Global Manipulation Planning in Robot Joint Space With Task Constraints

Mike Stilman

Abstract—**We explore global randomized joint-space path planning for articulated robots that are subjected to task-space constraints. This paper describes 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). FR is formally proven to provide global sampling for linear task-space transformations. Constrained joint-space planning is important for many real-world problems, which involves redundant manipulators. On the one hand, tasks are designated in workspace coordinates: to rotate doors about fixed axes, to slide drawers along fixed trajectories, or to hold objects level during transport. On the other hand, joint-space planning gives alternative paths that use redundant degrees of freedom (DOFs) to avoid obstacles or satisfy additional goals while performing a task. We demonstrate that our methods are faster and more invariant to parameter choices than the techniques that exist.**

Index Terms—**Kinematics, manipulation planning, path planning for manipulators, path planning with task constraints, sampling-based planning, service robots.**

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 aligning beams for construction and removing 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 task constraints without being trapped in local minima. An early version of this paper was presented in [1]. It is now expanded with a formal proof of global sampling and, further, theoretical analysis.

II. Related Work

In addition to find a collision-free joint-space path [2], our problem requires that each configuration along the path must satisfy workspace constraints for the end-effector [3], [4]. We distinguish this problem from a single goal pose to specify for the end-effector [5] or a path that the end-effector must follow [6]–[8]. 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, to handle robot redundancy poses a difficult challenge. Typically, redundancy resolution

Manuscript received December 4, 2008; revised May 24, 2009 and July 27, 2009; accepted November 16, 2009. Date of current version June 9, 2010. This paper was recommended for publication by Associate Editor F. Lamiraux and Editor J.-P. Laumond upon evaluation of the reviewers' comments.

The author is with the Robotics and Intelligent Machines Center, Department of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: mstilman@cc.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2010.2044949

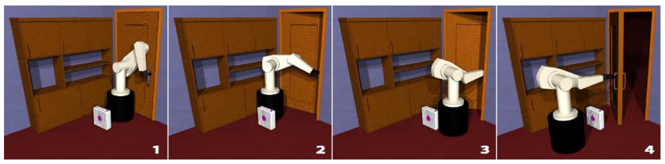

Fig. 1. Door: Simulated motion plan to open the door. The robot must avoid joint limits and the box fan to successfully complete the plan.

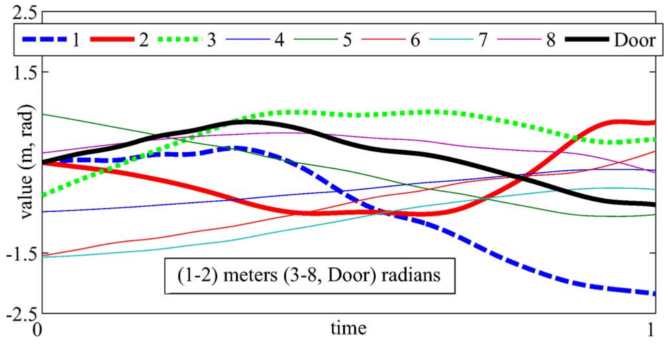

Fig. 2. Door (FR): Paths for robot joints and door rotation (about z-axis) in Fig. 1. (1) and (2) Base translation and (3) rotation diverge significantly from straight paths to globally avoid obstacles and satisfy constraints.

is performed with local [9], [10] or global [11] techniques for optimal control. These methods optimize configuration-dependent criteria, such as manipulability. Obstacle distance, which is a common criterion for collision avoidance, has a highly nonlinear relationship to joint configurations and leads to the use of local optimization [12]–[14]. These methods require the generation of distance/potential functions for obstacles. Moreover, they do not find solutions in the presence of local minima that exist in all three of our examples.

For a more comprehensive exploration of the search space, motion-planning research has headed toward feasible solutions and probabilistically complete algorithms, such as probabilistic roadmap method (PRM) [15] and rapidly exploring random trees (RRT) [16], [17]. 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 either not significant or is zero. This is shown for problems that involve closed chains in [18] and [19].

To address this challenge, some approaches use domain-specific attributes, such as closed-chain structure [18]–[23]. Tang *et al.* [21], [22] does not incorporate joint limits or constraints on end-effector orientations. Likewise, dynamic filtering in [23] does not consider kinematic constraints. These methods are not sufficiently general to handle the most common types of geometric and kinematic constraints. Other closed-chain techniques, such as [19] and [20], are similar to methods that are described in task-constrained planning, such as [8], [18], and [24], and would require the overhead of translating task-planning problems into problems of closed chains. We restrict our discussion to the adaptations that exist. Among these techniques, one algorithm, which is known as randomized gradient descent (RGD) [18] has been directly extended for use with arbitrary constraints [24], [25]. RGD randomly shifts sampled configurations in directions that minimize constraint error. However, Yao and Gupta [24] applies this algorithm with general constraints on a case-by-case basis. We propose to adapt the task-frame formalism that exists [3] to unify the representation of constraints, and initiate the comparison of algorithms for constrained sampling.

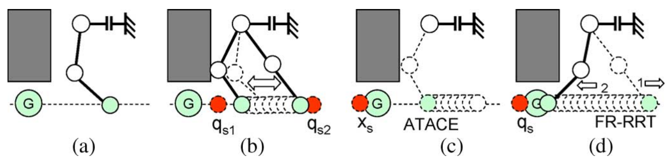

Fig. 3. (a) PRR manipulator, end-effector constraint line, and goal. (b) Space explored by both the proposed planners and ATACE. (c) ATACE selects the nearest neighbor in task space, always choosing elbow-down. (d) Our planners distinguish between joint-space configurations to find the solution.

