional DFT, we have:*

$$\sum_{i=0}^{M-1}\sum_{k=0}^{N-1}|x_{i,k}|^2 = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}|X_{u,v}|^2. \tag{9}$$

where $\boldsymbol{X} \in \mathbb{C}^{M\times N}$ is the DFT of $\boldsymbol{x} \in \mathbb{R}^{M\times N}$.

Parseval's theorem reveals the relationship between a signal in the time domain and its representation in the frequency domain. Both the sum of the squares of the time-domain signal and the sum of the squares of the magnitudes in the frequency domain represent the total energy of the signal. By using the DFT, spectrum analysis can better capture the overall properties of the signal, such as periodicity and shape information. We then introduce the extended version of Parseval's theorem to describe the relationship in the time and frequency domains between two different signals.

Lemma 2 (Parseval's theorem, (78) in [28])**.** *For the 2-dimensional DFT for different signals $\boldsymbol{x}$ and $\boldsymbol{y}$, we have:*

$$\sum_{i=0}^{M-1}\sum_{k=0}^{N-1}x_{i,k}y_{i,k} = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}X_{u,v}Y^*_{u,v}. \tag{10}$$

where $\boldsymbol{X} \in \mathbb{C}^{M\times N}$ is the DFT of $\boldsymbol{x} \in \mathbb{R}^{M\times N}$, $\boldsymbol{Y} \in \mathbb{C}^{M\times N}$ is the DFT of $\boldsymbol{y} \in \mathbb{R}^{M\times N}$, and $()^*$ means the conjugate operation.

Intuitively, by minimizing the difference between $\mathcal{F}$ and $\hat{\mathcal{F}}$ in (8), the demonstration trajectory can be effectively imitated in the 3-dimensional Euclidean space. We take the Mean Square Error (MSE) as a metric to calculate the difference between $\mathcal{F}$ and $\hat{\mathcal{F}}$. The MSE between vectors $\boldsymbol{y}, \hat{\boldsymbol{y}} \in \mathbb{R}^N$ is given by:

$$\mathrm{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{N}\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2. \tag{11}$$

To demonstrate that minimizing the MSE in the spectrum (MSES) is consistent with minimizing the DTW in the time domain, the following lemma is presented.

Lemma 3. *For different trajectories after zero-padding $\boldsymbol{A}, \hat{\boldsymbol{A}}$ and their spectrum representations $\mathcal{F}, \hat{\mathcal{F}} \in \mathbb{C}^{N\times 3}$, we have:*

$$\mathrm{DTW}(\boldsymbol{A}, \hat{\boldsymbol{A}}) \le \mathrm{MSES}(\mathcal{F}, \hat{\mathcal{F}}) \tag{12}$$

$$= \frac{1}{3N}\sum_{u=0}^{N-1}\sum_{v=0}^{2}(f_{u,v} - \hat{f}_{u,v})(f_{u,v} - \hat{f}_{u,v})^*. \tag{13}$$

Lemma 3 indicates that the $\mathrm{MSES}(\mathcal{F}, \hat{\mathcal{F}})$ can serve as an upper bound for $\mathrm{DTW}(\boldsymbol{A}, \hat{\boldsymbol{A}})$. Note that the MSES is only one method to measure the difference between $\mathcal{F}$ and $\hat{\mathcal{F}}$. Other approaches for calculating the difference can also be employed. Due to the strong correlation between frequency and time domains, the MSE methods often exhibit better interpretability. In lemma 4, we introduce another method to measure the difference in the power spectrum.

