Determining Null-space Motion to Satisfy Both Task Constraints and Obstacle Avoidance

Liqin Zhu*, Huitan Mao+, Xiang Luo*, and Jing Xiao+

***Abstract*— It is a common practice to take advantage of redundancy in a manipulator to avoid obstacles while the end-effector is conducting a task-constrained motion, such as compliant motion for assembly. However, existing approaches rely on increasing the distance between the obstacles and the robot arm to avoid obstacles, which may result in a conflict between task motion and obstacle avoidance, i.e., when the rest of the arm avoids obstacles, the end-effector can no longer follow the task-constrained motion. We have observed that, in some cases, the robot arm has to decrease the distance to obstacles purposefully in order to both enable obstacle avoidance and the end-effector task constrained motion. This paper introduces a novel approach of null-space motion control to enable such manipulator motion to achieve both task objectives and obstacle avoidance, which is called the *coordination of task motion and self motion* (CTS) method. The method is implemented and tested on both a planar 4 degrees of freedom (DOF) manipulator in simulation and a spatial 7-DOF manipulator in real experiments, which show its effectiveness.**

I. Introduction

Achieving both task-constrained motion and obstacle avoidance is crucial for assembly and manufacturing using robot manipulators. There has been considerable study on how to make a redundant manipulator avoid obstacles while performing a task. The existing approaches include potential-field based, null-space based, and sampling-based approaches, configuration control methods, Jacobian transpose methods, and quadratic programming methods.

A. Related literature

Inspired by the idea of artificial potential field [1], potential-field based approaches [2], [3] use a vector field over velocity space to generate movement trajectories. The elastic strip framework introduced in [4] allows the suspension and resumption of task execution to enable other desired behaviors, such as obstacle avoidance. However, the task execution is interrupted in this case. One inherent problem associated with the potential-field based approaches is that attractive force from the goal and repulsive forces from different obstacles can conflict with one another in opposite directions, and therefore the effect of obstacle avoidance is reduced or the robot is stalled in local minima [3].

A representative null-space based method is gradient projection introduced in [5]. A scalar k is used to adjust the gradient projection in this method, and the value of k greatly affects the behavior of the manipulator. However, how to select a proper value for k to have both fast convergence of optimization and stable joint motions is not trivial [6], [7].

To resolve conflicts among different task constraints in gradient projection, some researchers impose a hierarchy of motion by prioritizing constraints. The hierarchy is based on projecting the control of lower priority motion into the null space of higher priority motion. In [8], force and positioning tasks in joint and Cartesian space are combined with a hierarchical controller to enforce a strict task hierarchy. A multi-level hierarchy control structure was introduced in [9] by establishing general priorities among behavioral primitives. However, there can be violations of constraints so that certain objectives cannot be achieved.

Sampling-based methods [10], [11], [12] search robot configurations satisfying constraints, but are computationally more expensive, conducted offline, and may not find solutions. The manipulation planning framework presented in [12] allows robots to plan in the presence of constraints on end-effector pose as well as other constraints without prioritizing the constraints.

Configuration control methods are derived from the idea of minimizing a cost function[13]. Some recent work also adopts this idea to control redundant manipulators with multiple constraints[14], [15]. These methods essentially only provide approximate solutions and achieve a trade-off among tasks.

A novel approach of collision avoidance based on the Jacobian transpose method is proposed in [16], which does not require the calculation of the gradient of the minimum distance function, and hence it is a computationally simple algorithm. It is robust for multiple obstacle avoidance but suffers from poor convergence properties, which is the inherent problem of the Jacobian transpose method.

Quadratic programming methods have been reported to search for joint configurations (inverse kinematics) that maximize the distance between the manipulator and obstacles under some inequality constraints (such as constraints on joint limits) [17], [18], [19]. Dynamic equality constraints (about escape velocity), which are regarded as the collision-free criterion, are also found in literature [20]. However, these methods suffer from heavy computational costs.

B. The stereotype of obstacle avoidance

Obstacle avoidance is normally considered as a subtask while the motion of the end-effector, which is subject to task constraints, is the main task. The general approach is to neglect obstacle avoidance when the robot is far away from

* Liqin Zhu and Xiang Luo are with the Mechanical Engineering School, Southeast University, China. Zhuliqin110@126.com, luox@seu.edu.cn

+ Huitan Mao and Jing Xiao are with the Department of Computer Science, University of North Carolina at Charlotte, USA. {hmao4, xiao}@uncc.edu

the obstacle (using a threshold or an activation function) and take it into account when the robot is close to the obstacle. Note that when obstacle avoidance becomes active, it may be considered a primary task instead, i.e, the priority is reversed [14], [15], [21], [22], [23], [24], [25].

In the existing literature, to achieve obstacle avoidance is to move the robot away from obstacles instead of closer to obstacles. For instance, the potential-field based approaches only allow the forces from the obstacles to be repulsive[2], [13]. In the recent work of [21], the surroundings of an obstacle are modelled as springs. The closer the robot is to the obstacle, the more the springs will be compressed (creating repulsive forces), and therefore the larger the pseudo elastic energy is stored. This method also prevents the robot from getting closer to the obstacle by minimizing the energy. In [23], a repulsive force that increases the distance between the robot and the obstacle is constructed and used to reshape the joint velocity bounds to be tighter. As a result, joint motions that are contrary to the constraints are slowed down or denied. The approaches that solve obstacle avoidance by finding the minimum norm of joint velocities are also inherently constrained to deny any motion that moves the robot closer to obstacles since the corresponding joint velocities are commanded to be zero [14], [15].

