Trajectory Redundancy in Point-to-Point Tracking and Applications to Obstacle Avoidance

Shou-Han Zhou, Ying Tan, Denny Oetomo, Chris Freeman and Iven Mareels

***Abstract*— For tasks which require a robot to track some particular points along a trajectory (instead of the whole trajectory), there exits redundancy. This redundancy, which results from the robot tracking more points than that required for the task, is called trajectory redundancy. This redundancy results in a increase in the feasibility of the task, enabling the possibility of the robot to obtain better performance by satisfying secondary objectives whilst performing the primary objective of tracking the target points.**

Point-to-point learning control (LC) has been shown to be an effect tool to deal with the trajectory redundancy since it has the ability to fully explore the increased feasibility resulting from such redundancy. This is done using the decomposition technique, which is widely used in solving kinematic redundancy, where the primary and secondary tasks are achieved independently using the appropriate LC algorithms. In order to apply the proposed point-to-point algorithms for tasks such as tracking a target point whilst avoiding an obstacle, the algorithms are modified and implemented in an online fashion. Simulation results shows good performance from the proposed point-to-point LC online algorithms.

I. INTRODUCTION

In the literature, redundancy generally refers to kinematic redundancy. In this case, the actuation command in the joint space which is prescribed for the robot to track a particular trajectory in task space is not unique. Current frameworks propose methods to decompose the actuation commands into those which affect motion in the task space and those which does not affect motion in the task space [1][2][3]. The actuation commands which affect task space motion are used to perform the primary task, leaving the rest of the actuation commands to be used for secondary task such as minimization of joint variations [4] or joint obstacle avoidance [5][6].

When a robot is required to perform a task, conventional controller design involves the proposition of a vector of input commands which drives the robot to track a prescribed trajectory. However, for tasks such as point to point tracking, the robot is only required to track specific target points at particular times. In such cases, the trajectory which the robot follows is redundant with respect to the task, resulting in the proposed vector of input commands to be also non-unique. This type of redundancy is called trajectory redundancy in this paper and it exists for both kinematically redundant and non-redundant manipulators.

This paper presents a method of decomposing the vector of input commands for a task with trajectory redundancy in a similar manner as that of kinematic redundancy. To do this, the Operational Space Formulation (OSF) [4] is first used to develop an elegant relationship describing the dynamics between the trajectory and its corresponding vector of input commands. The vector is then decomposed into a vector of input commands which affects the tracking of the task-relevant points and a vector which does not affect the tracking of the target points. It is observed that such decomposition naturally agrees with the design of Learning Controllers (LC) [7]. Such controllers learn the tasks over iterations. However, LC is not suited for certain reactive tasks such as obstacle avoidance. These tasks require online controller design over time instead of over iterations. To address this, a novel method is proposed for online controller implementation.

This paper is outlined as follows. First, a review of the OSF and kinematic redundancy resolution is presented in Section II. Trajectory redundancy is then presented in Section III. It is shown in the section that the OSF presents a method to develop a linear relation between the trajectory and the corresponding vector of input commands. This enables a clear separation of the relation into its range and null space components. Exploiting the two components, a point-to-point LC updating law is designed in Section IV to enable the robot end-effector to accomplish the primary task of tracking and a secondary task using the vector of inputs which does not affect the target point tracking. An on-line implementation of the proposed control law is applied in Section IV-B to enable a manipulator to perform the task of obstacle avoidance.

II. BACKGROUND

In this paper, the motion control problem with trajectory redundancy is presented and resolved in the task space following OSF [4], which not only represents but also decouples the dynamics of the manipulator in task space.

Denote the set of real numbers as $\mathbb{R}$, and the set of integers as $\mathbb{N}$. The iteration number is denoted as k which, by definition, satisfies $k \in \mathbb{N}_{\geq 0}$. For any vector $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}$ and $\mathbf{x}^k$ refers to the vector at the k^{th} iteration. For any matrix $A \in \mathbb{R}^{n\times m}$, $\underline{\sigma}(A)$ is the minimum singular value of A, $\overline{\sigma}(A)$ is the maximum singular value of A,

S. H. Zhou and D. Oetomo are with Mechanical Engineering Department, the University of Melbourne, Parkville, VIC 3010, Australia. s.zhou@pgrad.unimelb.edu.au and doetomo@unimelb.edu.au. Y. Tan is with the Electrical and Electronic Engineering Department, the University of Melbourne, Parkville, VIC 3010, Australia. yingt@unimelb.edu.au. C. Freeman is with the School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK. cf@ecs.soton.ac.uk. I. Mareels is with the School of Engineering, the University of Melbourne, Parkville, VIC 3010, Australia i.mareels@unimelb.edu.au.