Lemma 4. *For different trajectories after zero-padding $\boldsymbol{A}, \hat{\boldsymbol{A}}$ and their spectrum representations $\mathcal{F}, \hat{\mathcal{F}} \in \mathbb{C}^{N\times 3}$, we have:*

$$\begin{aligned}\mathrm{DTW}(\boldsymbol{A}, \hat{\boldsymbol{A}}) \le{}& \underbrace{\frac{1}{3N}\sum_{u=0}^{N-1}\sum_{v=0}^{2}\left(\sqrt{f_{u,v}f^*_{u,v}} - \sqrt{\hat{f}_{u,v}\hat{f}^*_{u,v}}\right)^2}_{①} \\ &+ \underbrace{\frac{2}{3N}\sum_{u=0}^{N-1}\sum_{v=0}^{2}\left(\sqrt{f_{u,v}f^*_{u,v}\hat{f}_{u,v}\hat{f}^*_{u,v}} - f_{u,v}\hat{f}^*_{u,v}\right)}_{②}.\end{aligned} \tag{14}$$

The ① term in (14) is the MSE in the power spectrum (MSEPS) and the ② term can be proved strictly greater than zero. MSEPS, compared to MSES, is closer to DTW in magnitude. In our experiments, we find that optimizing with MSEPS leads to faster convergence compared to optimization with MSES and DTW.

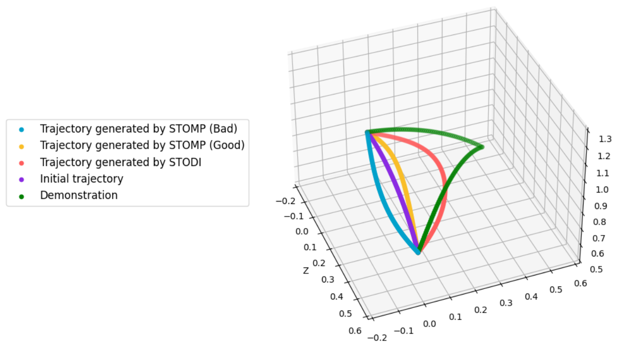
IV. Experiments on a Robot Arm

In this section, we verify the effectiveness of the STODI algorithm in Pybullet [8]. To show that our algorithm is competent for solving high-dimensional trajectory optimization problems, we use a 7-DOF Panda manipulator in the Pybullet simulator. We compare our algorithm with STOMP [7] and show its stability and exploration capability. We also implement various metrics to calculate trajectory similarity and demonstrate their performance in trajectory imitation, respectively. We then explain the superiority of describing the similarities of trajectories in the frequency domain. To emphasize the contributions of our approach to imitation and trajectory optimization, we finally deploy our algorithm in specific task scenarios that require high-speed motion and demonstrate its optimization capability.

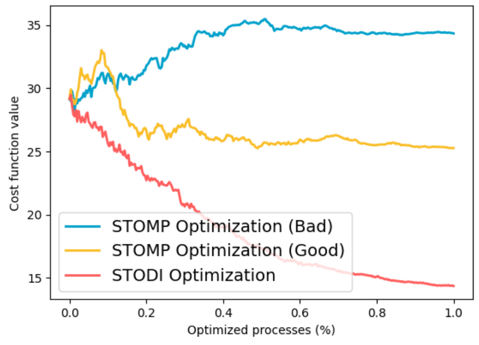
A. Performance Comparison with STOMP

To demonstrate the effectiveness of STODI, an experiment is designed where a robotic arm replicate a predetermined demonstration trajectory. In this setup, both STOMP and STODI are implemented to showcase their imitation capabilities, shown in Fig. 1. As depicted in Fig. 1(a), the demonstration trajectory is visualized in green, the initial trajectory in purple, and other different colors represent the trajectories optimized by different methods. In both STOMP and STODI, the cost function consists exclusively of the imitation term, calculated by the DTW algorithm described in Section III-C. By excluding other cost terms, we can effectively analyze the performance of these two algorithms in imitating the demonstration trajectory. With both algorithms having identical parameter settings and iterations, the trends of the costs during the optimization process are depicted in Fig. 1(b).

