Task Constrained Motion Planning in Robot Joint Space

Mike Stilman
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
robot@cmu.edu
CMU-RI-TR-06-43

***Abstract*— We explore randomized joint space path planning for articulated robots that are subject to task space constraints. This paper presents a representation of constrained motion for joint space planners and develops two simple and efficient methods for constrained sampling of joint configurations: Tangent Space Sampling (TS) and First-Order Retraction (FR). Constrained joint space planning is important for many real world problems involving redundant manipulators. On the one hand, tasks are designated in work space coordinates: rotating doors about fixed axes, sliding drawers along fixed trajectories or holding objects level during transport. On the other, joint space planning gives alternative paths that use redundant degrees of freedom to avoid obstacles or satisfy additional goals while performing a task. In simulation, we demonstrate that our methods are faster and significantly more invariant to problem/algorithm parameters than existing techniques.**

I. Introduction

In this paper we explore the application of randomized motion planning algorithms to problems where the robot path is required to obey workspace constraints. Task compliance or constrained motion is critical in situations where the robot comes in contact with constrained objects. Many real-world tasks ranging from opening doors and pushing carts to helping align beams for construction and removing piled up rubble exhibit workspace constraints. In these circumstances, the robot must not only preserve the task constraint, but also avoid collisions and joint limits throughout a planned motion. Redundant robots, such as mobile manipulators and humanoids, have the dexterity to accomplish both objectives. The challenge is to efficiently explore alternative joint space paths that satisfy the task constraints without being trapped in local minima.

To address constrained joint space planning, we first present some background for our study and identify the key characteristics of the problem and the proposed solution. Sections IV and V develop a simple and general representation of constraints based on [1] and [2]. This representation is shown to be well suited for a wide range of problems in motion planning. Sections VI and VIII describe three joint space sampling techniques that preserve any constraint expressed with our representation. Section IX experimentally evaluates the three techniques in simulation. The comparison includes different tasks, choices for time step and error tolerance. Finally, the paper discusses extensions to our proposed tools and directions for future research.

II. Related Work

In addition to finding a collision-free joint space path [3], our problem requires that each configuration along the path satisfy workspace constraints for the end-effector [1],[2]. We distinguish this problem from specifying a single goal pose for the end-effector [4] or a path that the end-effector must follow [5], [6], [7]. In our case, the end-effector path is not predetermined and the constraints must be satisfied throughout the path.

Even when the end-effector path is specified, handling robot redundancy poses a difficult challenge. Typically, redundancy resolution is performed with local [8], [9] or global [10] techniques for optimal control. These methods optimize configuration-dependent criteria such as manipulability. Obstacle distance, a common criterion for collision avoidance, has a highly non-linear relationship to joint configurations and leads to the use of local optimization [11], [12], [13]. These methods require the generation of distance/potential functions for obstacles. Moreover, they are susceptible to local minima in the search space.

For a more comprehensive exploration of the search space, motion planning research has headed towards feasible solutions and probabilistically complete algorithms such as PRM[14] and RRT [15][16]. These algorithms operate by generating random joint space displacements. However, in the case of constrained motion, the probability of randomly choosing configurations that satisfy the constraints is not significant or zero. This is shown for problems involving closed chains in [17] and [18].

To address this challenge, some approaches use domain-specific attributes such as closed chain structure [17], [18], [19] or dynamic filtering [20]. Among these techniques, one algorithm, Randomized Gradient Descent (RGD)[17] has been extended for use with arbitrary constraints [21]. RGD randomly shifts sampled configurations in directions that minimize constraint error. However, [21] applies this algorithm with general constraints on a case-by-case basis. We propose to adapt the existing task frame formalism [1] to unify the representation of constraints and initiate the comparison of algorithms for constrained sampling.

In addition to RGD, we considered ATACE [21] as well as adapting the path-following strategies from [6], [7]. However, ATACE explores task space motions with RRTs and locally

follows them in joint space. This approach is incomplete when more than one joint space path must be explored along a single task space path (Figure 7). [6] and [7] require a partition of the robot into "base" and "redundant" joints. Error due to perturbations of redundant joints is compensated by inverse kinematics of the base joints. This algorithm relies on significant freedom for base joints, since obstacle and joint limit constraints will prevent error compensation even when redundancy in the remaining joints is available.

This paper presents two contributions in the domain of constrained joint space planning. First of all, we develop an intuitive formulation of task space constraints that encompasses a wide range of common robot goals. Secondly, we propose and evaluate a strategy for generating joint space samples that satisfy these constraints. Our strategy is shown to be experimentally successful in comparison to RGD with regard to different tasks and variations in parameter choices.

III. Preliminaries

To facilitate our discussion of task constraints we will employ three spaces of coordinates: joint space, work space and task space:

- $\mathbf{q}_i$ Joint space coordinates refer to a vector of single-axis translations and rotations of the robot joints.
- $\mathbf{T}_B^A$ Work space homogeneous matrices represent the rigid transformation of frame $\mathcal{F}^B$ with respect to frame $\mathcal{F}^A$.
- $\mathbf{x}_i$ Task space coordinates will be used for computing error with respect to the task frame (Section IV).

Note that task space is equivalent to work space by rigid transformation. We use a distinct set of coordinates to promote a simple and versatile formulation of constraints. We will also employ $\mathbf{R}_B^A$ for rotations of $\mathcal{F}^B$ with respect to $\mathcal{F}^A$. Likewise, $\mathbf{p}^A$ and $\mathbf{z}^A$ are vectors in frame A.

IV. Representation of Constraints

We define a task constraint as a restriction on the freedom of motion of a robot end-effector. The degrees of freedom for a rigid body are defined by translations and rotations in a given coordinate frame. The task frame, $\mathcal{F}^t$, is derived from the world frame, $\mathcal{F}^0$, by a rigid transformation of the world axes. The matrix $\mathbf{T}_t^0$ specifies the position and orientation of $\mathcal{F}^t$ with respect to the world frame.

In the task frame we have various options for quantifying end-effector error. For instance, translations may be in Cartesian or Spherical coordinates, while rotations could be described by Euler Angles or Quaternions. The choice of coordinates will affect the types of constraints that we can represent. [2] For any representation we define $\mathbb{C}$, the *motion constraint vector*. $\mathbb{C}$ is a vector of binary values for each of the coordinates. A value of one indicates that the end-effector motion may not change the coordinate.