This work is supported by the Australian Research Council (ARC) under Future Fellow Project: FT0991385

$A^{\dagger} = A(AA^T)^{-1}$ is the generalized inverse of A and $\Xi(A) = I - A^{\dagger}A$ is the null-space projection of A. For the identity and matrix I, I_p denotes a $p \times p$ identity matrix. Similarly, 0_p denotes $p \times p$ matrix of zeros.

The joint space dynamics of a manipulator with n degrees of freedoms (DOFs) is described by

$$\boldsymbol{\Gamma} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{g}(\boldsymbol{q}), \tag{1}$$

where the states $\boldsymbol{q} \in \mathbb{R}^n$ is the vector representing the n joint coordinates, $M(\boldsymbol{q}) \in \mathbb{R}^{n \times n}$ is the kinetic energy matrix, $\mathbf{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is the vectors of the Coriolis and centrifugal forces and and $\mathbf{g}(\boldsymbol{q}) : \mathbb{R}^n \to \mathbb{R}^n$ represents the vector of gravity. $\boldsymbol{\Gamma} \in \mathbb{R}^n$ is the vector of generalized joint torque.

Under the OSF, the dynamics of the manipulator of Equation (1) in task space is modeled as

$$\mathbf{F} = \Lambda(\boldsymbol{x})\ddot{\boldsymbol{x}} + \boldsymbol{\mu}(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \mathbf{p}(\boldsymbol{x}), \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^m$ is the number of DOFs of the operation task, which is less than or equal to the number of DOFs of the manipulator end-effector, $\Lambda(\boldsymbol{x}) \in \mathbb{R}^{m \times m}$ is the kinetic energy matrix as observed from the operational space, $\boldsymbol{\mu}(\boldsymbol{x}, \dot{\boldsymbol{x}}) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ is the vectors of the Coriolis and centrifugal forces and $\mathbf{p}(\boldsymbol{x}) : \mathbb{R}^m \to \mathbb{R}^m$ represents the vector of gravity, all acting in operational space. Moreover $\mathbf{F} \in \mathbb{R}^m$ is the vector of generalized forces as observed from operational space and is the input vector to the manipulator plant under the framework. It is related to the generalized torque vector through the relationship

$$\boldsymbol{\Gamma} = J(\boldsymbol{q})^T \mathbf{F}, \tag{3}$$

where $J(\boldsymbol{q}) \in \mathbb{R}^{m \times n}$ is the Jacobian matrix relating the end-effector and joint velocities.

The OSF enables the control of the end-effector through decoupling the task space dynamics in the task degrees of freedom and designing the controller depending on the task. This is done by:

$$\mathbf{F} = \hat{\Lambda}(\boldsymbol{x})\boldsymbol{\nu} + \hat{\boldsymbol{\mu}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \hat{\mathbf{p}}(\boldsymbol{x}), \tag{4}$$

where $\boldsymbol{\nu} \in \mathbb{R}^m$ represents the task space command to the decoupled system and is designed depending on the given task, $\hat{}$ represents estimates of the model parameters. Note that for clarity of this paper, $\boldsymbol{\nu}$ is used instead of $\mathbf{F}^*$ found in traditional OSF literature [4].

Assuming perfect estimation of the task space parameters, such that $\hat{(\cdot)} = (\cdot)$, the system is fully decoupled and the end effector dynamics is equivalent to that of a single unit mass. The position $\boldsymbol{x}$ and velocity $\dot{\boldsymbol{x}}$ of the manipulator end-effector is related to the input $\boldsymbol{\nu}$ through the following continuous-time linear-time-invariant (LTI) system [8]:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{x} \\ \ddot{\boldsymbol{x}} &= \boldsymbol{\nu} \end{aligned} \tag{5}$$

whose the exact discrete-time equivalent sampled at fixed time step h is determined as

$$\begin{bmatrix} \boldsymbol{x}(j+1) \\ \dot{\boldsymbol{x}}(j+1) \end{bmatrix} = A \begin{bmatrix} \boldsymbol{x}(j) \\ \dot{\boldsymbol{x}}(j) \end{bmatrix} + B\boldsymbol{\nu}(j), \tag{6}$$

where $A = \begin{bmatrix} I_{m\times m} & hI_{m\times m} \\ 0 & I_{m\times m} \end{bmatrix}$ and $B = \begin{bmatrix} \frac{h^2}{2} I_{m\times m} \\ hI_{m\times m} \end{bmatrix}$ and j is the sampling instant. Both position and velocity of the robot end-effector are considered as the output of the system, that is, $\mathbf{z} = \begin{bmatrix} \boldsymbol{x} & \dot{\boldsymbol{x}} \end{bmatrix}^T \in \mathbb{R}^p$ where $p = 2m$ has twice the dimensions of the operational space size.