However, when the robot end-effector executes a task-constrained motion, preventing the arm from getting closer to obstacles can miss the opportunity of achieving both obstacle avoidance and the task-constrained motion, resulting in the conflict between the two objectives.

Even though some existing methods could be potentially extended to make the robot arm move closer to obstacles, such as [25], [24], [16], [22], how to decide such motion seems non-trival and depends on environmental information.

C. The contribution of this paper

This paper introduces a novel method of *coordination of task motion and self motion* (CTS) to enable a redundant robot manipulator to get closer to obstacles if moving away from obstacles can conflict with the task-constrained motion of its end-effector, in order to find a solution that achieves both obstacle avoidance and task-constrained motion. The method is based on analysing and exploiting the effects of the two terms in the expression of gradient projection to effectively control the null-space motion with a proper projection matrix. If a collision-free manipulator trajectory exists for satisfying both obstacle avoidance and task-constrainted motion, the method will enable the manipulator to find it and execute it in real-time.

The paper is organized as follows. Section II describes task and self motions of redundant manipulators. Section III presents and discusses distance criteria between obstacles and the manipulator. In Section IV, the concepts of obstacle avoidance indicators are defined for task motion and self motion to explore their coordination in achieving both the task objectives and obstacle avoidance, and accordingly, a method to parameterize the projection matrix for self motion is introduced. In Section V, the algorithms of self motion generation and control strategy are presented. In Section VI, the effectiveness of the strategy is verified on example tasks using a planar 4 degrees of freedom (DOF) manipulator in simulation and a spatial 7-DOF manipulator in real experiments. Section VII presents the conclusions.

II. Task and Self Motion of Redundant Manipulators

A redundant manipulator with n joint DOFs in m dimensional workspace ($m < n$) satisfies $x = f(q)$, where $x \in R^m$ is the pose of the end-effector in Cartesian space, $q \in R^n$ is the joint angle vector, and $f(q) \in R^m$ is the forward kinematics of the manipulator. To fulfill the task requirements for the end-effector in the Cartesian space requires solving the non-linear inverse kinematics problem. It is common to differentiate x to obtain a linear system of equations

$$\dot{x} = J\dot{q}, \tag{1}$$

where $J \in R^{m\times n}$ is the manipulator Jacobian. According to [5], the inverse solution to (1) is

$$\dot{q} = G_1\dot{x} + (G_2J - I_n)Z, \tag{2}$$

where G_1 and G_2 are generalized inverse matrices of J, i.e. $JG_1J = J, JG_2J = J$. I_n is the $n \times n$ identity matrix, and $Z \in R^n$ is an arbitrary vector, which can be set as $Z = -k\nabla H$, where k is a real scalar, and ∇H is the gradient of a smooth function to be maximized. So the solution (2) can be rewritten as:

$$\dot{q} = G_1\dot{x} + k(I_n - G_2J)\nabla H, \tag{3}$$

The first part of (3) ($\dot{q}_h = G_1\dot{x}$) is the particular inverse solution of (1), which represents the joint motion mapped from the task trajectory to the end-effector and is called the *task motion*. The second part of (3) ($\dot{q}_s = k(I_n - G_2J)\nabla H$) is the homogeneous inverse solution of (1). Since $J\dot{q}_s = 0$, the motion of $\dot{q}_s$ only changes the manipulator configuration without affecting the task trajectory, and thus it is called *self motion*. In order to successfully execute a task trajectory, the manipulator has to avoid obstacles in the environment. So H can be a function related to the distance to obstacles.

III. Distance Criteria between Obstacles and Manipulator

We are interested in a distance function H to capture the relation between obstacles and a redundant manipulator. Let d_i be the minimum distance between link i of the manipulator and obstacles, as shown in Fig. 1.

Now we define $H^* = d_{min} = \min(d_i), i = 1, 2, ..., n$ to capture the minimum distance between obstacles and the manipulator. However, since the nearest link to an obstacle can change during motion, the gradient of the above H^* function will not be continuous, and thus the joint velocity in (3) will not be continuous. Therefore, we further define H as the weighted sum of distances between obstacles and each link:

$$H = \sum_{i=1}^{i=n} \beta_i d_i, i = 1, 2, ..., n, \tag{4}$$

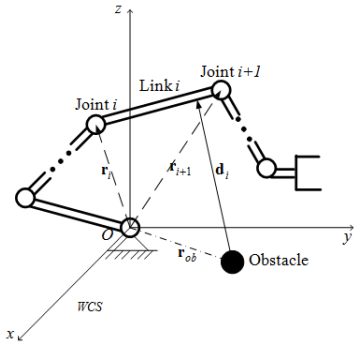

Fig. 1. The distance between an obstacle and link i.

where β_i is a weight:

$$\beta_i = \delta^{d_i - d_{min}}, 0 < \delta < 1. \quad (5)$$

The smaller a distance is, the greater its weight is. That is, the weight of the distance between obstacles and the nearest link is the largest. Now, H is a continuous function and so is its gradient. Therefore, we use H to generate self motion so that the joint motion is continuous. However, H^* is better at indicating the closest distance between the manipulator and obstacles compared to H.