In many cases it is advantageous to represent $\mathbb{C}$ as a diagonal matrix:

$$\mathbb{C} = \begin{bmatrix} c_1 \\ \cdots \\ c_n \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} c_1 & & \\ & \cdots & \\ & & c_n \end{bmatrix} \tag{1}$$

For instance, Eq. 2 restricts task space velocity $\dot{\mathbf{x}} = [\, \dot{x}_1 \; \dot{x}_2 \; \ldots \; \dot{x}_n \,]^T$ to a linear subspace of the coordinates.[1]

$$\mathbf{C}\dot{\mathbf{x}} = \mathbf{0} \tag{2}$$

We use $\mathbb{C}$ and $\mathbf{C}$ interchangeably. Without loss of generality we also highlight the example of a Cartesian representation of translations and roll/pitch/yaw rotations about fixed axes. In these coordinates, a rotation with respect to the task frame is defined about the x_t, y_t and z_t axes of $\mathcal{F}^t$:

$$\mathbf{R}_B^t = R(z_t, \phi)R(y_t, \theta)R(x_t, \psi) \tag{3}$$

This choice of coordinates facilitates intuitive definitions for many common workspace constraints. We directly specify permitted translations and rotations about the task frame axes. The $\mathbb{C}$ vector for our representation is six dimensional:

$$\mathbb{C}_{RPY} = [\; c_x \quad c_y \quad c_z \quad c_\psi \quad c_\theta \quad c_\phi \;]^T \tag{4}$$

In a sample application, setting all the elements except c_x and c_y to zero will free the end-effector to move in all directions except rotation about the x and y-axes of F_t.

The complete representation of a task constraint consists of the task frame $\mathcal{F}_t$, the coordinate system and the motion constraint vector $\mathbb{C}_t$. The description may be constant throughout the motion plan, or it could vary in accordance with the robot configuration or the state of a higher-level planner.

V. Specifying Constraints

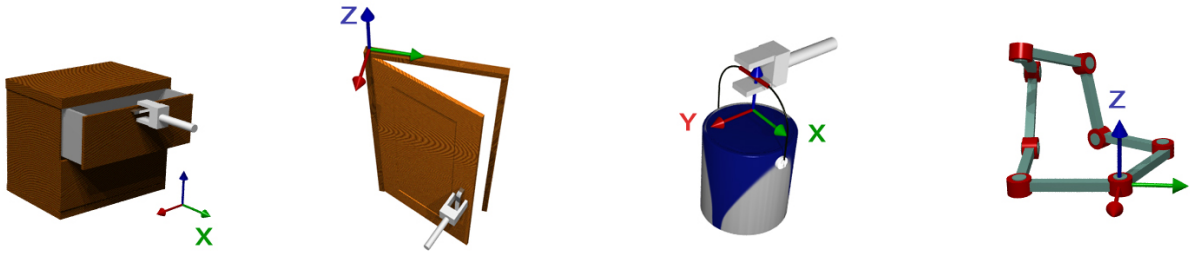
Section IV introduced a simple formulation of motion constraints that allows for significant variation in definitions. When defining a task constraint we can choose any frame transform, T_t^0, any permutation of $\mathbb{C}_t$ and any task coordinate system. Furthermore, we can parameterize these choices as functions of the planner state.

We now present four common constraints in robotics as task frame constraints for path planning. Our focus is to show that a spectrum of constraint definitions, from simple to complex, have intuitive formulations in our representation. Other forms of constraints can be modeled by modifying or combining the ones we describe.

A. Fixed Frames

The simplest task defines a single frame $\mathcal{F}_t$ and $\mathbb{C}_t$ for the entire path plan. Fixed frame constraints commonly occur when manipulating objects that are kinematically linked to the environment. Figures 6 and 7 show two examples where linkages are hinges and rails. $\mathcal{F}^t$ is chosen as any frame in which the axes align with the directions of constrained motion. Usually the transformation describing $\mathcal{F}^t$, $\mathbf{T}_t^0$ is simply the position and orientation of the object in the world frame. $\mathbb{C}_t$ indicates which axes of $\mathcal{F}_t$ permit valid translations and rotations.

When planning constrained manipulation, one can assume that the object is rigidly attached to the robot during grasp. At the time of grasp, let $\mathbf{T}_A^0$ and $\mathbf{T}_e^0$ represent the position



(a) Fixed $\mathbb{C} = [\,0\;1\;1\;1\;1\;1\,]^T$ (b) Fixed $\mathbb{C} = [\,1\;1\;1\;1\;1\;0\,]^T$ (c) Para. $\mathbb{C} = [\,0\;0\;0\;1\;1\;0\,]^T$ (d) Para. $\mathbb{C} = [\,1\;1\;1\;1\;1\;0\,]^T$

Fig. 1. Examples of constraints implemented with a roll/pitch/yaw specification of task coordinates. The task frames in (c) and (d) are parameterized by the configuration of the robot.

and orientation of object A and the end-effector respectively. We define the grasp transform $\mathbf{T_G}$:

$$\mathbf{T_G} = \mathbf{T}^e_A = \mathbf{T}^e_0 \mathbf{T}^0_A = (\mathbf{T}^0_e)^{-1} \mathbf{T}^0_A \tag{5}$$

Subsequently, during manipulation, the kinematics of the end-effector are extended to reflect the work space position of the manipulated object:

$$\mathbf{T}^0_{e'}(\mathbf{q}) = \mathbf{T}^0_e(\mathbf{q}) \mathbf{T_G} \tag{6}$$

With this kinematic extension, $\mathbb{C}$ directly specifies the permitted displacements of the end-effector frame with respect to the task frame.

B. Simple Frame Parameters

Section V-A described a global constraint on the robot end-effector. Suppose the constraint is defined locally with respect to the position of the end-effector or another function of the planner state. Section V-D addresses complex continuous definitions of constraints. However, for common tasks a simple parametrization of the task frame is sufficient.

For instance, a task may consist of manipulating a sequence of constrained objects. Each object is assigned a distinct task frame and constraint vector. The path planner selects the task frame based on the object or subspace with which it is in contact.

Alternatively, the constraint for a single object may be defined locally with respect to the configuration of the robot. For example, a local constraint on rotation is meaningful for transporting open containers of liquid such as paint (Figure 8). In this case, the task frame orientation is designated by the world frame and the position is determined by the end-effector configuration. The constraint vector remains a specification of the directions of permissible motion.

C. Kinematic Closure Constraints

An important constraint for multi-arm manipulation and reconfigurable robots is a linkage with a closed kinematic chain. One approach to specifying kinematic closure is to break the chain at an arbitrary joint $\mathbf{j}_k$. The linkage becomes an open chain with a constraint defined by the freedoms of the detached joint $\mathbf{j}_k$.

We formulate closure as a special case of parameterized task frame constraints. For any open chain joint configuration, $\mathbf{q}$, the kinematic expression for $\mathbf{j}_k$ along each chain yields a work space transform $\mathbf{T}(\mathbf{q})$ describing $\mathbf{j}_k$ with respect to $\mathcal{F}^0$. One of these transforms can be specified as the task frame, $\mathbf{T}^0_t(\mathbf{q})$ while the others are end-effectors $\mathbf{T}^0_e(\mathbf{q})$. As with previous examples, the constraint vector intuitively specifies the degrees of freedom for the joint $\mathbf{j}_k$.