Our investigation considered ATACE [24], [25] as well as the path-following strategies from [7] and [8]. In contrast to planners that exist, our strategies solve strictly larger classes of problems by satisfying the conditions for global probabilistic completeness, as shown in Section IX. For instance, consider the planar PRR manipulation problem shown in Fig. 3. There exists only one solution, and it requires that the manipulator first change to an elbow-up configuration prior to reaching the goal. Both ATACE and our proposed strategies explore the space in Fig. 3(b). However, ATACE considers task space motions with RRTs and locally follows them in joint space. After the initial exploration, whenever a sample is chosen close to the goal, ATACE chooses the joint-space configuration with a *workspace* distance that is closest to the goal. This is always an elbow-down configuration. In contrast, our methods respect the distinction between joint-space configurations that map to same workspace point. Given a joint-space sample with the elbow down, a planner using tangent-space sampling RRT (TS-RRT) or first-order retraction RRT (FR-RRT) continues to make progress toward the goal by first straightening the arm and then bending it in the opposite direction.

The three examples given in this paper require the robot to explore multiple joint-space paths along a single path in task space. Although [7] and [8] also consider alternatives in joint space, they prespecify the end-effector path itself. Our method does not ask for an initial workspace path. Furthermore, these methods depend on 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. Such algorithms rely 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. Our strategies find both joint-space and end-effector paths with no initialization. They perform equally well, regardless of which joint motions are constrained.

This paper presents three significant contributions in the domain of constrained joint-space planning. First, we propose two new global-sampling schemes that exploit joint redundancy to explore the entire configuration space of the robot subject to constraints. These are the first such techniques that promise a globally resolution complete planning, under the assumption of local linearity. Second, we introduce the intuitive formulation of task-space constraints to the sampling-based motion-planning community. Finally, we evaluate our new sampling strategies in comparison with methods that exist. Our strategies are shown to be experimentally successful in comparison to RGD with regard to different tasks and parameter choices.

III. Representation of Constraints

A task constraint is a restriction on the freedom of motion of a robot end-effector [3], [4]. We use the following three spaces of coordinates:

$\mathbf{q}_i$ joint-space coordinates refer to a vector of single-axis translations and rotations of the robot joints;

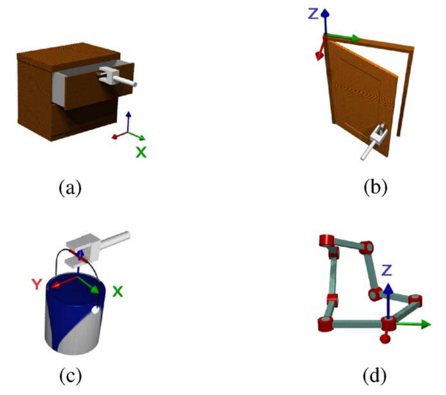

Fig. 4. Examples of constraints implemented with a roll/pitch/yaw specification of task coordinates. The task frames in (c) and (d) are parameterized by the configuration of the robot. (a) Fixed $\mathbb{C} = [\,0\;1\;1\;1\;1\;1\,]^T$. (b) Fixed $\mathbb{C} = [\,1\;1\;1\;1\;1\;0\,]^T$. (c) Parameterized $\mathbb{C} = [\,0\;0\;0\;1\;1\;0\,]^T$. (d) Parameterized $\mathbb{C} = [\,1\;1\;1\;1\;1\;0\,]^T$.

$\mathbf{T}_B^A$ workspace 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 (see Section III).

Note that task space is equivalent to the workspace by rigid transformation. The distinction allows for a simple formulation of constraints. We 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.

The degrees of freedom (DOFs) 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.

Given a task frame, we can choose from a variety of coordinate systems in which to quantify 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 [4]. For any representation, we define $\mathbb{C}$, which is 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.

We can also represent $\mathbb{C}$ as a diagonal-selection matrix [3] as follows:

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

We use $\mathbb{C}$ and $\mathbf{C}$ interchangeably. Without the loss of generality, this paper focuses on Cartesian translations and roll/pitch/yaw fixed-axis rotations. Rotations are defined about the x_t, y_t, and z_t axes of $\mathcal{F}^t$

$$\mathbf{R}_B^t = R(z_t, \phi) R(y_t, \theta) R(x_t, \psi). \tag{2}$$

Coordinates determine permitted translations and rotations about the task-frame axes. For instance, our $\mathbb{C}$ is six dimensional

$$\mathbb{C}_{RPY} = [\, c_x \quad c_y \quad c_z \quad c_\psi \quad c_\theta \quad c_\phi \,]^T. \tag{3}$$

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.

IV. Specifying Constraints

The definition given in Section III generalizes to many common constraint definitions. This section shows a spectrum of possibilities.

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 occur when objects are manipulated, which are kinematically linked to the environment (see Figs. 1 and 9). $\mathcal{F}^t$ is 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 the position and orientation of the object. $\mathbb{C}_t$ indicates which axes of $\mathcal{F}_t$ permit valid displacements.

In constrained manipulation, assume that the object is rigidly attached to the end-effector after grasp. At the time of grasp, let $\mathbf{T}_A^0$ and $\mathbf{T}_e^0$ represent the position of object A and the end-effector, respectively. The grasp transform $\mathbf{T_G}$ is given as follows:

$$\mathbf{T_G} = \mathbf{T}_A^e = \mathbf{T}_0^e\mathbf{T}_A^0 = (\mathbf{T}_e^0)^{-1}\mathbf{T}_A^0. \tag{4}$$

During manipulation, the kinematics of the end-effector are extended to include the manipulated object

$$\mathbf{T}_{e'}^0(\mathbf{q}) = \mathbf{T}_e^0(\mathbf{q})\mathbf{T_G}. \tag{5}$$

In (5), $\mathbb{C}$ directly specifies the permitted displacements of the object.