We compute ∇H_i, the partial derivative of H with respect to the ith joint angle, by a discrete difference:

$$\nabla H_i = \frac{\partial H}{\partial q_i} = \frac{H(q_i + \partial q_i) - H(q_i)}{\partial q_i}, i = 1, 2, ..., n, \quad (6)$$

where the constant ∂q_i is obtained empirically. We further denote $\nabla H = [\nabla H_1, \nabla H_2, ..., \nabla H_n]^T$.

∇H^* is calculated similarly as ∇H.

IV. Coordination of Task and Self Motion

While it is a common practice to optimize the null-space motion of a redundant manipulator for obstacle avoidance during the execution of the task motion in existing methods of gradient projection, we notice that the manipulator can often better achieve both the task objective and obstacle avoidance if the self motion does not always optimize, i.e., does not always maximize a distance function between the manipulator and obstacles. There are cases that the distance needs to be decreased to avoid obstacles. Moreover, the task motion can often help obstacle avoidance.

Hence, we are interested in controlling self motion to coordinate with task motion so that both the task objective and obstacle avoidance can be achieved. We first define the concept of "obstacle avoidance indicator" for task motion and self motion respectively and next introduce a method to parameterize self motion in order to facilitate the control of self motion through the values of those parameters.

A. Obstacle Avoidance Indicators

As the change of H^* value during joint motion is more indicative of how the manipulator relates to obstacles than that of H, we define the *obstacle avoidance indicator of joint motion* as:

$$\Gamma = \frac{dH^*}{dt} = \frac{\partial H^*}{\partial q}\frac{dq}{dt} = \nabla H^* \cdot \dot{q} = \nabla H^* \cdot \dot{q_h} + \nabla H^* \cdot \dot{q_s}. \quad (7)$$

Note that since H^* can be viewed as a piecewise continuous function, Γ may not be continuous. However, we are only interested in the values of Γ (and not its derivative). We further define

$$\Gamma_h = \nabla H^* \dot{q_h}, \quad (8)$$

as the *obstacle avoidance indicator of task motion*, and

$$\Gamma_s = \nabla H^* \dot{q_s}, \quad (9)$$

as the *obstacle avoidance indicator of self motion*.

Self motion is not needed when $\Gamma_h \geq 0$, i.e., when the task motion can satisfy the task objective while not making the manipulator closer to any obstacle. On the other hand, self motion should be conducted when $\Gamma_h < 0$ and the manipulator is close to colliding with an obstacle.

When $\Gamma_h < 0$ and the value of H^* is lower than some threshold (i.e., the manipulator is very close to some obstacle), self motion is needed, and depending on the starting value of Γ_s when self motion is added, there can be two kinds of behaviors of self motion with respect to task motion:
(1) $\Gamma_s \geq 0$ **at the beginning of a self motion**. The effect of the self motion is to pull the manipulator away from obstacles while executing the task motion. However, sometimes this effect can be weaker than that of the task motion, which pulls the manipulator closer to obstacles (since $\Gamma_h < 0$). In such a case, the manipulator will be unable to avoid some obstacle if it continues its (combined task and self) motion. On the other hand, if the effect of self motion is stronger than that of the task motion, self motion can prevent the manipulator from continuing task motion while avoiding obstacles.
(2) $\Gamma_s < 0$ **at the beginning of a self motion**, i.e., both $\Gamma_h < 0$ and $\Gamma_s < 0$ hold when a self motion is added. If there exists a collision-free solution for the manipulator to execute the task motion, the values of Γ_s and Γ_h will increase over time and become positive, and as the result, the task objective and obstacle avoidance can be both achieved.

It is important to note that the behavior (1) above is the reason that existing gradient projection methods (as discussed in Section I) sometimes either cannot make the manipulator avoid obstacles or cannot make it continue task execution. Therefore, we are interested in adding a self motion with behavior (2) when behavior (1) cannot succeed in achieving both task objective and obstacle avoidance.

A key insight about the desired effect of self motion in behavior (2) is that Γ_h increases from initially being negative to become positive as Γ_s increases *as long as there exists a collision-free solution for the manipulator to execute the task motion*. By allowing the manipulator move closer to obstacles initially (with $\Gamma_s < 0$), the behavior of the self motion is equivalent to *adding search in the direction of the negative gradient* for such a collision-free solution (i.e., the collision-free manipulator motion that also fulfills the task objective) to exhaust the search scope. If such a solution

exists, it will be found, and Γ_h will increase its value from negative to positive over time. In other words, if search for a solution in the direction of the positive value of Γ_s, as the effect of behavior (1) above, fails, conducting search in the direction of the negative gradient, as shown in behavior (2), will find the solution if the solution exists.

We next introduce a method to parameterize self motion so that we can control self motion to be in the behavior (2) through changing some parameter values when needed.

B. Parameterization of Self Motion

Here we present a parameterization method to enable the control of self motion in order to obtain the desired self motion behavior. We adopt the weighted generalized inverse to express G_1 and G_2 in (3) as:

$$G_1 = N_1^{-1}J^T(JN_1^{-1}J^T)^{-1}, G_2 = N_2^{-1}J^T(JN_2^{-1}J^T)^{-1} \tag{10}$$

where N_1 and N_2 are diagonal matrices, and their diagonal elements are positive. We set N_1 as an identity matrix so that G_1 is Moore-Penrose generalized inverse of J and $\dot{q_h}$ is unique, which is known as the least norm solution.