D. Constraint Paths and Surfaces

Some elaborate constraints may require the end-effector to remain on a non-linear path or maintain contact with a complex surface. In this case there are at least three options:

1) Parameterize the task frame by the nearest position on the path or surface to the sampled configuration.
2) Parameterize the surface by a subset of standard orthogonal coordinates and constrain the remaining coordinates to the surface values.
3) Define a coordinate system that implicitly encodes distance from the surface as a change to the coordinates.

The first option may require global or local optimization to determine the nearest surface point. The second avoids this computation by deciding the values for the parameter coordinates as a function of the sampled configuration. However, this approach may not always be possible and it would also lead to an asymmetric exploration of the constraint manifold. The final alternative could prove difficult to represent globally, yet it may be implemented locally with numerical regression techniques [22].

Notice that in the case where a path is parameterized by time, the end-effector trajectory is a continuous sequence of task frame parameters. Extending the search space with time, the frame of a sampled configuration is decided by time.

VI. Introducing Constrained Sampling

In selecting a representation of task constraints we identified a linear subspace of constrained configurations for the end-effector in the task space. This is the space of all coordinates x in $\mathcal{F}^t$ such that $x_i = 0$ when $c_i = 1$. This space may map to a non-linear manifold in robot joint space. When sampling the joint space, randomized planners will typically produce samples that lie outside the constraint manifold. Our proposed methods use $\mathbb{C}$ to formulate a distance metric in task space and either select or project samples within a tolerance distance (ϵ) of the constraint manifold.

A. Computing Distance

Having identified a sampled configuration $\mathbf{q}_s$, we compute the forward kinematics for $\mathbf{q}_s$. Commonly the end-effector frame, $\mathcal{F}^e$, is found in homogeneous coordinates as the transformation $\mathbf{T}_e^0(\mathbf{q}_s)$. The displacement of $\mathcal{F}^e$ with respect to the task frame $\mathcal{F}^t$ is found by:

$$\mathbf{T}_e^t(\mathbf{q}_s) = \mathbf{T}_0^t \mathbf{T}_e^0(\mathbf{q}_s) = (\mathbf{T}_t^0)^{-1} \mathbf{T}_e^0(\mathbf{q}_s) \tag{7}$$

We now represent the transform for end-effector displacement with respect to the task frame in task coordinates. The Appendix also defines this change in representation for roll/pitch/yaw.

$$\Delta\mathbf{x} \equiv \mathbf{T}_e^t(\mathbf{q}_s) \tag{8}$$

The task space error is found simply by the product in Eq. 9. This product has the effect of selection presented in Eq. 10.

$$\Delta\mathbf{x}_{err} = \begin{bmatrix} e_1 & \dots & e_n \end{bmatrix} = \mathbf{C}\boldsymbol{\Delta}\mathbf{x} \tag{9}$$

$$e_i = \begin{cases} 0 & c_i = 0 \\ \Delta\mathbf{x}_i & c_i = 1 \end{cases} \tag{10}$$

B. Baseline: Randomized Gradient Descent

The proposed distance metric detects when a configuration satisfies the constraint tolerance. Furthermore, it can be used to identify task space motions that reduce error. Our algorithms use this information to construct constrained samples.

The three methods we compare are Randomized Gradient Descent (RGD) [17][21], Tangent-Space Sapmling (TS), and First-Order Retraction (FR). For simplicity, we will describe each approach as a modification to the basic RRT algorithm summarized in Figure 2. The algorithm samples a random configuration $\mathbf{q}_{rand}$ and finds its nearest neighbor in the tree $\mathbf{q}_{near}$. The sampled configuration $\mathbf{q}_s$ is placed at a fixed distance Δt along the vector from $\mathbf{q}_{near}$ to $\mathbf{q}_{rand}$.

As a baseline, let us consider the RGD algorithm detailed in Figure 3. After computing the task space error of $\mathbf{q}_s$, this method continues to uniformly sample the neighborhood of the configuration within a radius d_{max}. The error of each new configuration $\mathbf{q}'_s$ is evaluated and compared with $\mathbf{q}_s$. If the error is less, $\mathbf{q}'_s$ is substituted for $\mathbf{q}_s$. The procedure terminates when the error is less than ϵ or the maximum iteration count is met.

VII. Relating Joint Motion to Constraint Error

Due to random selection, the RGD algorithm will typically evaluate forward kinematics for a large number of configurations that result in greater task space error. These configurations are discarded. To avoid this computation, we propose to identify the relationship between task space error and joint space motion. Since this relationship is nonlinear, we use a first-order approximation.

The basic Jacobian, $\mathbf{J}^0$, is a matrix of partial derivatives relating joint space velocities to end-effector linear and angular velocities. We will compute $\mathbf{J}^t$, the Jacobian for the task frame. The Jacobian inverse will perform the inverse mapping, allowing us to identify joint space displacements that resolve task space error.

TASK_CONSTRAINED_RRT($\mathbf{q}_{init}, \Delta t$)
1 $\mathcal{T}.init(\mathbf{q}_{init})$;
2 **for** $a = 1$ **to** A
3 **do** $\mathbf{q}_{rand} \leftarrow$ RANDOM_CONFIG;
4 $\mathbf{q}_{near} \leftarrow$ NEAREST_NEIGHBOR($\mathbf{q}_{rand}, \mathcal{T}$);
5 $\mathbf{q}_{dir} \leftarrow (\mathbf{q}_{rand} - \mathbf{q}_{near})/|\mathbf{q}_{rand} - \mathbf{q}_{near}|$;
6 $\mathbf{q}_s = \mathbf{q}_{near} + \mathbf{q}_{dir}\ \Delta t$;
7 **if** *CONSTRAINED*_NEW_CONFIG($\mathbf{q}_s, \mathbf{q}_{near}$)
8 **then** $\mathcal{T}$. add_vertex ($\mathbf{q}_s$);
9 $\mathcal{T}$. add_edge ($\mathbf{q}_{near}, \mathbf{q}_s$);
10 **return** $\mathcal{T}$

COMPUTE_TASK_ERROR($\mathbf{q}_s, \mathbf{q}_{near}$)
1 $(\mathbf{C}, \mathbf{T_0^t}) \leftarrow$ RETRIEVE_CONSTRAINT($\mathbf{q_s}, \mathbf{q_{near}}$);
2 $\mathbf{T}_e^0 \leftarrow$ FORWARD_KINEMATICS($\mathbf{q}_s$);
3 $\mathbf{T}_e^t \leftarrow \mathbf{T}_0^t \mathbf{T}_e^0$;
4 $\Delta\mathbf{x} \leftarrow$ TASK_COORDINATES($\mathbf{T}_e^t$);
5 $\Delta\mathbf{x}_{err} \leftarrow \mathbf{C}\boldsymbol{\Delta}\mathbf{x}$
6 **return** $\Delta\mathbf{x}_{err}$;

Fig. 2. Pseudo-code for the Task-Constrained RRT (TC-RRT) construction algorithm. The word *CONSTRAINED* represents either RGD, TS or FR. COMPUTE_TASK_ERROR is common among all three subroutines.