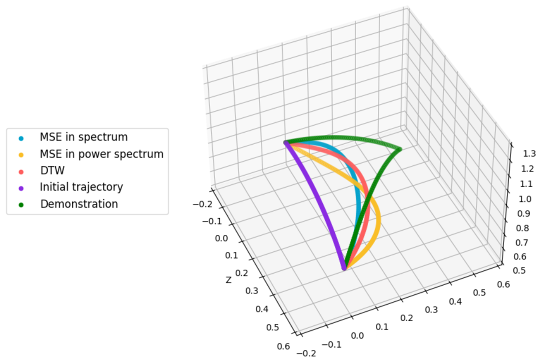
As shown in Fig. 1(b), the performance of STOMP heavily relies on the quality of the generated noise sequences. While STOMP retains some exploratory capability, it fails to prevent trajectories from worsening caused by random noise. When faced with unfavorable conditions, the cost of STOMP increases, leading to a greater deviation of the iterative trajectory



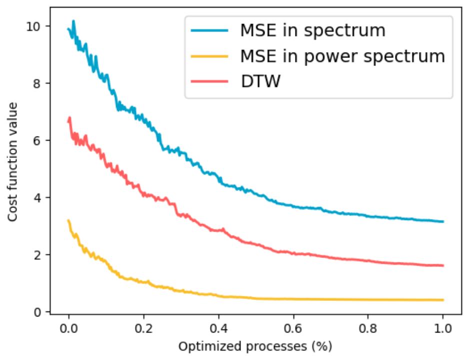
(a) 3D visualization of trajectories optimized by STOMP and STODI



(b) Trends of the costs under STOMP and STODI



(c) 3D visualization of trajectories optimized by STODI with three different metrics



(d) Trends of the costs under STODI with three different metrics

Fig. 1. (a) and (b) show comparisons of the performance of the trajectory optimization under the STOMP algorithm and the STODI algorithm. (c) and (d) show the imitating ability of STODI by applying three different metrics.

from the demonstration trajectory. In most cases, the best outcome achieved by STOMP only slightly approximates the demonstration trajectory. Following the iterative process described in Section III-B, STODI maintains random exploration capabilities during optimization while ensuring stability. It consistently produces optimization results that are at least as good, if not better, than that shown in the Fig. 1(a). By comparing the optimized trajectories generated by STOMP and STODI, we have verified the outstanding performance of STODI in optimization and stability.

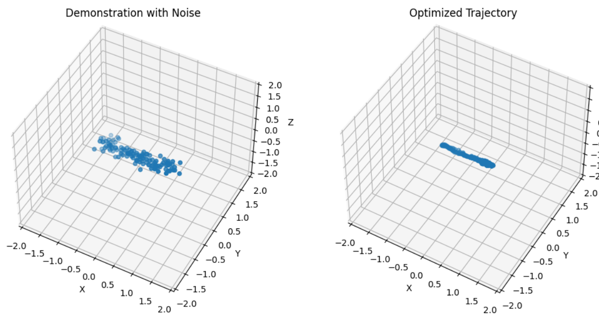
The STODI is further evaluated by incorporating different cost functions: DTW, MSES, and MSEPS. The three-dimensional trajectories optimized by these respective functions are shown in Fig. 1(c), with the corresponding costs depicted in Fig. 1(d). The results indicate that while the performances in three-dimensional trajectory optimization among these functions are similar, MSEPS converges to the final result more rapidly. DTW and MSES exhibit close optimization effects, consistent with the analysis in Lemma 3.

B. The Benefits of Spectrum Analysis

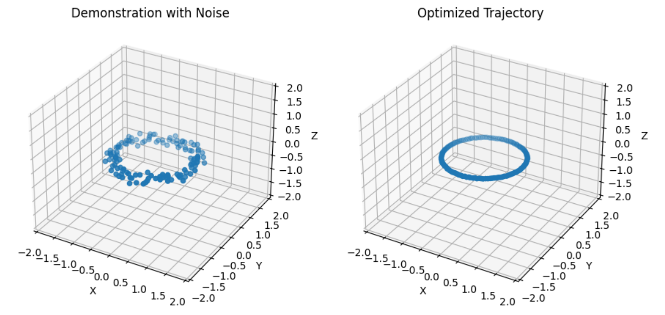
In this section, we verify the advantages of replacing DTW with spectrum analysis in Section III-D. The DFT operation can be seen as an explainable encoder, converting a time-domain sequence into a frequency-domain spectrum to distill out overall trajectory information. Processing a signal in the frequency domain, such as using a filter to remove noise, can improve the overall quality of the signal. In contrast, it is hard to change the overall characteristic of the signal by fine-tuning points in the time domain. We further consider the impact of noise on the various trajectory shapes in the samples and propose several denoising methods along with corresponding analyses, shown in Fig. 2.