We take N_2 as a matrix of parameters to control self motion. Let

$$N_2 = \begin{pmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_n \end{pmatrix} \tag{11}$$

We further denote the coefficient matrix of ∇H in (3) as Y:

$$\begin{aligned} Y = k(I_n - G_2J) &= \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \\ &= \begin{pmatrix} A_1 & A_2 & \dots & A_n \end{pmatrix} \end{aligned} \tag{12}$$

Y is a real symmetric matrix and its rank is $n-m$. Since the redundancy of the manipulator is $n-m$, $n-m$ parameters in N_2 can be tuned to change Y and consequently, the self motion $Y\nabla H$ in (3).

In particular, we consider the case $n-m=1$, i.e., there is one redundant degree of freedom to realize self motion, to show how N_2 can be used to control the self motion. In this case, the rank of Y is 1, and $A_2, A_3, ..., A_n$ are linear functions of A_1 as the following:

$$\begin{cases} A_1 = \sigma_1 A_1, \sigma_1 = 1 \\ w_1 A_2 = w_2 \sigma_2 A_1 \\ \quad \vdots \\ w_1 A_n = w_n \sigma_n A_1 \end{cases} \tag{13}$$

where $\sigma_1, \sigma_2, ..., \sigma_n$ are determined by the manipulator Jacobian J by setting N_2 to be an identity matrix, i.e., $w_1 = w_2 = ... = w_n = 1$, in the above equation (13).

The first row of equation (13) can be written as

$$a_{11} = a_{11}\sigma_1, a_{12} = \frac{w_2}{w_1}a_{11}\sigma_2, \dots, a_{1n} = \frac{w_n}{w_1}a_{11}\sigma_n, \tag{14}$$

and the first row of $Y\nabla H$ is:

$$Y\nabla H(1) = a_{11}\nabla H_1 + a_{12}\nabla H_2 + \dots + a_{1n}\nabla H_n. \tag{15}$$

Let (15) $= 0$, It follows that,

$$w_1\sigma_1\nabla H_1 + w_2\sigma_2\nabla H_2 + \dots + w_n\sigma_n\nabla H_n = 0. \tag{16}$$

Since we only need a single parameter to control the self motion with one redundant degree of freedom, we choose w_1 as the single parameter and let $w_2 = w_3 = \dots = w_n = 1$. [1] From (16), we can solve w_1 as a real number solution $w_1^{'}$, i.e., $w_1 = w_1^{'}$.

Since the common gradient projection has self motion always maximize the distance to obstacles, the obstacle avoidance indicator Γ_s of the self motion is positive. The common gradient projection has $w_1^{'} = 1$ (N_2 to be an identity matrix), so value of 1 is indicative and we can use $w_1^{'} = 1$ to locate the sign of Γ_s.

As shown in Fig. 2, we can set w_1 to be either greater or less than $w_1^{'}$ to control the behavior of self motion. If $w_1^{'} > 1$, $\Gamma_s > 0$ if we set $w_1 < w_1^{'}$, and $\Gamma_s < 0$ if we set $w_1 > w_1^{'}$. If $w_1^{'} < 1$, $\Gamma_s > 0$ if we set $w_1 > w_1^{'}$, and $\Gamma_s < 0$ if we set $w_1 < w_1^{'}$.

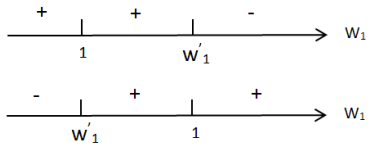

Fig. 2. Sign of the obstacle avoidance indicator Γ_s of self motion when the motion starts can be controlled by setting the value of parameter w_1 relative to the solution $w_1^{'}$ of equations (16).

Note that as we change w_1 to change N_2 in (3), the self motion remains in the null space of J, and the two parts on the right hand side of (3) remain mutually orthogonal. The parameterization of self motion introduced above allows us to decide which self motion to add based on the sign of Γ_s and its relation to Γ_h in our self motion generation and control strategy, which will be introduced next.

By using N_2 to adjust self motion, our approach renders the value of parameter k in (3) less important and can be set to a value from a large interval without affecting the performance of the manipulator, which will be demonstrated in the examples presented in Section VI.

V. Self Motion Generation and Control Strategy

We now describe how to generate self motion in different situations for effective coordination with the task motion. Self motion is needed only when task motion alone cannot avoid obstacles. When the minimum distance H^* is greater than or equal to a threshold, i.e., $H^* \geq H_{th}$, self motion is not necessary, and we make $\dot{q}_s = 0$, i.e., only task motion is adopted. When $H^* < H_{th}$ and $\Gamma_h < 0$, a self motion is

[1] Of course we can also use a different parameter, such as w_2, and set the other parameter values to be 1.

used to coordinate with task motion for obstacle avoidance. Our strategy first searches a self motion starting with $\Gamma_s > 0$, i.e., of behavior (1) (see Section IV.A), and if it fails to make the manipulator avoid obstacles, our strategy next generates a self motion starting with $\Gamma_s < 0$, i.e., of behavior (2). The threshold H_{th} is set large enough so that the manipulator will not collide with obstacles even when $\Gamma_s < 0$.