B. Simple Frame Parameters

Section IV-A described a universal 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. For instance, a task may consist of manipulation of a sequence of constrained objects [26]. Each object is assigned a distinct task frame and constraint vector. The planner selects the task frame that is based on the object 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 to transport open containers of liquid, such as paint (see Fig. 10). 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 multiarm manipulation and reconfigurable robots is a linkage with a closed kinematic chain. One approach to specify kinematic closure is to break the chain at an arbitrary joint $\mathbf{j}_k$. The linkage becomes an open chain with a constraint that was 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 workspace transform $\mathbf{T}(\mathbf{q})$ that describes $\mathbf{j}_k$ with respect to $\mathcal{F}^0$. One of these transforms can be specified as the task frame $\mathbf{T}_t^0(\mathbf{q})$, while the others are end-effectors $\mathbf{T}_e^0(\mathbf{q})$. As with previous examples, the constraint vector intuitively specifies the DOF for the joint $\mathbf{j}_k$.

D. Constraint Paths and Surfaces

Some constraints may require the end-effector to remain on a nonlinear path or maintain contact with a complex surface. One may

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

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

V. Introducing Constrained Sampling

Section IV-A–D introduced a subspace of constraints for the end-effector. These are task-space coordinates x in $\mathcal{F}^t$, such that $x_i = 0$ when $c_i = 1$. In robot joint space, they map to a nonlinear manifold. Randomized joint-space planners select samples that lie outside the constraint manifold. Our methods use a distance metric in task space to project samples within ϵ-tolerance of the constraint.

A. Computing Distance

Having identified a sampled configuration $\mathbf{q}_s$, we compute the the forward kinematics. Typically, the end-effector frame $\mathcal{F}^e$ is found in SE (3), as the transformation $\mathbf{T}_e^0(\mathbf{q}_s)$. We also identify $\mathcal{F}^e$ with respect to the task frame $\mathcal{F}^t$ as follows:

$$\mathbf{T}_e^t(\mathbf{q}_s) = \mathbf{T}_0^t\mathbf{T}_e^0(\mathbf{q}_s) = (\mathbf{T}_t^0)^{-1}\mathbf{T}_e^0(\mathbf{q}_s). \tag{6}$$

We can now find the relationship between the end-effector and the task frame as follows. More detail is given in the Appendix

$$\Delta\mathbf{x} \equiv \mathbf{T}_e^t(\mathbf{q}_s). \tag{7}$$

The task-space error is found simply by the product in (8). This product has the effect of selection presented in (9)

$$\Delta\mathbf{x}_{\text{err}} = [\, e_1 \quad \dots \quad e_n \,] = \mathbf{C\Delta x} \tag{8}$$

$$e_i = \begin{cases} 0, & c_i = 0 \\ \Delta\mathbf{x}_i, & c_i = 1. \end{cases} \tag{9}$$

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 that we compare are RGD [18], [24], TS, and FR. For simplicity, we will describe each approach as a modification to the basic RRT algorithm summarized in Fig. 5. The algorithm samples a random configuration $\mathbf{q}_{\text{rand}}$, and finds its nearest neighbor in the tree $\mathbf{q}_{\text{near}}$. The sampled configuration $\mathbf{q}_s$ is placed at a fixed distance Δt along the vector from $\mathbf{q}_{\text{near}}$ to $\mathbf{q}_{\text{rand}}$.

As a baseline, consider the RGD algorithm detailed in Fig. 6. After computing the task-space error of $\mathbf{q}_s$, RGD continues to uniformly sample the neighborhood of the configuration within a radius $d_{\max}$. The error of each new configuration $\mathbf{q}_s'$ is compared with $\mathbf{q}_s$. If the error is less, $\mathbf{q}_s'$ replaces $\mathbf{q}_s$. The procedure terminates after a maximum number of iterations or an error less than ϵ.

```
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. 5. Pseudocode 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.

```
RGD_NEW_CONFIG($\mathbf{q}_s, \mathbf{q}_{near}$)
1  $i \leftarrow 0$;  $j \leftarrow 0$;
2  $\Delta\mathbf{x}_{err} \leftarrow$ COMPUTE_TASK_ERROR($\mathbf{q}_s, \mathbf{q}_{near}$);
3  while $i < I$ and $j < J$ and $|\Delta\mathbf{x}_{err}| > \epsilon$
4  do $i \leftarrow i + 1$;  $j \leftarrow j + 1$;
5      $\mathbf{q}_s' = \mathbf{q}_s +$ RANDOM_DISPLACEMENT($\mathbf{d_{max}}$);
6      $\Delta\mathbf{x}_{err}' \leftarrow$ COMPUTE_TASK_ERROR($\mathbf{q}_s, \mathbf{q}_{near}$);
7      if $\Delta\mathbf{x}_{err}' < \Delta\mathbf{x}_{err}$
8        then $j \leftarrow 0$;  $\mathbf{q}_s = \mathbf{q}_s'$;  $\Delta\mathbf{x}_{err} = \Delta\mathbf{x}_{err}'$;
9      if $\Delta\mathbf{x}_{err} \leq \epsilon$
10       then if IN_COLLISION($\mathbf{q}_s$)
11                then return false;
12                else return true;
13 return false;
```