As shown in Fig. 2(a), we add the white Gaussian noise $\mathcal{N}(\mathbf{0}, 0.1\boldsymbol{I}_3)$ to every point $\boldsymbol{p}_i$ in a linear trajectory $\boldsymbol{\theta}$. After converting $\boldsymbol{\theta}$ into a sequence $\mathcal{F}$ in the frequency domain, we scale the sequence, as shown in (15), suppressing the white Gaussian noise at full frequency and retaining the shape of the straight line.

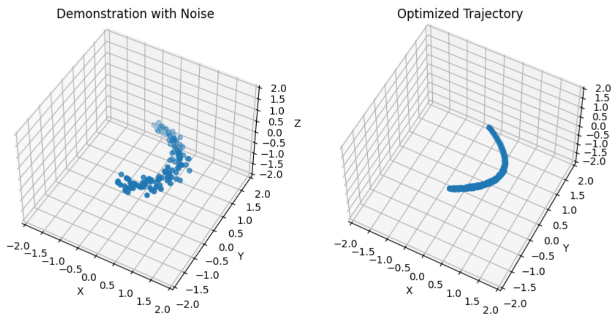
$$f_{i,k} \leftarrow \frac{f_{i,k}}{\max(|\mathcal{F}_k|)}. \tag{15}$$



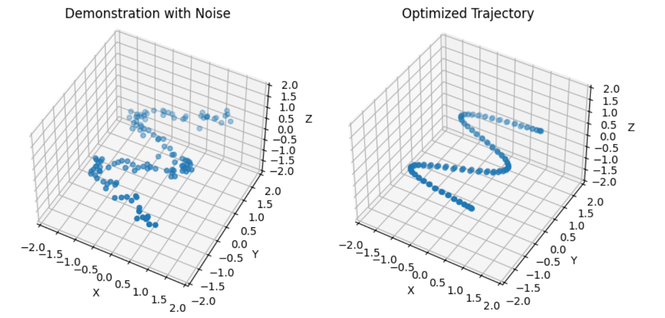
(a) Straight line



(b) Circle



(c) Semicircle



(d) M-shape

Fig. 2. The denoising performance of the filters in the frequency domain on different trajectories. We categorize the graphics into different types: line in (a), symmetric closed graphic in (b), and unclosed complex graphic in (c) and (d).

Nevertheless, the operation may induce scaling down or deformation of the trajectory. When applying it to a noisy circular trajectory, the circular path is transformed into an ellipse. To overcome this limitation, we employ the gain-controlling method described in (16) to attenuate frequencies with low amplitudes, converting the noisy circular trajectory into a clear one. The constant γ can be manually chosen, and we set $\gamma = 20$ in our experiment. The result is shown in Fig. 2(b).

$$f_{i,k} \leftarrow \frac{f_{i,k}}{\gamma}, \text{if } |f_{i,k}| \leq \gamma. \tag{16}$$

However, these methods fail when dealing with non-closed trajectory shapes. For both the semicircle and M-shape trajectories, when processed in the frequency domain, the trajectories tend to be transformed into closed shapes that are distinctly different from their original forms. Past studies have also shown that denoising methods in the frequency domain perform better in closed graphics [29]. To enhance the gain-controlling method's effectiveness for non-closed curves, we introduce a method that constructs desired curves by replicating trajectories in the time domain and merging them with the original trajectories through back-stitching. After polishing trajectories in the frequency domain and converting them back to the time domain, we can retain half of the trajectories to eliminate the doubling effect on trajectory length caused by the previous operation. Using the method above, the polished trajectories of the semicircle and M-shape are shown in Figs. 2(c) and 2(d).