Let $[0, T]$ be the duration of the task trajectory. We use **Algorithm 1** and **Algorithm 2** to decide a proper self motion whenever needed by setting a proper value for w_1, and then compute and execute the corresponding self motion $\dot{q}_s(t)$ and task motion $\dot{q}_h(t)$ using equation (3). In **Algorithm 2**, $\Delta < T$ is an upper bound for a period of self motion, which can be decided based on the threshold H_{th} and the velocity bounds from the task trajectory. The resulting motions both satisfy task constraints and avoid obstacles.

Algorithm 1: *ControlStrategy*

$t = 0$; $\dot{q}_s(t) = 0$ for all $t \leq T$;
while $t \leq T$ **do**
  if $H^*(t) < H_{th}$ *and* $\Gamma_h(t) < 0$ **then**
    solve for w_1', and choose a value of w_1 based on w_1' to make $\Gamma_s(t) > 0$ and compute $\dot{q}_s(t)$;
    call **Algorithm 2** and get $collisionFlag$;
    if $collisionFlag = true$ **then**
      choose a value of w_1 based on w_1' to make $\Gamma_s(t) < 0$;
      call **Algorithm 2** and get $collisionFlag$;
      if $collisionFlag = true$ **then**
        return "no collision-free solution that satisfies both obstacle avoidance and task motion"
      end
    end
    simultaneously execute $\dot{q}(u) = \dot{q}_h(u) + \dot{q}_s(u) \forall u \in [t, t + \tau]$ and continue the while loop;
    $t = t + \tau + 1$;
  end
  else
    $t = t + 1$;
    execute $\dot{q}(t) = \dot{q}_h(t)$ after $\dot{q}(t-1)$ is done
  end
end

Algorithm 2: *GenerateSelfMotion*

begin
  $u = t$;
  while $H^*(u) \leq H_{th}$ *and* $\Gamma_h(u) \leq 0$ *and* $u \leq t + \Delta$ **do**
    if *the combined task and self motion* $\dot{q}_h(u) + \dot{q}_s(u)$ *does not cause collision* **then**
      save $\dot{q}_h(u)$ and $\dot{q}_s(u)$;
      $u = u + 1$
    end
    else
      return $collisionFlag = True$
    end
  end
  $\tau = u - t$; /* τ is the actual period of self motion. */
  return $collisionFlag = False$ and τ
end

VI. Implementation and Testing

We have implemented our coordination of task and self motion (CTS) approach both in simulation and in real experiments and verified its effectiveness.

A. Testing with a simulated planar manipulator

We have used a simulated planar 4-DOF manipulator as an example, shown in Fig. 3. The length of each link is 2 units (1 unit = 25 cm) in simulation. The position of end-effector is (x_1, x_2). The orientation of end-effector is the angle x_3 between the end-effector and the x_1 axis.

We consider a linear trajectory for the end-effector in the Cartesian space, as shown in Fig. 3. The initial pose of the end-effector is $(0, 0, 3\pi/2)$. The final desired pose of the end-effector is $(5, 5, \pi/4)$. We set 500 time steps in the simulation. The initial manipulator configuration is $q_1 = 0, q_2 = q_3 = q_4 = \pi/2$. A point obstacle is set at $(3, 1)$. This is a case that existing methods (solely increasing the distance to obstacle) cannot achieve both task objective and obstacle avoidance. With our CTS approach, the manipulator can achieve both objectives simultaneously with a self motion of behavior (2), as detailed below.

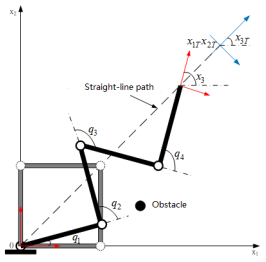

Fig. 3. A planar 4-DOF manipulator, its initial configuration, an intermediate configuration, the straight-line path for the end-effector and an obstacle.

We set $k = 0.02$ and $H_{th} = 0.5unit$. We found that the manipulator task performance is not sensitive to k so that it is relatively easy to determine a k value.

With CTS approach, the execution of the end-effector trajectory consists of 4 processes shown in Table I. In process 1, minimum distance H^* is greater than the threshold (shown as the black line in Fig. 4), so only task motion is conducted (shown in Fig. 5). H^* decreases with time, and it becomes less than the threshold from $t = 32$ steps, at which a self motion is added, and the manipulator enters process 2.

To determine the self motion, our algorithm follows the derivation in Section IV.B to get $w_1' = 1.2857$. By first

selecting a $w_1 < w_1'$, a self motion of behavior (1) is obtained and the distance between the manipulator and current nearest linke (link 2) is trying to be maximized. This cannot lead to obstacle avoidance as evidenced in the attached video. Therefore, our algorithm next sets $w_1 = 2$ in process 2 to generate a self motion of behavior (2). As shown in Fig. 5, the obstacle avoidance indicator Γ_s starts being negative (see the local amplification in Fig. 5) and increases to be positive over time. As observed in Fig. 4, moving purposefully closer to the obstacle changes the closest link from link 2 to link 3. This change of the nearest link is the opportunity gained by moving the robot closer to the obstacle in the short-term, and it is the key factor that enables achieving both the task objective and obstacle avoidance simultaneously.