```
TS_NEW_CONFIG($\mathbf{q}_s, \mathbf{q}_{near}$)
1  $(\mathbf{C}, \mathbf{T_0^t}) \leftarrow$ RETRIEVE_CONSTRAINT($\mathbf{q_s}, \mathbf{q_{near}}$);
2  $\mathbf{J} \leftarrow$ JACOBIAN($\mathbf{q}_{near}$);
3  $\Delta\mathbf{q} = \mathbf{q}_s - \mathbf{q}_{near}$;
4  $\Delta\mathbf{q}' = \Delta\mathbf{q} - \mathbf{J}^\dagger\mathbf{C}\mathbf{J}\boldsymbol{\Delta}\mathbf{q}$;
5  $\mathbf{q}_s \leftarrow \mathbf{q}_{near} + \Delta\mathbf{q}'$;
6  return RGD_NEW_CONFIG($\mathbf{q}_s, \mathbf{q}_{near}$);
```

Fig. 6. Pseudocode for the RGD and TS strategies. Both methods constrain sampled configurations.

VI. 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. To avoid this computation, we identify the relationship between task-space error and joint-space motion. Since the relationship is nonlinear, we use a first-order approximation.

The basic Jacobian $\mathbf{J}^0$ is a matrix of partial derivatives that relates joint-space velocities to end-effector linear and angular velocities. We compute the task-frame Jacobian $\mathbf{J}^t$ and use its inverse to find joint-space displacements that resolve task-space error.

A. *Task-Frame Jacobian*

The manipulator Jacobian $\mathbf{J}^0$ is computed analytically, given the kinematics of each joint in configuration $\mathbf{q}_s$ [27]. Each column $\mathcal{J}_i$ represents the contribution of joint i

$$\mathbf{J}^0 = [\,\mathcal{J}_1 \quad \dots \quad \mathcal{J}_n\,] \tag{10}$$

$$[\,\mathcal{J}_i\,] = \begin{cases} \begin{bmatrix} \mathbf{z}_{i-1} \\ 0 \end{bmatrix} & \text{(prismatic joint)} \\ \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{p} - \mathbf{p}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix} & \text{(revolute joint).} \end{cases} \tag{11}$$

This computation is performed in the world frame $\mathcal{F}^0$. We transform the Jacobian into the task frame $\mathcal{F}^t$ as follows:

$$\mathbf{J}^t = \begin{bmatrix} \mathbf{R}_0^t & 0 \\ 0 & \mathbf{R}_0^t \end{bmatrix} \mathbf{J}^0. \tag{12}$$

The lower three rows of $\mathbf{J}^t$ map to angular velocities that are not necessarily equivalent to changes in task parameters. In configuration $\mathbf{q}_s$, instantaneous velocities are linearly related by $\mathbf{E}(\mathbf{q}_s)$ to the time derivatives of task parameters, as given in the Appendix [4]

$$\mathbf{J}(\mathbf{q}_s) = \mathbf{E}(\mathbf{q}_s)\mathbf{J}^t(\mathbf{q}_s). \tag{13}$$

B. *Jacobian Inverse*

Given an instantaneous mapping between joint and task spaces, we invert this relationship. For redundant manipulators, many joint displacements map to a single task-space displacement. Our algorithms use the right pseudoinverse $\mathbf{J}^\dagger$ to map instantaneous error in frame $\mathcal{F}^t$ to the *least-norm* joint-space velocities that correct it

$$\mathbf{J}^\dagger = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}. \tag{14}$$

While there are many algorithms to compute the pseudoinverse, we have used LU decomposition ($O(n^3)$) for speed. LU is faster than iterative singular value decomposition and more common than the Greville method [28].

VII. TANGENTIAL AND ORTHOGONAL TECHNIQUES

The differential mapping between joint space and task space yields two techniques for constrained sampling. TS projects each joint-space sample into the linear tangent space of its nearest neighbor. FR iteratively applies the minimal displacement that removes task-space error.

A. *Tangent-Space Sampling*

First, observe that any existing $\mathbf{q}_{\text{near}}$ in the RRT is within tolerance of the constraint manifold. For closed chains, Yakey *et al.* [29] find that small joint displacements from $\mathbf{q}_{\text{near}}$, which 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. For displacement $\Delta\mathbf{q}$ and Jacobian $\mathbf{J}(\mathbf{q}_{\text{near}})$

$$\mathbf{C}\mathbf{J}\boldsymbol{\Delta}\mathbf{q} = \mathbf{0}. \tag{15}$$

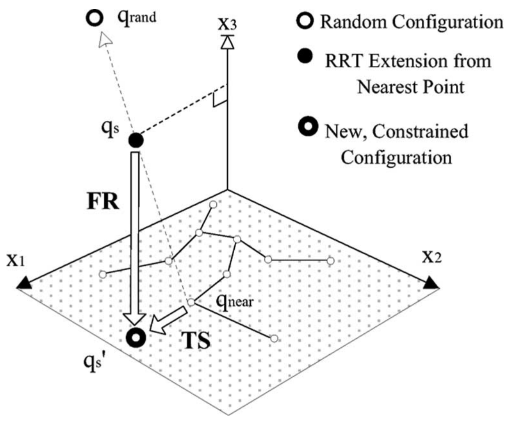
qrand
x3
Random Configuration
RRT Extension from Nearest Point
New, Constrained Configuration
qs
FR
x1
x2
qnear
TS
qs'

Fig. 7. Joint-space samples for a 3-D task space are retracted to the plane.

To apply this principle during RRT search, let $\mathbf{q}_s$ be a sampled configuration at a small displacement $\Delta\mathbf{q}$ from $\mathbf{q}_{\text{near}}$ and project it into the null space of the task constraint

$$\Delta\mathbf{q}' = (\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{C}\mathbf{J})\Delta\mathbf{q}. \tag{16}$$

Since $\mathbf{C}$ is idempotent, $\mathbf{C}^2 = \mathbf{C}$, and (15) can be verified as follows:

$$\mathbf{C}\mathbf{J}\Delta\mathbf{q}' = (\mathbf{C}\mathbf{J} - \mathbf{C}\mathbf{C}\mathbf{J}\mathbf{J}^{\dagger}\mathbf{J})\Delta\mathbf{q} = (\mathbf{C}\mathbf{J} - \mathbf{C}\mathbf{J})\Delta\mathbf{q} = \mathbf{0}. \tag{17}$$

Equation (16) represents the least-norm change to $\Delta\mathbf{q}$ that places it in the tangent space. Note that $\Delta\mathbf{q}'$ is distinct from the minimal joint motion that achieves an equivalent task-space displacement $\mathbf{J}\Delta\mathbf{q}'$.

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

B. First-Order Retraction

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

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

$$\Delta\mathbf{q}_{\text{err}} = \mathbf{J}^{\dagger}\Delta\mathbf{x}_{\text{err}} \tag{18}$$

$$\mathbf{q}_s' = \mathbf{q}_s - \Delta\mathbf{q}_{\text{err}}. \tag{19}$$

Since the Jacobian is a first-order approximation, task space changes do not linearly map to changes in the joint space. Locally, we apply gradient descent, as shown in Fig. 8.

Close to singularities, the pseudoinverse may become unstable. To resolve this, we discard samples when the magnitude of adjustment exceeds the original displacement. This choice may lead to inefficiency when sampling around singular configurations, and future improvements should consider early detection of such conditions.

VIII. Results

We evaluated the sampling strategies on three sets of simulated experiments: Door, Drawer and Paint. The task is to plan a path for a PUMA 560 manipulator on a holonomic mobile base. The base adds two redundant DOFs to the 6-DOF PUMA.