A. Task Frame Jacobian

The Jacobian of a robot manipulator, $\mathbf{J}^0$, can be computed analytically given the kinematics of each joint at a sampled configuration $\mathbf{q}_s$ of the robot. [23] For end-effector position $\mathbf{p}^0$, joint positions $\mathbf{p}_i^0$ and oriented joint axes $\mathbf{z}_i^0$, we have:

$$\mathbf{J}^0 = \begin{bmatrix} \mathcal{J}_{P1} & \dots & \mathcal{J}_{Pn} \\ \mathcal{J}_{O1} & \dots & \mathcal{J}_{On} \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} \mathcal{J}_{Pi} \\ \mathcal{J}_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} \mathbf{z}_{i-1} \\ 0 \end{bmatrix} & \text{(prismatic joint)} \\ \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{p} - \mathbf{p}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix} & \text{(revolute joint)} \end{cases} \tag{12}$$

Typically, this computation is performed in the world frame $\mathcal{F}^0$. We invert the orientation of the task frame, $\mathbf{R}_t^0$, and transform the Jacobian into $\mathcal{F}^t$ as follows:

$$\mathbf{J}^t = \begin{bmatrix} \mathbf{R}_0^t & 0 \\ 0 & \mathbf{R}_0^t \end{bmatrix} \mathbf{J}^0 \tag{13}$$

The lower three rows of $\mathbf{J}^0$ and $\mathbf{J}^t$ map to angular velocities. However, angular velocities are not instantaneous changes to any actual parameters that can be used to represent orientation. Given the configuration $\mathbf{q}_s$, instantaneous velocities have a linear relationship $\mathbf{E}(\mathbf{q}_s)$ to the time derivatives of many position and orientation parameters.[2]

$$\mathbf{J}(\mathbf{q}_s) = \mathbf{E}(\mathbf{q}_s)\mathbf{J}^t(\mathbf{q}_s) \tag{14}$$

The matrix $\mathbf{E}(\mathbf{q})$ is presented in the Appendix for Cartesian translations and RPY rotations.

```
RGD_NEW_CONFIG(q_s, q_near)
 1  i ← 0;   j ← 0;
 2  Δx_err ← COMPUTE_TASK_ERROR(q_s, q_near);
 3  while i < I and j < J and |Δx_err| > ε
 4  do i ← i + 1;   j ← j + 1;
 5     q'_s = q_s + RANDOM_DISPLACEMENT(d_max);
 6     Δx'_err ← COMPUTE_TASK_ERROR(q_s, q_near);
 7     if Δx'_err < Δx_err
 8        then j ← 0;   q_s = q'_s;   Δx_err = Δx'_err;
 9     if Δx_err ≤ ε
10        then if IN_COLLISION(q_s)
11                then return false;
12                else return true;
13  return false;
```

```
TS_NEW_CONFIG(q_s, q_near)
1  (C, T_0^t) ← RETRIEVE_CONSTRAINT(q_s, q_near);
2  J ← JACOBIAN(q_near);
3  Δq = q_s − q_near;
4  Δq' = Δq − J†CJΔq;
5  q_s ← q_near + Δq';
6  return RGD_NEW_CONFIG(q_s, q_near);
```

Fig. 3. Pseudo-code for the Randomized Gradient Descent and Tangent Space Sampling strategies. Both methods constrain sampled configurations.

B. Jacobian Inverse

Having established a mapping of instantaneous motion between joint and task space we invert this relationship. For redundant manipulators, the Jacobian inverse does not exist and there are many joint motions that map to a single task space displacement. Our algorithms use the right pseudo-inverse, $\mathbf{J}^\dagger$, which maps instantaneous position error in frame $\mathcal{F}^t$ to the *least-norm velocities* in joint space required to correct it.

$$\mathbf{J}^\dagger = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1} \tag{15}$$

While there are many algorithms for computing the pseudo-inverse, we have used LU decomposition, a fast $O(n^3)$ operation. This method is faster than iterative Singular Value Decomposition and more commonly available than the Greville method [24].

VIII. Tangential and Orthogonal Techniques

The mapping between joint space and task space given in Section VII allows the introduction of two simple techniques for constrained sampling. Tangent Space Sampling (TS) projects each RRT sample into the linear tangent space of its nearest neighbor. First-Order Retraction (FR) iteratively approximates the minimal displacement that removes task space error and applies it to the sample.

A. Tangent Space Sampling

First, observe that any configuration $\mathbf{q}_{near}$ in the RRT is within tolerance of the task constraint manifold. For closed chains, [25] finds that small joint displacements from $\mathbf{q}_{near}$ that are tangent to the constraint manifold have a higher probability of also being within tolerance. In our case, these displacements have no instantaneous component in the direction of task error. In other words, for displacement $\Delta\mathbf{q}$ and Jacobian $\mathbf{J}(\mathbf{q}_{near})$,

$$\mathbf{C}\mathbf{J}\Delta\mathbf{q} = \mathbf{0}. \tag{16}$$

To use this insight in RRT search, let $\mathbf{q}_s$ be a sampled configuration at a small displacement $\Delta\mathbf{q}$ from $\mathbf{q}_{near}$. Eq. 17 projects the displacement into the null space of the task constraint.

$$\Delta\mathbf{q}' = (\mathbf{I} - \mathbf{J}^\dagger\mathbf{C}\mathbf{J})\Delta\mathbf{q} \tag{17}$$

Since $\mathbf{C}$ is idempotent, $\mathbf{C^2} = \mathbf{C}$. We verify that Eq. 16 holds as follows:

$$\begin{aligned}\mathbf{C}\mathbf{J}\Delta\mathbf{q}' &= (\mathbf{C}\mathbf{J} - \mathbf{C}\mathbf{C}\mathbf{J}\mathbf{J}^\dagger\mathbf{J})\Delta\mathbf{q} \\ &= (\mathbf{C}\mathbf{J} - \mathbf{C}\mathbf{J})\Delta\mathbf{q} = \mathbf{0}\end{aligned} \tag{18}$$

Eq. 17 represents the least-norm change to $\Delta\mathbf{q}$ that places it in the tangent space. Notice that this is distinct from the minimal $\Delta\mathbf{q}$ that achieves an equivalent task space displacement $\mathbf{J}\Delta\mathbf{q}'$. The random direction of joint space motion is largely preserved.

The projected sample is simply $\mathbf{q}'_s = \mathbf{q}_{near} + \Delta\mathbf{q}'$. Due to the non-linearity of the constraint manifold,$\mathbf{q}'_s$ is still unlikely to be within error tolerance. RGD is applied to the sample to further reduce task space error.