In a kinematically redundant robot, i.e. when $n > m$, the corresponding set of joint torques $\boldsymbol{\Gamma}$ is not unique for a given end-effector task space command $\boldsymbol{\nu}$. The OSF provides an elegant method to decompose the joint torques into those which affect the end-effector task and those which do not, enabling the control of the end-effector task to be independent of the internal joint motion. Hence the joint torques can be represented as

$$\begin{aligned} \boldsymbol{\Gamma} &= \boldsymbol{\Gamma}_R + \boldsymbol{\Gamma}_N, \\ \boldsymbol{\Gamma}_R &= J(\boldsymbol{q})^T \mathbf{F}, \qquad (7) \\ \boldsymbol{\Gamma}_N &= (I - \overline{J}(\boldsymbol{q})J(\boldsymbol{q}))^T \boldsymbol{\tau}, \qquad (8) \end{aligned}$$

where $\boldsymbol{\Gamma}_R$ and $\boldsymbol{\Gamma}_N$ are the range and null space torques, respectively; $\boldsymbol{\tau}$ is the gradient descent of the cost function to be minimised in order to satisfy the secondary task.

The null space torque $\boldsymbol{\Gamma}_N$ is obtained by projecting $\boldsymbol{\tau}$ onto the null space of the dynamically consistent generalized inverse $\overline{J}(\boldsymbol{q})$ [9] where, for the given dynamical system in (2), $\overline{J}(\boldsymbol{q}) = (M(\boldsymbol{q}))^{-1} J(\boldsymbol{q}))^T \Lambda(\boldsymbol{q})$ and $\Lambda(\boldsymbol{q}) = (J(\boldsymbol{q})M(\boldsymbol{q}))^{-1} J(\boldsymbol{q})^T)^{-1}$. The cost function to be minimised would dictate the desired behaviour of the internal joint motion (secondary task), such as joint energy minimization [10], joint collision avoidance [6] and joint movement constraints [2] without conflicting the end-effector (primary) task.

In the next section, the idea of trajectory redundancy is formally introduced. The decoupling of the range and null space projections reviewed in this section is extended to the case of trajectory redundancy. The redundancy occurs on the task space and therefore instead of the generalised torque, it is the task space command $\boldsymbol{\nu}$ that is now decomposed.

III. Trajectory Redundancy

Trajectory redundancy occurs when the task is to track particular points rather than a whole trajectory. An extreme example is the task of arriving at the goal position from an initial location of end-effector. In this case, the prescribed trajectory is redundant with respect to the task since there exist multiple trajectories which pass through the desired point. Note that the task and the trajectories are specified in task/operational space. Using the OSF, the task space dynamics is decoupled in the task space degrees of freedom, under the assumption of perfect parameter estimation, leading to the system shown in (5), which is subsequently discretized in (6). From the discretized system, an elegant relationship between the task space input vector and the corresponding task space motion vector, i.e. position and velocity vectors, is determined in Section III-A. A factor is then introduced to identify the task-relevant points along the trajectory to be tracked in Section III-B. This factor is used along with the relationship in Section III-A to decompose the task space input $\boldsymbol{\nu}$ into those which affects the task-relevant points and those which do not in Section III-C, analogous to the range and null space decomposition in kinematically redundant robots, respectively, as presented in Section II.

A. Trajectory Dynamic Behavior

The OSF use the control structure given in Equation (4) achieve the decoupling of dynamics in task space. In the event of perfect estimation, the relationship between the manipulator output $\mathbf{z}$, representing the manipulator end-effector position and velocity, and the task space command $\boldsymbol{\nu}$ is governed by Equation (5) whose exact discrete-time equivalent model (6) can be re-written as:

$$\mathbf{z}(j+1) = A\mathbf{z}(j) + B\boldsymbol{\nu}(j), j = 0, \ldots, \mathcal{N}-1, \tag{9}$$

where $\mathcal{N}$ is the total number of samples taken across the time interval $[0\ T]$. The motion trajectory is described by a vector of end-effector outputs $\overrightarrow{\mathbf{z}}_T$ defined as

$$\overrightarrow{\mathbf{z}}_T = \begin{bmatrix} \mathbf{z}(0)^T & \mathbf{z}(1)^T & ... & \mathbf{z}(\mathcal{N}-1)^T \end{bmatrix}^T \in \mathbb{R}^{p\cdot\mathcal{N}}. \tag{10}$$

The vector of input commands required to drive the manipulator is given by $\overrightarrow{\boldsymbol{\nu}}_T$ defined as

$$\overrightarrow{\boldsymbol{\nu}}_T = \begin{bmatrix} \boldsymbol{\nu}(0)^T & \boldsymbol{\nu}(1)^T & ... & \boldsymbol{\nu}(\mathcal{N}-1)^T \end{bmatrix}^T \in \mathbb{R}^{m\cdot\mathcal{N}} \tag{11}$$

Given the dynamics between the end-effector motion and the command inputs in Equation (9), the relation between $\overrightarrow{\boldsymbol{\nu}}_T$ and $\overrightarrow{\mathbf{z}}_T$ is determined via the relationship

$$\overrightarrow{\mathbf{z}}_T = G\overrightarrow{\boldsymbol{\nu}}_T, \tag{12}$$

where the mapping $G \in \mathbb{R}^{p\cdot\mathcal{N}\times m\cdot\mathcal{N}}$ is defined as

$$G = \begin{bmatrix} B & 0 & \cdots & \cdots & 0 \\ AB & B & 0 & \cdots & 0 \\ A^2B & AB & B & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A^{\mathcal{N}-2}B & A^{\mathcal{N}-3}B & \cdots & \cdots & B \end{bmatrix}. \tag{13}$$

This relation is used to present trajectory redundancy in the next section.