```
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. 8. Pseudocode for FR of sampled configurations. Retract_Config is shown separately for clarity.

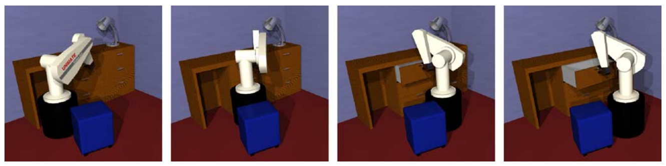

Fig. 9. Drawer: Simulated experiment that requires the robot to open the drawer. The robot must first move around the footstool to allow sufficient space for the end-effector trajectory.

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 achieve a goal that satisfies following 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. Figs. 1–10 show successful solutions, where 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 ten times and terminated when it was unsuccessful after 10 min. Both FR and TS required no additional parameter choices. For RGD, we set $I = 1000$, $J = 100$, and $d_{\text{max}} = \Delta t/10.0$. 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 [30]. For simplicity, the basic RRT algorithm was used without goal biasing of the samples. Computation was performed on an Intel T2600 2.16-GHz processor, with an average memory load of 40 MB. Final trajectories were smoothed with cubic splines.

Tables I–III summarize the average computation time when all ten trials succeeded. When at least one trial failed, the tables show the percentage of successful trials. The tables were blank when none of the ten trials resulted in a solution within the allocated 10 min per trial.

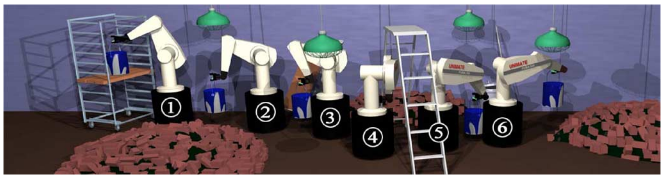

Fig. 10. Paint: Simulated experiment where the PUMA must transport a can of paint. The plan is required to satisfy a parameterized constraint $\mathbb{C} = [\ 0\ 0\ 0\ 1\ 1\ 0\ ]$, thus prohibiting rotations about the x and y axes.

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**

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%	

Experiments DOOR and DRAWER favor FR in terms of both computation time and stability. In these examples, we can see that RGD and TS perform optimally with the largest error tolerance and a step size of approximately 0.02. Lower performance, when $\Delta t = 0.04$, is most likely caused by the distance of the sample from the constraint manifold. For RGD, this distance implies a larger number of iterations to reach tolerance. In the case of TS, a linear approximation to the constraint manifold is less accurate for larger displacements.

For large tolerances, TS and RGD outperform FR in experiment PAINT (see Fig. 11). This is due to two factors: the significantly larger space of valid configurations and its linearity. Since only two coordinates are constrained, random samples have a higher probability of being within tolerance. Furthermore, translations of the base joints map directly to unconstrained coordinates. Observe that even in a lightly constrained setting, only FR succeeds for lower tolerances.

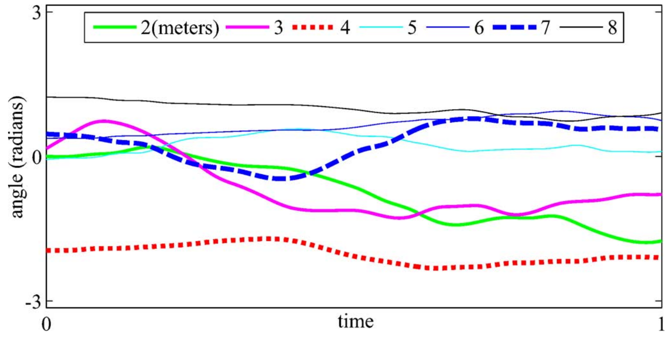

Fig. 11. Paint (FR): Paths for joints 2–8 in Paint experiment. (2) and (3) are base translation/rotation. Joint 4 raises the arm to avoid debris and then lowers under the ladder. Joint 7 moves accordingly to maintain the constraint.

Overall, we found that Jacobian-based algorithms required less computation and performed comparably with RGD under 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.