B. First-Order Retraction

Although similar in form to TS, First-Order Retraction behaves more like RGD in that it iteratively displaces the sample towards the constraint manifold. Unlike RGD, the displacement is not random but directed towards removing constraint error. Unlike TS, the Jacobian is computed at the sampled configuration $\mathbf{q}_s$ rather than $\mathbf{q}_{near}$.

Consider the retraction of a single configuration $\mathbf{q}_s$. First, we find the task space error, $\Delta\mathbf{x}_{err}$, according to Section VI-A. If the error exceeds tolerance, we compute $\Delta\mathbf{q}_{err}$, the least-norm joint space displacement that compensates for this error and adjust the sample to $\mathbf{q}'_s$:

$$\Delta\mathbf{q}_{err} = \mathbf{J}^\dagger\Delta\mathbf{x}_{err} \tag{19}$$

$$\mathbf{q}'_s = \mathbf{q}_s - \Delta\mathbf{q}_{err} \tag{20}$$

Since the Jacobian is a first-order approximation, changes in the task space do not linearly transform to changes in the joint space. Locally, this approximation allows us to follow a gradient descent procedure as shown in Figure 5. Experimentally, for small displacements, two iterations are sufficient for $\Delta\mathbf{x}_{err}$ to converge within tolerance.

Close to singularities, the pseudo-inverse may become unstable. To resolve this, we simply discard a configuration when the magnitude of the adjustment exceeds the original displacement.

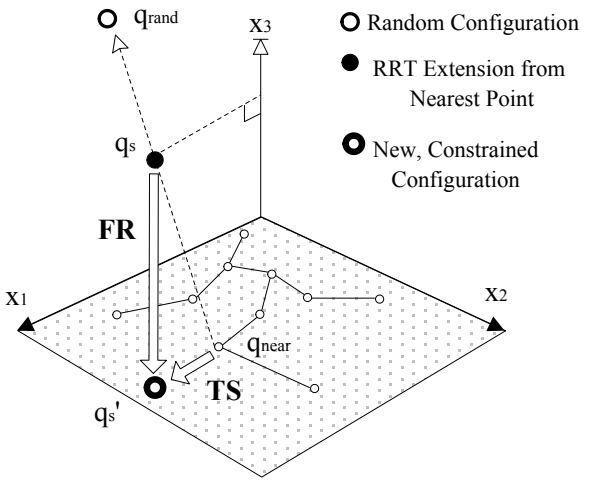


Fig. 4. A schematic for constrained sampling for a 3D task space. Motion is constrained to the x_1, x_2 plane. Nearest neighbors are found in joint space, where the task plane maps to a non-linear manifold.

FR_NEW_CONFIG$(\mathbf{q}_s, \mathbf{q}_{near})$
1 $\mathbf{q}_r \leftarrow \mathbf{q}_s$
2 $\Delta\mathbf{x}_{err} \leftarrow$ COMPUTE_TASK_ERROR$(\mathbf{q}_s, \mathbf{q}_{near})$;
3 **while** $|\Delta\mathbf{x}_{err}| > \epsilon$
4 **do** RETRACT_CONFIG$(\mathbf{q}_s, \Delta\mathbf{x}_{err})$;
5 **if** $|\mathbf{q}_s - \mathbf{q}_r| > |\mathbf{q}_r - \mathbf{q}_{near}|$
6 **then return false**;
7 $\Delta\mathbf{x}_{err} \leftarrow$ COMPUTE_TASK_ERROR$(\mathbf{q}_s, \mathbf{q}_{near})$;
8 **if** IN_COLLISION$(\mathbf{q}_s)$
9 **then return false**;
10 **return true**;

RETRACT_CONFIG$(\mathbf{q}_s, \Delta\mathbf{x}_{err})$
1 $\mathbf{J} \leftarrow$ JACOBIAN$(\mathbf{q}_s)$;
2 $\Delta\mathbf{q}_{err} \leftarrow \mathbf{J}^{\dagger}\Delta\mathbf{x}_{err}$;
3 $\mathbf{q}_s \leftarrow (\mathbf{q}_s - \Delta\mathbf{q}_{err})$;

Fig. 5. Pseudo-code for First-Order Retraction of sampled configurations. RETRACT_CONFIG is shown separately for clarity.

IX. RESULTS

We evaluated the proposed sampling strategies in three sets of simulated experiments: DOOR, DRAWER and PAINT. The task is to plan a path for a PUMA 560 manipulator mounted on a mobile base. The holonomic mobile base adds two degrees of freedom (dof) to the six-dof PUMA. The resulting system is redundant and capable of self-motion.

A. Experiment Design

In each example, the robot is given an initial grasping configuration from which we grow an RRT according to each constraint method. The robot must maintain the task constraint and find a goal configuration that satisfies a workspace criterion:

DOOR - Open the door past 0.6 rad.
DRAWER - Extend the drawer 0.3 m.
PAINT - Move the paint 6.0 m to the right.

Once a configuration is found that satisfies the constraint, the search is considered successful. During the search, the door only rotates at the hinge, the drawer only slides and the paint is not allowed to pitch or roll. Figures 6-8 show successful solutions and indicate that in each case the robot employs redundancy to avoid obstacles and joint limits.

To show the relative stability of our methods, we conducted experiments with different choices for step size, Δt, and error tolerance, ϵ. Each trial was performed 10 times and terminated when it was unsuccessful after 10 minutes. Both FR and TS required no additional parameter choices. For RGD, we set $I = 1000$, $J = 100$, $d_{max} = \Delta t/10.0$. These parameters resulted in the highest overall performance on the three examples in our preliminary testing.

For efficiency, the RRTs in all experiments used the VCollide collision checking package and the kd-tree nearest neighbor approximation.[26] For simplicity, the basic RRT algorithm was used without goal biasing of the samples. Computation was performed on an Intel T2600 2.16GHz processor, with an average memory load of 40MB.

B. Experimental Results

Tables I-III summarize the average computation time when all 10 trials succeeded. When at least one trial failed, the tables show the percentage of successful trials. The tables are blank when all 10 trials did not result in a solution within the allocated 10 minutes per trial.

Experiments DOOR and DRAWER favor FR in terms of both computation time and stability. In these examples, we see that RGD and TS perform optimally with the largest error tolerance and a step size of approximately .02. Lower performance when $\Delta t = .04$ is most likely caused by the distance of the sample from the constraint manifold. For RGD, this distance simply implies a larger number of iterations to reach tolerance. In the case of TS, a linear approximation to the constraint manifold becomes less accurate at for larger displacements.

For large tolerances, TS and RGD outperform FR in Experiment PAINT. This is likely due to two factors: the significantly larger space of valid configurations and its linearity. Since only two coordinates are constrained, random samples have a much higher likelihood of being within tolerance. Furthermore, translations of the base joints map directly to unconstrained coordinates. Observe, however, that even in this lightly constrained setting a smaller tolerance allows only FR to succeed.