B. Trajectory Redundancy

The reference trajectory is conventionally defined as a vector of desired outputs

$$\overrightarrow{\mathbf{z}}_d = [\ \mathbf{z}_d(0)^T \quad \mathbf{z}_d(1)^T \quad ... \quad \mathbf{z}_d(\mathcal{N}-1)^T \quad]^T \ \in \mathbb{R}^{p\cdot\mathcal{N}} \tag{14}$$

The controller is required to determine a vector of input commands which enables the system output to track the reference trajectory. However, the class of tasks of interest in this paper are those which require a set of points to be followed at particular times. More specifically, the controller objective is to drive the manipulator end-effector to track L particular points at sample instances given by $0 \leq w_1 < w_2 < \cdots < w_L < \mathcal{N}-1$. The remaining points of the output vector are free variables. Considering that there exist multiple trajectories which pass through the points of interest, the trajectory which the manipulator follows is redundant with respective to the task.

To present this redundancy in the given problem formulation, a task matrix is first defined as $\Phi \in \mathbb{R}^{p\cdot L\times p\cdot\mathcal{N}}$

$$\Phi_{l_1,l_2} = \begin{cases} I_p & \textbf{if} \quad l_2 = w_i, \quad l_1 = 1, 2, \ldots L \\ 0_p & \textbf{otherwise} \end{cases}. \tag{15}$$

This matrix extracts the task-relevant points from the total vector of output $\overrightarrow{\mathbf{z}}_T$ and presents them in a task-relevant sub-vector

$$\Phi\overrightarrow{\mathbf{z}}_T = \left[\mathbf{z}(0)^T, \mathbf{z}(w_1)^T, \cdots, \mathbf{z}(w_L)^T\right]^T \in \mathbb{R}^{p\cdot L} \tag{16}$$

Similarly, desired points of the task trajectory to be tracked is determined as

$$\overrightarrow{\mathbf{z}}_{ref} = \Phi\overrightarrow{\mathbf{z}}_d = [\ \mathbf{z}_d(0)^T \quad \mathbf{z}_d(w_1)^T \quad \ldots \quad \mathbf{z}_d(w_L)^T]^T \tag{17}$$

where $\overrightarrow{\mathbf{z}}_{ref} \in \mathbb{R}^{p\cdot L}$ is the vector of target points. Using the relation given in Equation (12), the objective of the controller is determined as finding a vector of input commands $\overrightarrow{\boldsymbol{\nu}}_T$ such that the error between the system output, i.e. the end-effector states, and the task-relevant points is minimized at the sample instances.

$$\begin{aligned} \min_{\overrightarrow{\boldsymbol{\nu}}_T} \mathcal{J}_1(\overrightarrow{\boldsymbol{\nu}}_T) &= \min_{\overrightarrow{\boldsymbol{\nu}}_T} \left\| \overrightarrow{\mathbf{z}}_{ref} - \Phi\overrightarrow{\mathbf{z}}_T \right\|^2 \\ &= \min_{\overrightarrow{\boldsymbol{\nu}}_T} \left\| \overrightarrow{\mathbf{z}}_{ref} - \Phi G\overrightarrow{\boldsymbol{\nu}}_T \right\|^2 \end{aligned} \tag{18}$$

where $\Phi G \in \mathbb{R}^{p\cdot L\times m\cdot\mathcal{N}}$ is the matrix governing the relation between $\overrightarrow{\boldsymbol{\nu}}_T$ and the desired target points.

It has been shown that the vector of input commands which solves the task given by Equation (18) is determined by the generalized inverse of the task-input relation [11]:

$$\overrightarrow{\boldsymbol{\nu}}_T = (\Phi G)^{\dagger}\overrightarrow{\mathbf{z}}_{ref}. \tag{19}$$

Remark 1: For discrete time representations, trajectory redundancy exists when the total number of samples $\mathcal{N}$ in the given time satisfies $m\cdot\mathcal{N} - \max\{d, (m-p)\cdot\mathcal{N}\} \geq p\cdot L$ where d is the relative degree of the system [7, Theorem 2]. This condition suggests that there exists a higher number of sampled points than the number of target points. The condition therefore guarantees the existence and non-uniqueness of the vector $\overrightarrow{\boldsymbol{\nu}}_T$ which drives the robot states through the target points.