Moreover, we analyze the time complexity of DTW and MSES. As we mention in Section III-C, the time complexity of DTW is $O(N\hat{N})$, in which N and $\hat{N}$ are the length of two trajectories respectively. To speed up DFT calculations and reduce the complexity of polynomial multiplication, we employ Fast Fourier Transformation (FFT) for Fourier transform in our experiments. For 2-dimensional signals in (5), the time complexity of FFT is $O(3N\log(3N))$ for trajectories with the dimension of $N \times 3$. The time complexity of MSE is $O(\max(N, \hat{N}))$. Therefore, if N and $\hat{N}$ are of the same magnitude, spectrum analysis is more time-efficient.

C. Performance on Specific Tasks

This part focuses on validating the performance of STODI within the simulation environment. The task involves the robotic arm striking a golf ball, requiring a golf club swing. The robot needs to imitate human demonstrations accurately and optimize the club's speed at the precise moment of striking the ball. The simulation results of club swing process is depicted in Fig. 3. To streamline the model, we directly utilize the end effector to strike the ball. Through remote control in the simulator, the robot records a demonstration of a polygonal shape shown in Fig. 3(a). The robot learns the motion trajectory of the swinging club and converts this motion into joint space-controlled sequences to mimic the trajectory. Images in Figs. 3(a) to 3(c) depict the preparatory movements of the swing club, outlining an arc to enhance swing velocity. Figs. 3(d) to 3(f) illustrate the slow-motion capture of a golf club striking the ball. Actually, in the

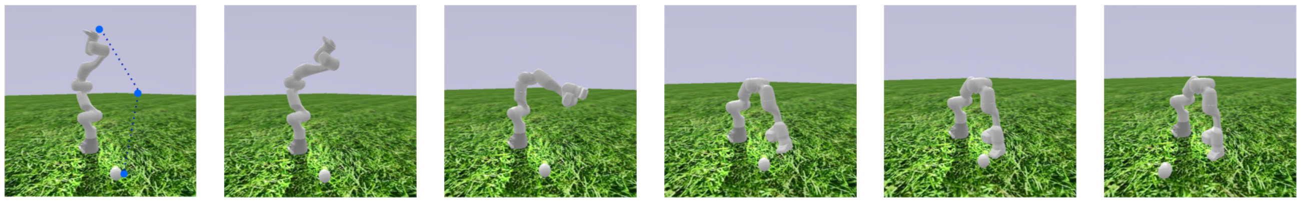


Fig. 3. The simulation process for Panda robotic club swing in PyBullet. The robot imitates the demonstration and performs like an expert human.

simulation , the swing-hit process unfolds swiftly, resembling the professional motion executed by an expert human.

V. Conclusion

In this paper, we present a novel approach, STODI, for guiding stochastic trajectory optimization by imitating human demonstrations. By leveraging the strengths of STOMP and incorporating three similarity metrics (DTW, MSES, and MSEPS), STODI demonstrates significant improvements in both exploration effectiveness and stability. Future research directions include learning from multiple demonstrations, enabling the robot to acquire diverse skills and adapt to different scenarios.

References

[1] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[2] C. W. Warren, “Global path planning using artificial potential fields,” in *IEEE International Conference on Robotics and Automation*, 1989, pp. 316–317.

[3] O. Brock and O. Khatib, “Elastic strips: A framework for motion generation in human environments,” *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.

[4] S. Raza, S. Haider, and M.-A. Williams, “Teaching coordinated strategies to soccer robots via imitation,” in *IEEE International Conference on Robotics and Biomimetics*, 2012, pp. 1434–1439.

[5] W. Li, Y. Wang, Y. Liang, and D. T. Pham, “Learning from demonstration for autonomous generation of robotic trajectory: Status quo and forward-looking overview,” *Advanced Engineering Informatics*, vol. 62, p. 102625, 2024.