TABLE I

PARAMETERS OF MOTION PROCESS

Process	Time step	Motion	N_2 Diagonal Entry
(1)	0-31	Task motion only	
(2)	32-60	Task and self motion	2, 1, 1, 1
(3)	61-266	Task and self motion	2, 1, 1, 1
(4)	266-500	Task motion only	

As the nearest link changes from link 2 to link 3, process 3 is entered, which differs from process (2) only in practical implementation of our algorithm, in that the analytical expression for H^* changes as the nearest link changes. In process 3, the self motion (which has $\Gamma_s > 0$) reduces the negative effect of task motion (which has $\Gamma_h < 0$), as shown in Fig. 5. In process 4, $\Gamma_h > 0$, and thus only the task

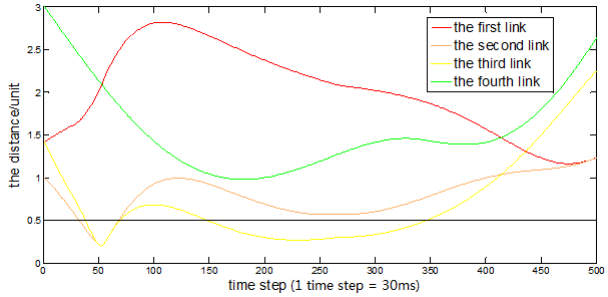

Fig. 4. Distances between links and the obstacle.

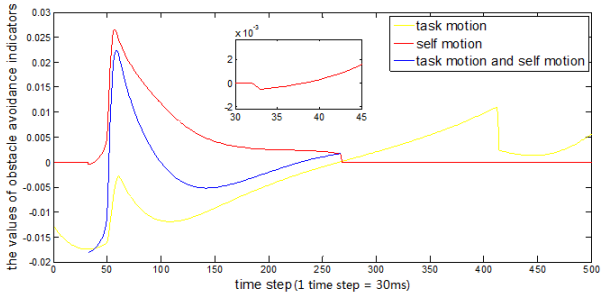

Fig. 5. Obstacle avoidance indicator as a function of time.

motion is conducted until the task is completed at $t = 500$.

B. Testing with a spatial manipulator both in simulation and real experiments

A spatial 7-DOF Barrett WAM redundant manipulator is also used to demonstrate the effectiveness of our CTS method. The manipulator starts at a given configuration having the end-effector at $(0.5m, 0.1m, 0.5m)$ and aligned with the Z axis. The task is to move the end-effector along a straight line from the starting configuration to $(0.5m, -0.5m, 0.5m)$ while maintaining the end-effector orientation, i.e., both the position and the orientation of the end-effector are constrained. Here we set $k = 0.01$ (k in the interval [0.01, 0.03] presents desired motion behavior). The test cases (simulation and real experiments) are presented in the following section and also in the attached video.

1) Avoiding one obstacle: We first consider the case where the manipulator needs to avoid a cylindrical pole aligned with the Z direction during the execution of the end-effector task motion. The bottom of the pole is located at $(0.35m, 0.0m)$, the height of the pole is $0.7m$, and the diameter is $0.02m$. We set $H_{th} = 0.1m$. With our proposed CTS method, when $H^* < H_{th}$ and $\Gamma_h < 0$, a self motion of behavior (1), i.e., starting in the positive direction of Γ_s is first conducted, but it fails to enable the manipulator to avoid the obstacle. Next, a self motion of behavior (2), i.e., starting in the negative direction of Γ_s is obtained. Moving the arm purposefully closer to the obstacle enables the arm to bypass the top of the obstacle, after which the arm can move even further away from the obstacle by solely increasing the distance. Fig. 6 shows snapshots of the task-constrained collision-free trajectory obtained. All the joint angles are mostly smooth within the limits, as shown in Fig. 7.

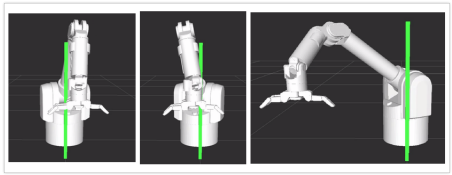

Fig. 6. A task-constrained collision-free trajectory.

In this example, the self motion is not turned on until $H^* < H_{th}$ at time step 350, when a self motion of behavior (2) is added. As shown in Fig. 8, Γ_s starts being negative and increases to positive values, which moves the manipulator closer first and then further away from the pole. The arrow marked in Fig. 9 also indicates that the distance is being decreased with a larger slope at that time step under the effect of self motion with behavior (2). The self motion is turned off when Γ_h becomes positive at time step $1,080$. Between time steps $1,080$ and $6,000$, only task motion is executed, which moves the manipulator away from the obstacle.

2) Comparative study: We compared our approach with existing gradient projection and configuration control method [5], [13] using the one-pole example described above, and the results are shown in the attached video. The existing method relies on increasing the distance to obstacles for obstacle

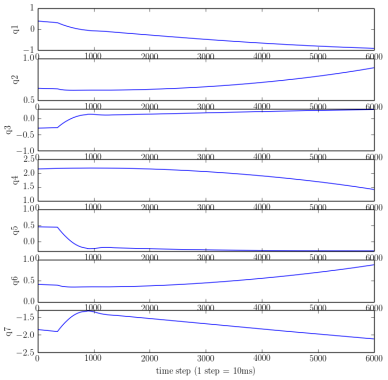

Fig. 7. Joint angles (radian) in the 1-pole case.