C. Decoupling of the Task Input

Following the idea of range and null space decomposition in kinematic redundancy, the corresponding vector of input commands is decomposed into two components: (1) the vector of input commands which affects the tracking of the task relevant points $\Phi\mathbf{z}_T$, denoted in this paper as the tracking relevant vector $\overrightarrow{\boldsymbol{\nu}}_R \in \mathbb{R}^{m\cdot\mathcal{N}}$, and (2) the vector of input commands which does not affect tracking of the task relevant points, denoted in this paper as tracking independent vector $\overrightarrow{\boldsymbol{\nu}}_N \in \mathbb{R}^{m\cdot\mathcal{N}}$:

$$\overrightarrow{\boldsymbol{\nu}}_T = \overrightarrow{\boldsymbol{\nu}}_R + \overrightarrow{\boldsymbol{\nu}}_N$$

To achieve point to point tracking, the controller is designed to minimize the error between the task-relevant points and the target points at the specified time. Because only $\vec{\boldsymbol{\nu}}_R$ affects tracking of the target points, searching the vector of inputs $\vec{\boldsymbol{\nu}}_T$ which minimizes the tracking error is equivalent to searching for the vector $\vec{\boldsymbol{\nu}}_R$ within the set of tracking relevant vectors $\Omega_R \subseteq \mathbb{R}^{m\cdot\mathcal{N}}$

$$\min_{\vec{\boldsymbol{\nu}}_T} \mathcal{J}_1(\vec{\boldsymbol{\nu}}_T) = \min_{\vec{\boldsymbol{\nu}}_R\in\Omega_R} \mathcal{J}_1(\vec{\boldsymbol{\nu}}_T) \tag{20}$$
$$\mathcal{J}_1(\vec{\boldsymbol{\nu}}_T) = \left\|\mathbf{z}_{ref} - \Phi G\vec{\boldsymbol{\nu}}_T\right\|^2 = \left\|\vec{\mathbf{z}}_{ref} - \Phi G\vec{\boldsymbol{\nu}}_R\right\|^2$$

It has been determined that the vector for the tracking of target points given by (20) is determined by the generalized inverse of the task-input relation $((\Phi G)^\dagger)$ [11]

$$\vec{\boldsymbol{\nu}}_R^* = (\Phi G)^\dagger \vec{\mathbf{z}}_{ref} \tag{21}$$

where $(\Phi G)^\dagger = (\Phi G)(\Phi G(\Phi G)^T)^{-1} \in \mathbb{R}^{m\cdot\mathcal{N}\times p\cdot L}$. The tracking independent vector $\vec{\boldsymbol{\nu}}_N$, which does not affect tracking of the target points, can be used to satisfy the secondary tasks, such as obstacle avoidance. In this case, the controller is required to search within the set $\Omega_N \subseteq \mathbb{R}^{m\cdot\mathcal{N}}$ for the vector of input commands which satisfies the cost function $\mathcal{J}_2$ describing the secondary task

$$\min_{\vec{\boldsymbol{\nu}}_N\in\Omega_N} \mathcal{J}_2(\vec{\boldsymbol{\nu}}_R^* + \vec{\boldsymbol{\nu}}_N) \tag{22}$$

where Ω_N represents the set of vectors which does not affect the tracking of the target points. In this cost function, tracking relevant vector $\vec{\boldsymbol{\nu}}_R^*$ is a constant defined in Equation (21). The tracking independent vector $\vec{\boldsymbol{\nu}}_N$ can be determined as a projection of an arbitrary vector of input commands $\vec{\boldsymbol{\nu}}_O \in \mathbb{R}^{m\cdot\mathcal{N}}$ onto the null space of $(\Phi G)^\dagger$ using the matrix $\Xi(\Phi G) = (I_{m\cdot\mathcal{N}} - (\Phi G)^\dagger\Phi G) \in \mathbb{R}^{m\cdot\mathcal{N}\times m\cdot\mathcal{N}}$ of rank $m\cdot\mathcal{N} - p\cdot L$. The cost function of Equation (22) is therefore equivalent to solving the unconstrained optimization problem of

$$\min_{\vec{\boldsymbol{\nu}}_O} \mathcal{J}_2(\vec{\boldsymbol{\nu}}_R^* + \Xi(\Phi G)\vec{\boldsymbol{\nu}}_O). \tag{23}$$