[6] O. Koç, G. Maeda, and J. Peters, “Optimizing the execution of dynamic robot movements with learning control,” *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 909–924, 2019.

[7] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, “STOMP: Stochastic trajectory optimization for motion planning,” in *International Conference on Robotics and Automation*. IEEE, 2011, pp. 4569–4574.

[8] E. Coumans and Y. Bai, “Pybullet, a Python module for physics simulation for games, robotics and machine learning,” http://pybullet.org, 2016–2021.

[9] M. Zucker, N. Ratliff, A. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. Bagnell, and S. Srinivasa, “CHOMP: Covariant Hamiltonian optimization for motion planning,” *The International Journal of Robotics Research*, vol. 32, pp. 1164–1193, 2013.

[10] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, “Motion planning with sequential convex optimization and convex collision checking,” *The International Journal of Robotics Research*, vol. 33, pp. 1251–1270, 2014.

[11] B. Liu, G. Jiang, F. Zhao, and X. Mei, “Collision-free motion generation based on stochastic optimization and composite signed distance field networks of articulated robot,” *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7082–7089, 2023.

[12] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[13] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” *The Annual Research Report*, 1998.

[14] L. Jaillet and T. Siméon, “A PRM-based motion planner for dynamically changing environments,” in *IEEE International Conference on Intelligent Robots and Systems*, 2004, pp. 1606–1611.

[15] M. Faroni and D. Berenson, “Motion planning as online learning: A multi-armed bandit approach to kinodynamic sampling-based planning,” *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6651–6658, 2023.

[16] J. Zhang, M. Liu, S. Zhang, R. Zheng, and S. Dong, “A path planning approach for multi-AUV systems with concurrent stationary node access and adaptive sampling,” *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2343–2350, 2024.

[17] F. Torabi, G. Warnell, and P. Stone, “Behavioral cloning from observation,” *arXiv preprint arXiv:1805.01954*, 2018.

[18] A. Y. Ng and S. J. Russell, “Algorithms for inverse reinforcement learning,” in *International Conference on Machine Learning*, 2000, p. 663–670.

[19] G. Ye and R. Alterovitz, “Guided motion planning,” in *Robotics Research: The 15th International Symposium*, 2017, pp. 291–307.

[20] D. Koert, G. Maeda, R. Lioutikov, G. Neumann, and J. Peters, “Demonstration based trajectory optimization for generalizable robot motions,” in *IEEE International Conference on Humanoid Robots*, 2016, pp. 515–522.

[21] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, “Guiding trajectory optimization by demonstrated distributions,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 819–826, 2017.

[22] E. Theodorou, J. Buchli, and S. Schaal, “Reinforcement learning of motor skills in high dimensions: A path integral approach,” in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2397–2403.

[23] M. Mukadam, X. Yan, and B. Boots, “Gaussian process motion planning,” in *IEEE International Conference on Robotics and Sutomation*, 2016, pp. 9–15.

[24] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[25] Y. Permanasari, E. H. Harahap, and E. P. Ali, “Speech recognition using dynamic time warping (DTW),” in *Journal of Physics: Conference Series*, 2019, p. 012091.

[26] J. Van Den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X.-Y. Fu, K. Goldberg, and P. Abbeel, “Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations,” in *IEEE International Conference on Robotics and Automation*, 2010, pp. 2074–2081.

[27] M.-A. Parseval, “Mémoire sur les séries et sur l’intégration complète d’une équation aux différences partielles linéaires du second ordre, à coefficients constants,” *Mém. prés. par divers savants, Acad. des Sciences, Paris,(1)*, vol. 1, no. 638-648, p. 42, 1806.

[28] N. Baddour, “Discrete two-dimensional Fourier transform in polar coordinates part I: Theory and operational rules,” *Mathematics*, vol. 7, no. 8, p. 698, 2019.

[29] J. Li, *Shape recognition by curvature changing points and Fourier transform*. Western Michigan University, 1987.