IX. ANALYSIS

Section VIII showed the efficiency of FR, and Section II demonstrated that FR solves a broader class of problems than the methods that exist. We now prove that underlying planners, such as RRT or PRM, retain probabilistic completeness under FR retraction to ensure the global planning when using FR. Typically, probabilistic completeness requires that any valid ϵ-neighborhood of any joint configuration $\mathbf{q}$ have a nonzero probability of being sampled [15] and [31]. We show that joint-space sampling followed by FR retraction yields the desired nontrivial probability for all ϵ-neighborhoods of valid configurations on the constraint manifold. We conclude that FR planner considers all redundant joint-space paths when solving for a task-space goal.

First, assume that the transformation between robot joint coordinates and end-effector coordinates is locally linear. This assumption is reasonable, since we apply retraction in a small neighborhood of joint displacements. The Jacobian, which is a matrix of partial derivatives of $\mathbf{x}$ with respect to $\mathbf{q}$, is then constant for all joint configurations, thus resulting in the linear relationship $\mathbf{x} = \mathbf{J}\mathbf{q}$.

The constraint manifold $\mathcal{M}$ is the solution to a set of linear equations in $\mathbf{q}$ of the form $a_1 q_1 + a_2 q_2 + \ldots a_n q_n = 0$. These equations define a hyperplane in $\mathbb{R}^n$, where $\mathbb{R}^n$ is the n-dimensional robot joint space. The constraint hyperplane has no volume in $\mathbb{R}^n$, and the probability of sampling a constrained configuration is zero.

We define FR with the function f that maps any joint sample $\mathbf{p}$ to its retraction $\mathbf{q}$ on the hyperplane: $f : \mathbb{R}^n \to \mathcal{M}$. To define f, we first compute task-space error: $\Delta\mathbf{x}_{\text{err}} = \mathbf{C}\mathbf{J}\mathbf{p}$. In the linear case, the definition of FR (18) is reduced to the following map:

$$\mathbf{q} = f(\mathbf{p}) = \mathbf{p} - \mathbf{J}^{\dagger}\mathbf{C}\mathbf{J}\mathbf{p} = (\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{C}\mathbf{J})\mathbf{p}. \tag{20}$$

Note that a constant $\mathbf{J}$ implies that $\mathbf{J}^{\dagger} = \mathbf{J}^T(JJ^T)^{-1}$ is also a constant matrix. Iteration is not required, since f results in configurations with zero-task-space error. Equation (21) shows that the task-space configuration $\mathbf{x}$ of a retracted $\mathbf{q}$ must be zero in each of the error components,

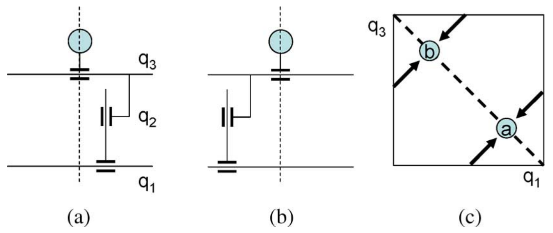

Fig. 12. Dashed vertical line is a constraint on end-effector position. (a) and (b) Redundant joint configurations that yield equivalent task configurations. (c) FR maps distinct hyperplanes in joint space that are orthogonal to the constraint to distinct valid configuration on the constraint manifold.

as represented by ones on the diagonal of $\mathbf{C}$

$$\begin{aligned}\mathbf{x_q} &= \mathbf{Jq} = \mathbf{J}f(\mathbf{p}) = \mathbf{Jp} - \mathbf{JJ}^\dagger\mathbf{CJp}\\ &= \mathbf{Jp} - \mathbf{CJp} = (\mathbf{I}-\mathbf{C})\mathbf{Jp}.\end{aligned} \tag{21}$$

This proves that f is a mapping from $\mathbb{R}^n$ to the constraint manifold $\mathcal{M}$. To demonstrate that any ϵ-neighborhood of any $\mathbf{q}$ on $\mathcal{M}$ has a nonzero probability of being sampled, we first show that f is *surjective* and *continuous*.

Lemma 9.1: The FR function f is surjective.

Proof: Any configuration $\mathbf{q}$ on the manifold $\mathcal{M}$ is itself an element of $\mathbb{R}^n$. Since $\mathbf{q}$ has no task-space error, $\mathbf{CJq} = \mathbf{0}$. Hence, $f(q)$ maps $\mathbf{q}$ to itself as follows: $f(\mathbf{q}) = \mathbf{q} - \mathbf{J}^\dagger\mathbf{CJq} = \mathbf{q} - \mathbf{0} = \mathbf{q}$. ■

In fact, there exists an entire subspace $U = \{\mathbf{p}\}$ of $\mathbb{R}^n$, such that $f(\mathbf{p}) = \mathbf{q}$. This subspace is defined as $U = \{\mathbf{p} \in \mathbb{R}^n \,|\, \mathbf{p} = \mathbf{q} + \mathbf{J}^\dagger\mathbf{C}\Delta\mathbf{x}\}$, where $\Delta\mathbf{x}$ is any vector in task space.

Lemma 9.2: The FR function f is continuous.

Proof: Equation (20) defines $f(\mathbf{p}) = \mathbf{Ap}$, where $\mathbf{A} = (\mathbf{I} - \mathbf{J}^\dagger\mathbf{CJ})$ is a constant matrix. Hence, f satisfies the following conditions for a linear map.

1) $f(\mathbf{p}+\mathbf{p}') = \mathbf{A}(\mathbf{p}+\mathbf{p}') = \mathbf{Ap} + \mathbf{Ap}' = f(\mathbf{p}) + f(\mathbf{p}')$.
2) $f(c\mathbf{p}) = \mathbf{A}(c\mathbf{p}) = c\mathbf{Ap} = cf(\mathbf{p})$.