A method of searching for the input vector from the cost function in Equation (23) is to use the gradient descent algorithm. The update rule is determined as:

$$\vec{\boldsymbol{\nu}}_O^{k+1} = \vec{\boldsymbol{\nu}}_O^k - \frac{\psi}{2}\nabla_{\vec{\boldsymbol{\nu}}_O}\mathcal{J}_2(\vec{\boldsymbol{\nu}}_R^* + \Xi(\Phi G)\vec{\boldsymbol{\nu}}_O^k) \tag{24}$$

where ψ is the update rate of the algorithm. Additionally, for objectives of the form $\mathcal{J}_2(\vec{\boldsymbol{\nu}}_R^* + \Xi(\Phi G)\vec{\boldsymbol{\nu}}_O^k) = \left\|R(\vec{\boldsymbol{\nu}}_R^* + \Xi(\Phi G)\vec{\boldsymbol{\nu}}_O^k) + c\right\|^2$, where R and c are the system parameters, it is possible to guarantee the convergence of the sequence $\vec{\boldsymbol{\nu}}_O^k$ by choosing the appropriate update rate. For gradient descent function (24), the update gain ψ is required to satisfy [7, Theorem 5].

$$0 < \psi < \min\left(\frac{2}{\overline{\sigma}((R)(R)^T)}\right) \tag{25}$$

The sequence of input trajectories $\vec{\boldsymbol{\nu}}_T^k$ is therefore determined as

$$\vec{\boldsymbol{\nu}}_T^k \quad = \vec{\boldsymbol{\nu}}_R^* + \Xi(\Phi G)\vec{\boldsymbol{\nu}}_O^k \tag{26}$$

and is used in the next section to design an online controller applicable to a manipulator.

IV. Point-to-Point LC Controller Implementation for Obstacle Avoidance

A. *On-line point-to-point LC controller design*

Point-to-point LC updates iteratively using (26) and ensures that the optimal input with respect to $\mathcal{J}_2$ can be found across iterations. However, under some situations, it is difficult to update the vector of inputs iteratively. One example is the task of obstacle avoidance which requires the robot to avoid a certain obstacle for all time instances. In this case, the robot cannot learn to "avoid" the obstacle from "experience" and it is preferable that the proposed point-to-point LC can be implemented on-line.

This paper proposes a novel method which implements the point-to-point LC online in a feed-forward fashion. Two facts are noted in the point-to-point LC algorithm (26):

Fact 1 The updating law (26) generates a sequence of output trajectories:

$$\vec{\mathbf{z}}_{ff}^k = G\vec{\boldsymbol{\nu}}_T^k. \tag{27}$$

This sequence is convergent and can be computed off-line for any $k \in \mathbb{N}_{\geq 0}$.

Fact 2 For any fixed iteration k, if a feedback controller along discrete-time domain is designed to track this sequence $\vec{\mathbf{z}}_{ff}^k$, it is possible to obtain a sequence, which is an approximation of $\left\{\vec{\mathbf{z}}_{ff}^k\right\}_{k\in\mathbb{N}_{\geq 0}}$ provided that the feedback controller settles down quickly enough.

Therefore, it is possible to combine the feedback controller with point-to-point LC to generate an online algorithm. The following notations are introduced

$$\boldsymbol{\zeta}^k(j) \quad := \quad \mathbf{z}_{ff}^k(j) - \mathbf{z}_T^k(j), \tag{28}$$
$$\boldsymbol{\nu}_{fb}^k(j) \quad := \quad -K\boldsymbol{\zeta}^k(j), \quad j = 0,\ldots,\mathcal{N}-1, \tag{29}$$

where K is a feedback gain matrix with an appropriate dimension. Here $\mathbf{z}^k(j)$ is the $j+1^{th}$ component of $\vec{\mathbf{z}}^k$. Moreover, we have

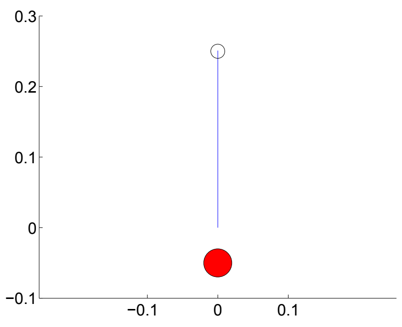
$$\boldsymbol{\nu}^k(j) \quad := \quad \boldsymbol{\nu}_{fb}^k(j) + \boldsymbol{\nu}_T^k(j), \quad j = 0,\ldots,\mathcal{N}-1, \tag{30}$$

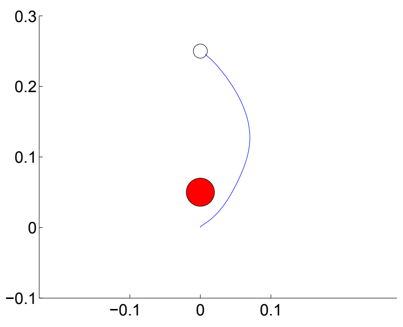
where $\boldsymbol{\nu}_{fb}^k(j)$ is obtained from (29), and $\boldsymbol{\nu}_T^k(j)$ is obtained from (26). In order to implement this updating law online, the following control law is used:

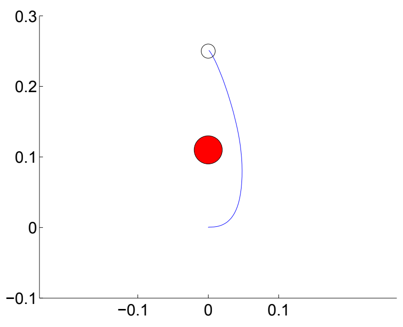
$$\boldsymbol{\nu}(j) \quad = \boldsymbol{\nu}^k(j), k = j, \quad j = 0,\ldots,\mathcal{N}-1 \tag{31}$$

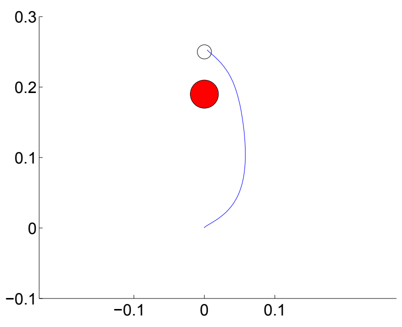
As observed in Figure 2, the controller input at sample j is chosen to be the sample j from the input sequence of iteration j.

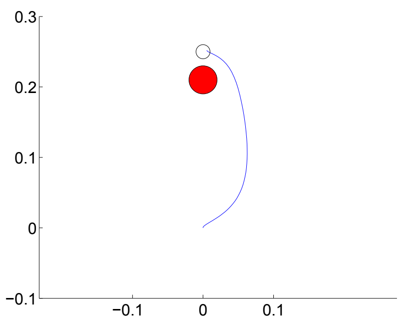
Remark 2: The updating law (26) is used in (30) as a feed-forward and serves as "model-based" prediction (or path planning). The feedback updating law in (29) serves as online "correction". To obtain good convergence performance using the updating law, the online "correction" term is required to be sufficiently fast such that the dynamics along the discrete-time domain converges quickly to the desired "set-point" computed from feed-forward updating law. This usually requires a high gain feedback controller.


(a) $\mu_O = -0.05m$


(b) $\mu_O = 0.05m$


(c) $\mu_O = 0.11m$


(d) $\mu_O = 0.19m$


(e) $\mu_O = 0.21m$

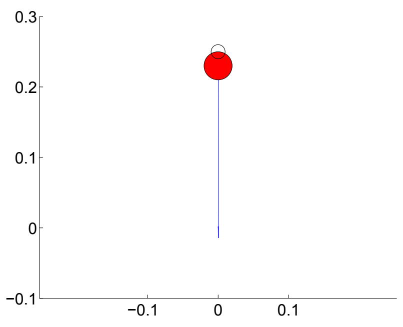

(f) $\mu_O = 0.23m$

Fig. 1: Trajectories of the end-effector given difference circle obstacles

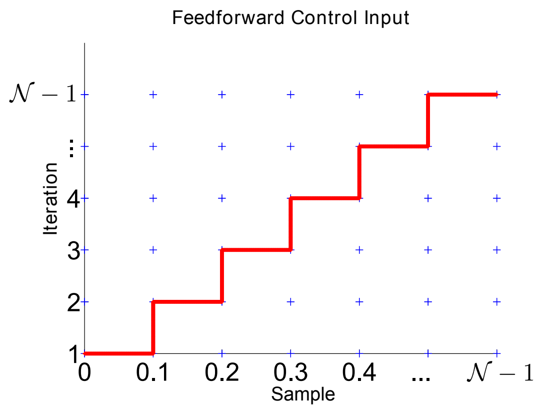


Fig. 2: The inputs used for the online controller

Robot Simulation Parameters	**Value**	**Units**
Link 1		
Length	1.93	m
Length to Center of Mass	0.165	m
Inertia	0.0141	kgm^2/rad^2
Mass	0.31	kg
Link 2		
Length	1.52	m
Length to Center of Mass	0.19	m
Inertia	0.0188	kgm^2/rad^2
Mass	0.34	kg

TABLE I: Table of Robot Parameters

Remark 3: Due to space limitation, the convergence analysis is omitted. Obviously, the convergence proof will use singular perturbation techniques [12] as there is a time-scale separation in discrete-time domain (feedback should be much faster than the feed-forward controller). The control input is computed on-line using (31). Therefore, only finite iterations ($\mathcal{N}$ iterations) are used. Due to existence of the feedback (an approximation along iteration domain) as well as finite iteration updating, the proposed updating law (30) ensures practical convergence (the output will converge to a neighborhood of the optimal value).

B. *Simulation Results*

In this paper, the controller defined in Equation (31) is applied to a simulation of a two-link robot. The parameters of the robot in the simulation are given in Table I.