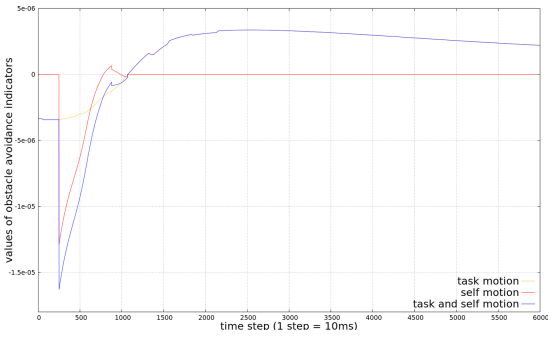

Fig. 8. Obstacle avoidance indicator in the 1-pole case.

avoidance. In the first comparison, the main task has the top priority. As can be seen in the video, the existing method requires the self motion to move away from the obstacle and results in a conflict with the main task. In the second comparison, the priority is switched, i,e, obstacle avoidance has the priority over the main task. As shown in the video, the existing method (solely increasing the distance) requires the arm to retract itself to avoid the obstacle, which leads to the detour of the end-effector and violation of task constraints.

3) Avoiding two obstacles: In this example, the manipulator needs to execute an end-effector straight-line trajectory (from $(0.35m, 0.25m, 0.5m)$ to $(0.35m, -0.4m, 0.5m)$ along Y-axis) and avoid two poles. The bottoms of the two poles are located at $(0.25m, 0.1m)$ and $(0.25m, -0.1m)$ respectively. They both have height $0.7m$. H_{th} is $0.05m$.

Fig. 10 shows snapshots of a task-constrained collision-free trajectory obtained in this case with our CTS method. The manipulator starts executing the task motion with self motion off until $H^* < H_{th}$. The manipulator enters the threshold two times (at time steps $1,140$ and $4,000$) in this case. Based on the negative sign of Γ_h, there are two self motions of behavior (2) (i.e., starting with $\Gamma_s < 0$) added at time step $1,140$ and $4,000$ (see Fig. 11 and the local amplification figure). The self motions enable the manipulator to move across the top of the obstacle to achieve the objective of obstacle avoidance while executing the end-effector task trajectory.

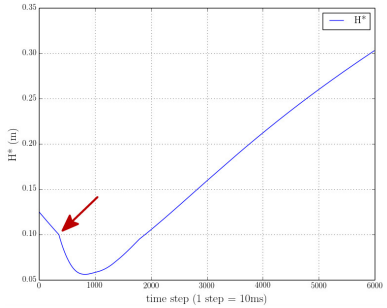

Fig. 9. H* in the 1-pole case.

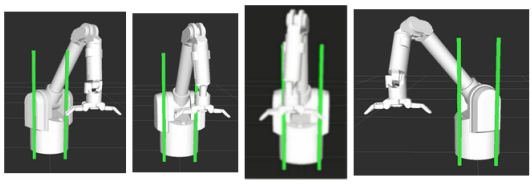

Fig. 10. A task-constrained collision-free trajectory in the 2-pole case.

Note that in time steps between $1,140$ and $4,000$, i.e., when the manipulator is in the middle of the two poles, there will be a local minimum if the conventional gradient projection method is used since the distance between the manipulator and the left pole decreases, the distance to the right pole increases as the manipulator tries to execute the task motion. Using the CTS method eliminates this local minimum, even if the distance between the poles is less than $2H_{th}$, i.e., $H^* < H_{th}$ remains true when the manipulator is in between the two poles. This is because during the period, self motion is not always on, but turned on and off based on the sign change of Γ_h, and each time self motion is needed again after being off, a new self motion is generated to ensure collision-free task-constrained movement.

The same tests in simulation are also executed in real experiments with position control. Fig. 12 shows a snapshop of the real Barrett WAM used in the 1-pole and 2-pole cases.

The attached video shows the simulations with both the 4-link planar manipulator and the Barrett WAM and the executions with an actual Barrett WAM. In all cases, the control strategy based on our CTS approach worked well, and the manipulator successfully avoided obstacles while executing the end-effector task trajectories without interruption.

VII. Conclusions

In this paper, we have introduced a novel method, the CTS method, to coordinate quasi-static task motion and self motion for redundant manipulators to enable the manipulator achieve both the task objective and obstacle avoidance simultaneously. We have defined the concepts of obstacle avoidance indicators for task and self motions at any time t and subsequently described a way to change the weight matrix of generalized inverse to obtain the kind of self motion when needed for obstacle avoidance. A planar 4-DOF manipulator and a spatial 7-DOF manipulator are

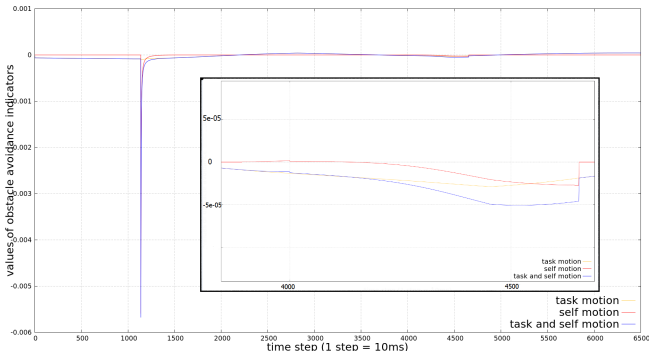

Fig. 11. Obstacle avoidance indicator in the 2-pole case.