Since f is a linear map with a finite dimensional domain $\mathbb{R}^n$ and range $\mathcal{M}$, we conclude that f is continuous [32]. ■

Theorem 9.3: For any ϵ, let Q be the m-dimensional ϵ-neighborhood of any configuration $\mathbf{q}$ in $\mathcal{M}$ s.t. $0 < |\mathbf{q}' - \mathbf{q}| < \epsilon$ for all $\mathbf{q}' \in Q$. Then, for some $\mathbf{p}$ in $\mathbb{R}^n$, there exists P, an n-dimensional δ-neighborhood of $\mathbf{p}$ in $\mathbb{R}^n$, such that $0 < |\mathbf{p}' - \mathbf{p}| < \delta$, and $f(\mathbf{p}') \in Q$ for all $\mathbf{p}' \in P$.

Proof: This conclusion follows directly from Lemmas 9.1 and 9.2. Surjectivity of f ensures the existence of $\mathbf{p}$ for any choice of $\mathbf{q}$. The rest of the theorem follows from the Cauchy definition of a continuous function. ■

Theorem 9.3 distinguishes the proposed sampling strategy from local gradient-planning methods. Under the assumption of local linearity between joint and task coordinates, it ensures that uniform joint-space sampling followed by FR retraction yields a nonzero probability to sample any patch Q on the constraint manifold. Probabilistically complete joint-space planners retain their global-planning properties when using FR retraction.

Our result is particularly important for robots with redundant joints. Consider the prismatic robot in Fig. 12(a) and (b) with three joints $\mathbf{q}_1$, $\mathbf{q}_2$, and $\mathbf{q}_3$. The task space has 2 DOF, x and y for the end-effector and a constraint, $x = 0$, indicated by the dashed lines. This robot may use the redundancy in $\mathbf{q}_1$ and $\mathbf{q}_3$ to avoid obstacles. Fig. 12(c) shows that sampling of $\mathbf{q}$ followed by FR retraction maps distinct lines in the space of $\mathbf{q}_1$ and $\mathbf{q}_3$ to distinct valid configurations in robot joint space. Furthermore, for any ϵ-neighborhood of configurations on the dashed line, there exists a volume of configurations in joint space that maps to it to yield a nontrivial probability that the planner would detect the solution. Our proof generalizes this intuition to robots in n-dimensions.

X. Extensions

The previous sections introduced and analyzed a set of algorithms for joint-space planning with workspace constraints. Not only these methods are efficient and theoretically sound, but they are also easily extended to common robotics challenges.

A. Unilateral Constraints

The motion-constraint vector $\mathbb{C}$ represents bilateral constraints to prohibit positive and negative coordinate changes. Suppose a constraint requires a coordinate to remain within an interval or to change 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. The latter option specifies $c_i = 1$, or $c_i = 0$ for coordinate i, which was based on whether or not the sampled configuration violates the boundary. This choice will generate more valid samples, yet it may bias these samples toward the constraint bounds.

Alternatively, methods such as [33] and [34] can direct robot motion and prevent close interaction with the unilateral constraint bound.

B. Alternative Planning Strategies

The algorithms evaluated in Section VIII were all based on the single-query RRT algorithm. However, both RGD and FR are well-suited for use in PRMs among other multiquery algorithms. As long as the constraint distance metric is well-defined throughout the space, arbitrary configurations can be displaced toward the constraint manifold. All three algorithms can be used to connect PRM samples.

The generality of our methods extends to the use of heuristics, such as RRT-Connect during search [31]. Since the only modification to the RRT algorithm is in the NEW_CONFIG method, variants of RRT are trivially extended to include the proposed strategies.

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 from a composite constraint vector $\mathbb{C}_M$. Let $\mathbb{C}_i$ represent the constraint vector for task i. We then have

$$\mathbb{C}_M = [\,\mathbb{C}_1^T \mid \mathbb{C}_2^T \mid \cdots \mid \mathbb{C}_n^T\,]^T. \tag{22}$$

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

$$\mathbf{J}_M = [\,\mathbf{J}_1 \quad \cdots \quad \mathbf{J}_n\,]^T. \tag{23}$$

The computational methods to sample remain unchanged.

D. Abstract End-Effectors

The generalization of task definitions also extends to abstract end-effectors. Section IV-A displaced the end-effector frame to the grasped object. In fact, any coordinate that is defined as a function of one or more robot links can be constrained. An important example is the robot center of mass (COM). The COM position is a linear combination of the positions $\mathbf{p}_{m\,i}$ of the individual link masses m_i. For a total mass M

$$\mathbf{p}_{\text{COM}} = \frac{1}{M}\sum_i m_i \mathbf{p}_{m\,i}. \tag{24}$$

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

$$\mathbf{J}^0_{\text{COM}} = [\, \mathcal{J}_{P_1} \quad \dots \quad \mathcal{J}_{P_n} \,] \tag{25}$$

$$\mathcal{J}_{P_i} = \begin{cases} \left(\dfrac{1}{M}\displaystyle\sum_{j=i}^{n} m_j\right)\mathbf{z}_{i-1} & \text{(prismatic)} \\ \dfrac{1}{M}\displaystyle\sum_{j=i}^{n} m_j(\mathbf{z}_{i-1}\times(\mathbf{p}_{m\,j}-\mathbf{p}_{i-1})) & \text{(revolute).} \end{cases}$$

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

XI. Conclusion

We have presented a unified representation for task-space constraints in the context of joint-space motion planning. We described three algorithms for task-constrained sampling in joint space. Our comparison of the algorithms indicated that FR is typically faster and significantly more invariant to step size and error tolerance than alternative techniques. Finally, we generalized our approach to various constraint definitions and algorithms.

Our experimental findings regarding the efficiency of Jacobian-based algorithms also contribute to plan execution. To compute the Jacobian during planning allows us to measure the manipulability of sampled configurations: $\det(\mathbf{J}\mathbf{J}^T)^{1/2}$ [35]. Maintenance of a manipulability threshold permits stable use of local compliance or impedance control and allows for error while following a computed path.