The task of interest is planar reaching for a target point while avoiding an obstacle. Considering that the task is planar reaching with two DOFs, the joint configuration size is the same as that of the operational coordinates $m = n$. Therefore, there does not exist any kinematic redundancy in the manipulator, only trajectory redundancy caused by the task being reaching for a target point. This enables a transparent observation of the controller's use of trajectory redundancy. It is worth to note that this scheme is extendable to redundant manipulators. In this case, the trajectory redundancy decoupling is considered along with kinematic redundancy decoupling performed using the dynamically consistent inverse [4].

The control design for online trajectory redundancy resolution described in Section IV-A is applied to the problem of obstacle avoidance. The task to be carried out is for the robot to perform planar reaching for a target point $25cm$ away in $600ms$ whilst avoiding an obstacle. The obstacle chosen is a circle with a radius of $5cm$ located at a particular point along the straight line trajectory.

The cost function describing the obstacle $\mathcal{J}_2$ is defined as

$$\mathcal{J}_2(\overrightarrow{\boldsymbol{\nu}}_O) = \sum_{j=1}^{\mathcal{N}} \frac{1}{r(j)\sigma\sqrt{2\pi}} \exp\left(\frac{-0.5(r(j)-\mu)}{\sigma}\right)^{10} \quad (32)$$

where $r(j)$ is the desired distance of the robot end-effector from the obstacle center at sample j and is defined as $\left\|\overrightarrow{\boldsymbol{\nu}}_T^k\right\|$ where $\overrightarrow{\boldsymbol{\nu}}_T^k$ is defined in (26), $\sigma = 2cm$ is the radius of the circle obstacle and μ is the center of the obstacle. Under this cost function, the obstacle is represented as a bounded artificial potential field function [5]. Using gradient descent of Equation (24) and the control law of Equation (31), the control input at any time instant $\boldsymbol{\nu}(j)$ is able to be defined. The gains of the controller are chosen as $K = 50$ and $\psi = 5$.

The results of the simulation using the controller are observed in Figure 1. The target is a transparent circle while the obstacle is the filled circle. From the results, it is observed that if the target is not in the path between the start and target points, the end-effector follows the straight line path (Figure 1a). If the obstacle is in between the start and target points (Figure 1b to Figure 1e), the end-effector moves around the obstacle to the target. If the obstacle is at the target point, the end-effector moves along the straight line and stops before the obstacle (Figure 1f), which shows the practical convergence as discussed in Remark 3.

V. Conclusions

When given a task, traditional controllers drive the robot end-effector to follow a prescribed trajectory. However, for tasks such as point-to-point tracking, multiple trajectories are available to complete the task. This is called trajectory redundancy. This paper presents a method of analyzing trajectory redundancy in a similar manner to the resolution of kinematic redundancy in the literature. Through the use of OSF and discretizing the resulting dynamics, an elegant relationship is developed between the trajectory sequence of the end-effector and the corresponding task command sequence to the robot. This relationship forms a basis of decoupling the sequence of task commands into those which affects tracking and those which do not affect tracking.

Previously, this theory has been incorporated in the framework of point-to-point learning control. Such controller enables the determination of a vector of inputs which drive the manipulator to perform the task over iterations. However, point-to-point LC is not suitable for some applications such as obstacle avoidance. Online implementation of the algorithm is proposed, enabling the robot to perform such tasks. Simulation results are obtained and are shown to be able to incorporate different obstacles.

Future work includes analysis of the update gains K, ψ and their relationships rigorously in order to ensure convergence of the proposed method.

References

[1] J. Hollerbach, "Redundancy resolution of manipulators through torque optimization," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 308–316, Aug. 1987.

[2] P. Hsu, J. Hauser, and S. Sastry, "Dynamic control of redundant manipulators," *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, no. 6, pp. 183–187, 1988.

[3] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of Intelligent and Robotic Systems*, vol. 3, no. 3, pp. 201–212, 1990.

[4] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, Feb. 1987.

[5] ——, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, Mar. 1986.

[6] A. a. Maciejewski and C. a. Klein, "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 109–117, Sep. 1985.

[7] C. T. Freeman and Y. Tan, "Iterative Learning Control With Mixed Constraints for Point-to-Point Tracking," *IEEE Transactions on Control Systems Technology*, pp. 1–13, 2012.

[8] O. Egeland, "Task-space tracking with redundant manipulators," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 471–475, Oct. 1987.

[9] O. Khatib, "Inertial Properties in Robotic Manipulation: An Object-Level Framework," *The International Journal of Robotics Research*, vol. 14, no. 1, pp. 19–36, Feb. 1995.

[10] S.-w. Kim, K.-b. Park, and J.-j. Lee, "Redundancy resolution of robot manipulators using optimal kinematic control," *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 683–688, 1994.

[11] J. Peters, M. Mistry, and F. Udwadia, "A unifying methodology for the control of robotic systems," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, no. 1, 2005, pp. 1824 – 1831.

[12] H. K. Khalil, *Nonlinear systems*, 3rd ed. Upper Saddle River, N.J: Prentice Hall, 2002.