Overall, we found that Jacobian based algorithms required less computation and performed comparably with RGD in the worst conditions. FR showed significantly more invariance with respect to error tolerance. We also observed an approximately linear relationship between computation time and time step. This increase is unavoidable since smaller time-steps increase the length of the motion plan.

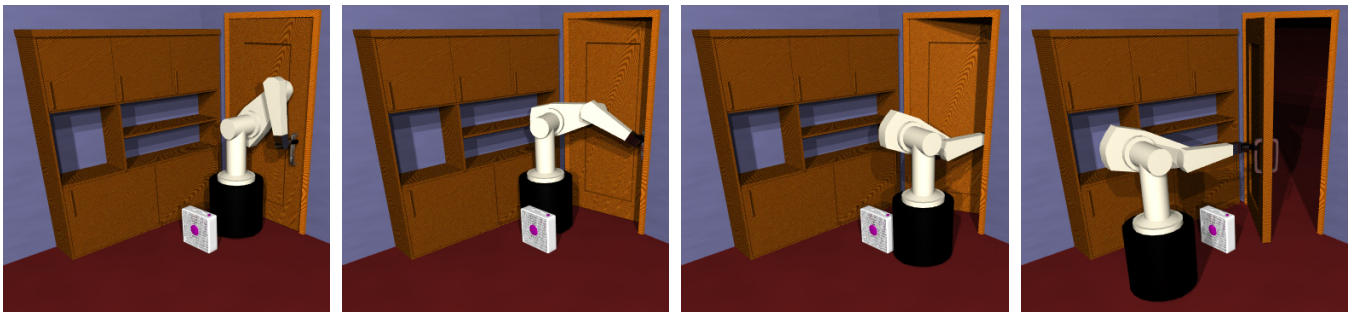


Fig. 6. DOOR: Still frames from our simulated environment demonstrating a motion plan to open the door. The robot must avoid joint limits and the box fan to successfully complete the plan.

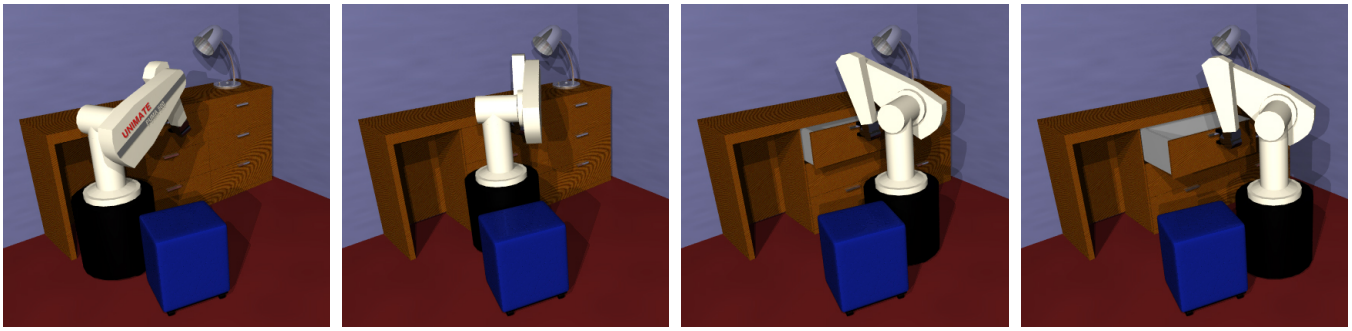


Fig. 7. DRAWER: A simulated experiment that requires the robot to open the drawer. The robot must first move around the footstool to allow sufficient space for the end-effector trajectory.

TABLE I

RUNTIMES FOR EXPERIMENT 1: DOOR

		Δt (Step Size)				
	$\epsilon(m,r)$	.04	.02	.01	.005	.0025
RGD	10^{-4}	**107.16**	**50.16**	**69.56**	**130.57**	**256.0**
	10^{-5}					60%
	10^{-6}					
TS	10^{-4}	**22.60**	**11.35**	**18.33**	**25.03**	**62.22**
	10^{-5}				90%	**114.77**
	10^{-6}					
FR	10^{-4}	**4.46**	**10.28**	**14.64**	**36.92**	**139.25**
	10^{-5}	**12.70**	**20.28**	**15.21**	**44.74**	**89.49**
	10^{-6}	**5.56**	**8.82**	**19.50**	**63.86**	**108.8**

TABLE II

RUNTIMES FOR EXPERIMENT 2: DRAWER

		Δt (Step Size)				
	$\epsilon(m,r)$	.04	.02	.01	.005	.0025
RGD	10^{-4}	**21.42**	**13.98**	**19.49**	**40.09**	**96.91**
	10^{-5}				**282.98**	90%
	10^{-6}					
TS	10^{-4}	**7.81**	**5.38**	**9.92**	**21.24**	**43.37**
	10^{-5}				**153.65**	**68.62**
	10^{-6}					
FR	10^{-4}	**1.58**	**4.53**	**6.05**	**19.52**	**34.44**
	10^{-5}	**2.50**	**3.40**	**6.18**	**16.84**	**33.72**
	10^{-6}	**1.64**	**4.65**	**8.18**	**17.21**	**32.45**

X. EXTENSIONS

In this paper we have addressed the description of workspace constraints for joint space planning and introduced a set of algorithms for implementing the constraints. Let us briefly address some of the simple extensions that can be applied to our framework. We will consider constraint intervals, alternative planning strategies, multiple constraints and generalized end-effector definitions.

A. Unilateral Constraints

The motion constraint vector $\mathbb{C}$ represents bilateral constraints, prohibiting positive and negative coordinate change. In some cases, a constraint requires that the coordinate remain above a threshold, within an interval or increase monotonically. These cases can be treated as either obstacles or parameterized constraints.

In the case of the former, samples that violate the constraint will be discarded. In analogy to DRAWER, where an actual obstacle prevents translation towards the cabinet, introducing a virtual boundary on motion will have the equivalent effect. The latter option specifies $c_i = 1$ or $c_i = 0$ for coordinate i based on whether or not the sampled configuration violates the boundaries of the constraint. This option will generate more valid samples, yet it may also bias these samples towards the constraint bounds.