(a) Execution in the 1-pole case

(b) Execution in the 2-pole case

Fig. 12. Executions of planned trajectories.

used in simulation and real experiments to demonstrate the effectiveness of the approach.

The CTS method offers several advantages. First, the method is more effective for obstacle avoidance while accomplishing the task objective than existing methods, as it both exploits the effect of obstacle avoidance of task motion and expands the search for a suitable self motion by allowing the manipulator to move closer to obstacles. Second, it only involves simple computation for each desired self motion and is thus efficient. Third, the method is insensitive to the value of the scalar k, and thus k can be easily chosen as a constant in a considerably large interval without affecting the performance of the method.

However, further study is necessary to ensure smooth transition when a self motion is added or removed. Our empirical tests show that selecting the value of w_1 to be close to w_1' increases the smoothness of transition. More investigation is also needed for cases with higher DOF redundancy, i.e., higher-dimensional null-space motion.

ACKNOWLEDGMENTS

This work is partly supported by the US National Science Foundation grant IIS-1439695.

REFERENCES

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research(IJRR)*, 1986.
[2] A. Muller, "Collision avoiding continuation method for the inverse kinematics of redundant manipulators," in *IEEE Int. Conf. on Robotics and Automation(ICRA)*, 2004.
[3] H. Reimann, I. Iossifidis, and G. Schöner, "Generating collision free reaching movements for redundant manipulators using dynamical systems," in *IEEE Int. Conf. on Intelligent Robots and Systems(IROS)*, 2010.
[4] O. Brock, O. Khatib, and S. Viji, "Task-consistent obstacle avoidance and motion behavior for mobile manipulation," in *IEEE ICRA*, 2002.
[5] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE transactions on systems, man, and cybernetics (SMC)*, 7(12):868-871, 1977.
[6] L. Li, W. Gruver, Q. Zhang, Z. Yang, *et al.*, "Kinematic control of redundant robots and the motion optimizability measure," *IEEE SMC, Part B: Cybernetics*, 31(1):155-160, 2001.
[7] Y. Liu, J. Zhao, and B. Xie, "Obstacle avoidance for redundant manipulators based on a novel gradient projection method with a functional scalar," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1704–1709, 2010.
[8] E. Lutscher and G. Cheng, "A practical approach to generalized hierarchical task specification for indirect force controlled robots," in *IEEE IROS*, pp. 1854–1859, 2013.
[9] L. Sentis and O. Khatib, "Synthesis of whole-body behaviors through hierarchical control of behavioral primitives," *International Journal of Humanoid Robotics*, 2(4):505-518, 2005.
[10] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Trans. Robot(TRO)*, 26(3):576-584, 2010.
[11] Z. Yao and K. Gupta, "Path planning with general end-effector constraints," *Robotics and Autonomous Systems*, 55(4):316-327, 2007.
[12] D. Berenson, S. S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *IJRR*, 30(12):1435-1460, 2011.
[13] F. Fahimi, *Autonomous robots: modeling, path planning, and control*, vol. 107. Springer Science & Business Media, 2008.
[14] J. Xiang, C. Zhong, and W. Wei, "A varied weights method for the kinematic control of redundant manipulators with multiple constraints," *IEEE TRO*, vol. 28, no. 2, pp. 330–340, 2012.
[15] J. Xiang, C. Zhong, and W. Wei, "General-weighted least-norm control for redundant manipulators," *IEEE TRO*, vol. 26, no. 4, 2010.
[16] K.-K. Lee, Y. Komoguchi, and M. Buss, "Multiple obstacles avoidance for kinematically redundant manipulators using jacobian transpose method," in *SICE Annual Conference*, pp. 1070–1076, 2007.
[17] Y. Zhang, Z. Li, and H.-Z. Tan, "Inequality-based manipulator-obstacle avoidance using the lvi-based primal-dual neural network," in *IEEE ROBIO*, pp. 1459–1464, 2006.
[18] Y. Zhang and J. Wang, "Obstacle avoidance of redundant manipulators using a dual neural network," in *IEEE ICRA*, vol. 2, 2003.
[19] D. Guo and Y. Zhang, "A new inequality-based obstacle-avoidance mvn scheme and its application to redundant robot manipulators," *IEEE SMC, Part C: Applications and Reviews*, 42(6):1326-1340, 2012.
[20] F.-T. Cheng, T.-H. Chen, Y.-S. Wang, and Y.-Y. Sun, "Obstacle avoidance for redundant manipulators using the compact qp method," in *IEEE ICRA*, pp. 262–269, 1993.
[21] P. Falco and C. Natale, "Low-level flexible planning for mobile manipulators: a distributed perception approach," *Advanced Robotics*, vol. 28, no. 21, pp. 1431–1444, 2014.
[22] K.-K. Lee and M. Buss, "Redundancy resolution with multiple criteria," in *IEEE IROS*, 2006.
[23] F. Flacco, A. De Luca, and O. Khatib, "Motion control of redundant robots under joint constraints: Saturation in the null space," in *IEEE ICRA*, pp. 285–292, 2012.
[24] T. Petrič and L. Žlajpah, "Smooth continuous transition between tasks on a kinematic control level: Obstacle avoidance as a control problem," *Robotics and Autonomous Systems*, vol. 61, no. 9, pp. 948–959, 2013.
[25] S.-i. An and D. Lee, "Prioritized inverse kinematics with multiple task definitions," in *IEEE ICRA*, 2015.