B. Alternative Planning Strategies

The algorithms evaluated in Section IX were all based on the single-query RRT algorithm. RGD, however, was first introduced for applications in multi-query domains. In fact, both RGD and FR are well suited for use in PRMs and other multi-query algorithms. In either case, as long as the constraint distance metric is well-defined throughout

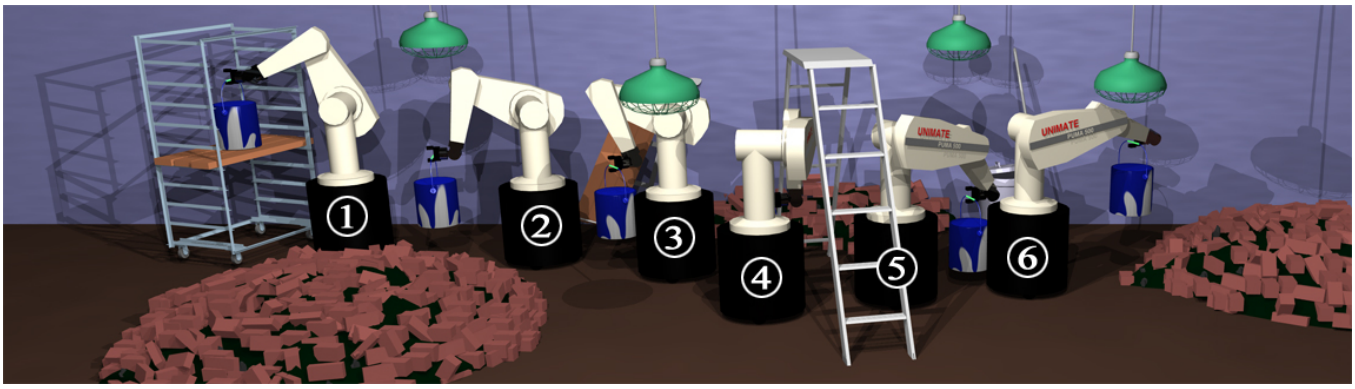


Fig. 8. PAINT: A simulated experiment where the PUMA must transport a can of paint. The plan is required to satisfy a parameterized constraint, $\mathbb{C} = [\ 0\ 0\ 0\ 1\ 1\ 0\ ]$, prohibiting rotations about the x and y axes.

the space, arbitrary configurations can be displaced towards the constraint manifold. For the purposes of connecting PRM samples, all three algorithms can be used directly as presented in this paper.

The generality of our methods extends to the use of heuristics during search. Since the only modification to the RRT algorithm is in the NEW_CONFIG method, most heuristic variants of RRT are easily extended to include the proposed strategies. For instance, the RRT samples may be biased towards the goal or multiple RRTs may be grown towards each other [27].

C. Multiple Constraints

For simplicity, this paper has focused on a single task constraint for the end-effector. However, some tasks require multiple end-effector constraints, more general specifications of end-effectors or multiple kinematic closure constraints. We address generalized end-effectors in Section X-D. For multiple task constraints, we simply form a composite constraint vector $\mathbb{C}_M$. Let $\mathbb{C}_i$ represent the constraint vector for task i. Then we have:

$$\mathbb{C}_M = [\ \mathbb{C}_1^T \mid \mathbb{C}_2^T \mid \cdots \mid \mathbb{C}_n^T\ ]^T \tag{21}$$

For the TS and FR algorithms, we also construct a generalized Jacobian matrix composed of the individual task Jacobians:

$$\mathbf{J}_M = \begin{bmatrix} \mathbf{J}_1 \\ \cdots \\ \mathbf{J}_n \end{bmatrix} \tag{22}$$

The computation of task space distance and the planning algorithms remain unchanged.

D. Abstract End-Effectors

The generalization of task definitions also extends to abstract end effectors. Section V-A displaced the end-effector frame to the grasped object. In fact, any point defined as a function of some robot link can be constrained. Moreover, the constrained point can be a function of multiple robot links. An important example is the robot center of mass (COM).

TABLE III

RUNTIMES FOR EXPERIMENT 3: PAINT

		Δt (Step Size)					
	$\epsilon(m,r)$	.08	.04	.02	.01	.005	.0025
RGD	10^{-4}	**18.88**	80%	**119.19**	80%	50%	10%
	10^{-5}	50%	80%	70%	50%	60%	10%
	10^{-6}						
TS	10^{-4}	**10.78**	**28.67**	**57.69**	**131.61**	80%	10%
	10^{-5}	60%	**293.88**	**186.94**	**209.14**	90%	1%
	10^{-6}						
FR	10^{-4}	**20.21**	**45.08**	**127.24**	**288.47**	60%	20%
	10^{-5}	**20.42**	**60.51**	**137.53**	90%	70%	10%
	10^{-6}	**22.36**	**42.95**	**126.94**	80%	30%	

The COM position is a linear combination of the positions $\mathbf{p}_{mi}$ of the individual link masses, m_i. For a total mass M,

$$\mathbf{p}_{com} = \tfrac{1}{M} \textstyle\sum_i m_i \mathbf{p}_{mi}. \tag{23}$$

This definition is sufficient to introduce a task frame and compute task space error. For TS and FR we also define the COM Jacobian as follows:

$$\mathbf{J}^0_{com} = [\ \mathcal{J}_{P1} \quad \cdots \quad \mathcal{J}_{Pn}\ ] \tag{24}$$

$$\mathcal{J}_{Pi} = \begin{cases} (\frac{1}{M}\sum_{j=i}^{n} m_j)\mathbf{z}_{i-1} & \text{(prismatic)} \\ \frac{1}{M}\sum_{j=i}^{n} m_j(\mathbf{z}_{i-1} \times (\mathbf{p}_{mj} - \mathbf{p}_{i-1})) & \text{(revolute)} \end{cases}$$

In the case where the end-effector is a non-linear function of the joints, the Jacobian can be approximated numerically by computing kinematics for small displacements in each joint.

XI. CONCLUSION

In this paper, we have presented a unified representation for task space constraints in the context of joint space motion planning. With regard to this representation, we described three algorithms for task constrained sampling in joint space. Our comparison of the algorithms indicated that First-Order Retraction is typically faster and significantly more invariant to step size and error tolerance than alternative techniques. Finally, we generalized our approach to unilateral constraints, abstract end-effector definitions, as well as alternative planning algorithms.

Our experimental findings regarding the efficiency and success of Jacobian based algorithms have additional implications for plan execution. Computing the Jacobian during

planning allows us to measure the manipulability of sampled configurations: $det(\mathbf{J}\mathbf{J}^T)^{1/2}$ [28]. Due to online errors, successful execution may require compliance or impedance control. Maintaining a manipulability threshold will ensure robot stability when it deviates from the motion plan.

Many facets of task constrained planning remain for future investigation. Each of the extensions in Section X can be explored. For instance, Section X-C applies our method with multiple hard constraints. A study of soft constraints may lead to results in biasing motion plans towards desirable robot postures. In these cases, task projection into the null space of $\mathbf{J}^\dagger$ could be used to prioritize constraints.[29], [30]

Of course, soft constraints are one of many possible deeper explorations of constrained search. We introduce such examples to show the modularity and accessibility of the methods described in this paper. Our proposed tools demonstrate the feasibility of task constrained joint space planning and provide a groundwork for further research.

XII. Acknowledgements

We are grateful to James Kuffner, Jaeheung Park and Chris Atkeson for their comments and advice throughout the development of this work. We thank Jan-Ullrich Schamburek for his contributions to the experimental environment that inspired and enabled this research.

Appendix

The examples in this paper, as well as many common constraints, can be expressed with a $\mathbb{C}$ vector in a coordinate system composed of Cartesian translation and fixed axis (roll/pitch/yaw) rotation. The use of any coordinate system requires us to derive translations of displacements.

For reference, we give the transformation from error in homogeneous coordinates to fixed axis coordinates. We also show the $\mathbf{E}(q)$ matrix for the fixed axis representation. Similar methods can be applied to other coordinate systems.

Since forward kinematics are typically expressed in terms of homogeneous transformations, Eq. 25 gives the mapping from a transformation matrix to Cartesian / fixed axis coordinates. The Cartesian component is equivalent to the last column of the transformation. For $T_{a,b}$ representing the value at row a, column b of T:

$$\begin{aligned} \psi &= atan2(T_{3,2}, T_{3,3}) \\ \theta &= -asin(T_{3,1}) \\ \phi &= atan2(T_{2,1}, T_{1,1}) \end{aligned} \tag{25}$$

In order to apply TS or FR, we also need to map velocities in workspace to velocities in task space. The angular velocity vector ω is found by a summation of fixed axis velocities in a common frame:

$$\omega = \begin{bmatrix} 0 \\ 0 \\ \dot{\phi} \end{bmatrix} + R_z(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R_z(\phi)R_y(\theta) \begin{bmatrix} \dot{\psi} \\ 0 \\ 0 \end{bmatrix} \tag{26}$$

This relationship is rearranged as a single matrix, $\mathbf{E}_\omega^{-1}(q)$:

$$\omega = \mathbf{E}_\omega^{-1}(q) \begin{bmatrix} \dot{\psi} & \dot{\theta} & \dot{\phi} \end{bmatrix}^T \tag{27}$$

Inverting $\mathbf{E}_\omega^{-1}(q)$ we find $\mathbf{E}_\omega(q)$, a matrix that maps angular velocities to fixed axis velocities. Adjoining the identity matrix for linear velocities Eq. 28 specifies $\mathbf{E}(q)$.

$$\mathbf{E}_{rpy}(q) = \begin{bmatrix} \mathbf{I_{3\times 3}} & \cdots & 0 & \cdots \\ \vdots & c_\phi/c_\theta & s_\phi/c_\theta & 0 \\ 0 & -s_\phi & c_\phi & 0 \\ \vdots & c_\phi s_\theta/c_\theta & s_\phi s_\theta/c_\theta & 1 \end{bmatrix} \tag{28}$$

References

[1] M. T. Mason. Compliance and force control for computer controlled manipulators. *IEEE Trans. on Systems, Man, and Cybernetics,*, 11(6), 1981.

[2] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics Research*, 3(1), 1987.

[3] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[4] J. Ahuactzin and K. Gupta. The kinematic roadmap: A motion planning based global approach for inverse kinematics of redundant robots. *IEEE Trans. on Robotics and Automation*, 15:653–669, 1999.

[5] Z. Guo and T. Hsia. Joint trajectory generation for redundant robots in an environment with obstacles. In *IEEE Int. Conf. on Robotics and Automation*, pages 157–162, 1990.

[6] G. Oriolo, M. Ottavi, and M. Vendittelli. Probabilistic motion planning for redundant robots along given end-effector paths. In *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2002.

[7] G. Oriolo and C. Mongillo. Motion planning for mobile manipulators along given end-effector paths. In *IEEE Int'l Conf. on Robotics and Automation (ICRA'05)*, 2005.

[8] D. P. Martin, J. Baillieul, and J. M. Hollerbach. Resolution of kienmatic redundancy using opetimization techniques. *IEEE Trans. on Robotics and Automation*, 5(4):529–533, 1989.

[9] B. Siciliano. Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent and Robotic Systems*, 3:201–212, 1990.

[10] S. Seereeram and J. T. Wen. A global approach to path planning for redundant manipulators. *IEEE Trans. on Robotics and Automation*, 11(1), 1995.

[11] A. McLean and S. Cameron. The virtual springs method: Path planning and collision avoidance for redundant manipulators. 15, 1996.

[12] Oliver Brock, Oussama Khatib, and Sriram Viji. Task-consistent obstacle avoidance and motion behavior for mobile manipulation. In *IEEE Int'l Conf. on Robotics and Automation*, 2002.

[13] O. Khatib, L. Sentis, J. Park, and J. Warren. Whole body dynamic behavior and control of human-like robots. *International Journal of Humanoid Robotics*, pages 29–44, 2004.

[14] L. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, 12(4), 1996.

[15] S.M.LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Dept., Iowa State University, 1998.

[16] S.M. LaValle and J.J. Kuffner. Rapidly exploring random trees: Progress and prospects. In *Workshop on the Algorithmic Foundations of Robotics*, 2000.

[17] S. M. LaValle, J. Yakey, and L. E. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *In Proc. IEEE Int'l Conf. on Robotics and Automation*, 1999.

[18] L. Han and N. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.

[19] J. Cortes T. Simeon and J.P. Laumond. A random loop generator for planning the motions of closed kinematic chains with prm methods. In *IEEE Int. Conf. on Robotics and Automation*, 2002.

[20] J.J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots under obstacle and dynamic balance constraints. In *IEEE Int'l Conf. on Robotics and Automation (ICRA'01)*, pages 692–698, 2001.

[21] Z. Yao and K. Gupta. Path planning with general end-effector constraints: Using task space to guide configuration space search. In *IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2005.

[22] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. 11:11–73, 1997.
[23] L. Sciaviccio and B. Siciliano. *Modeling and control of robot manipulators.* McGraw-Hill Co., 1996.
[24] T. N. E. Greville. Some applications of the pseudoinverse of a matrix. *SIAM Review*, 2:15–22, 1960.
[25] J. Yakey, S.M. LaValle, and L. E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *IEEE Transactions on Robotics and Automation*, 17(7), 2001.
[26] A. Atramentov and S. M. LaValle. Efficient nearest neighbor searching for motion planning. In *IEEE Int'l Conf. on Robotics and Automation (ICRA'02)*, 2002.
[27] J. Kuffner and S. M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Workshop on the Algorithmic Foundations of Robotics*, pages 995–1001.
[28] T. Yoshikawa. Manipulability of robotic mechanisms. *Int. Journal of Robotics Research*, 4:3–9, 1985.
[29] H. Hanafusa, T. Yoshikawa, and Y. Nakamura. Analaysis and control of articulated robot with redundancy. In *IFAC, 8th Triennal World Congress*, volume 4, pages 1927–1932, 1981.
[30] R. Boulic P. Baerlocher. Task-priority formulations for the kinematic control of highly redundant atriculated structures. In *Int. Conf. on Intelligent Robots and Systems*, 1998.