Many facets of task-constrained planning remain for future investigation. For instance, Section X applies our method with multiple hard constraints. Soft constraints may lead to results in biasing motion plans toward desirable robot postures. Task projection into the null space of $\mathbf{J}^\dagger$ could be used to prioritize constraints [36], [37].

Appendix

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

Here, we provide a transformation from error in homogeneous coordinates to error in fixed-axis coordinates. First, we map homogeneous transformation matrices to fixed-axis coordinates. Translation is equivalent to the last column of the matrix. For $T_{a,b}$ to represent the value at row a, column b of T

$$\psi = \text{atan2}(T_{3,2}, T_{3,3}), \quad \theta = -\text{asin}(T_{3,1}), \quad \phi = \text{atan2}(T_{2,1}, T_{1,1}).$$

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 summing 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 $\mathbf{E}_\omega^{-1}(q)$

$$\omega = \mathbf{E}_\omega^{-1}(q)\,[\dot{\psi} \quad \dot{\theta} \quad \dot{\phi}]^T. \tag{27}$$

By inverting $\mathbf{E}_\omega^{-1}(q)$, we find $\mathbf{E}_\omega(q)$, which is a matrix that maps angular velocities to fixed-axis velocities. Adjoining the identity matrix for linear velocities, the following equation 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. Stilman, "Task constrained motion planning in robot joint space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2007, pp. 3074–3081.

[2] J. C. Latombe, *Robot Motion Planning*. Norwell, MA: Kluwer, 1991.

[3] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, no. 6, pp. 418–432, Jun. 1981.

[4] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Int. J. Robot. Res.*, vol. 3, no. 1, pp. 43–53, Feb. 1987.

[5] J. Ahuactzin and K. Gupta, "The kinematic roadmap: A motion planning based global approach for inverse kinematics of redundant robots," *IEEE Trans. Robot. Autom.*, vol. 15, no. 4, pp. 653–669, Aug. 1999.

[6] Z. Guo and T. Hsia, "Joint trajectory generation for redundant robots in an environment with obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1990, pp. 157–162.

[7] G. Oriolo, M. Ottavi, and M. Vendittelli, "Probabilistic motion planning for redundant robots along given end-effector paths," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2002, pp. 1657–1662.

[8] G. Oriolo and C. Mongillo, "Motion planning for mobile manipulators along given end-effector paths," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 2154–2160.

[9] D. P. Martin, J. Baillieul, and J. M. Hollerbach, "Resolution of kienmatic redundancy using opetimization techniques," *IEEE Trans. Robot. Autom.*, vol. 5, no. 4, pp. 529–533, Aug. 1989.

[10] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *J. Intell. Robot. Syst.*, vol. 3, pp. 201–212, 1990.

[11] S. Seereeram and J. T. Wen, "A global approach to path planning for redundant manipulators," *IEEE Trans. Robot. Autom.*, vol. 11, no. 1, pp. 152–160, Feb. 1995.

[12] A. McLean and S. Cameron, "The virtual springs method: Path planning and collision avoidance for redundant manipulators," *Int. J. Robot. Res.*, vol. 15, pp. 300–319, 1996.

[13] O. Brock, O. Khatib, and S. Viji, "Task-consistent obstacle avoidance and motion behavior for mobile manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 388–393.

[14] O. Khatib, L. Sentis, J. Park, and J. Warren, "Whole body dynamic behavior and control of human-like robots," *Int. J. Humanoid Robot.*, vol. 1, pp. 29–44, 2004.

[15] L. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[16] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Comput. Sci. Dept., Iowa State University, Ames, IA, Tech. Rep. 98-11, 1998.

[17] S. M. LaValle and J. J. Kuffner, "Rapidly exploring random trees: Progress and prospects," in *Proc. WAFR*, 2000, pp. 1–19.

[18] S. M. LaValle, J. Yakey, and L. E. Kavraki, "A probabilistic roadmap approach for systems with closed kinematic chains," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1999, pp. 1671–1676.

[19] L. Han and N. Amato, "A kinematics-based probabilistic roadmap method for closed chain systems," in *Proc. Workshop Algorithmic Found. Robot.*, 2000, pp. 1–13.

[20] J. Cortes, T. Simeon, and J. P. Laumond, "A random loop generator for planning the motions of closed kinematic chains with prm methods," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 2141–2146.

[21] X. Tang, S. Thomas, and N. M. Amato, "Planning with reachable distances: Fast enforcement of closure constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 2694–2699.

[22] X. Tang, S. Thomas, and N. M. Amato, "Planning with reachable distances," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2008, pp. 1–16.

[23] J. J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Motion planning for humanoid robots under obstacle and dynamic balance constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2001, pp. 692–698.

[24] Z. Yao and K. Gupta, "Path planning with general end-effector constraints: Using task space to guide configuration space search," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2005, pp. 1875–1880.

[25] Z. Yao and K. Gupta, "Path planning with general end-effector constraints," *Robot. Auton. Syst.*, vol. 55, no. 4, pp. 316–327, 2007.

[26] M. Stilman, J. Schamburek, J. Kuffner, and T. Asfour, "Manipulation planning among movable obstacles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 3327–3332.

[27] L. Sciaviccio and B. Siciliano, *Modeling and Control of Robot Manipulators*. New York: McGraw-Hill, 1996.

[28] T. N. E. Greville, "Some applications of the pseudoinverse of a matrix," *SIAM Rev.*, vol. 2, pp. 15–22, 1960.

[29] J. Yakey, S. LaValle, and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Trans. Robot. Autom.*, vol. 17, no. 7, pp. 951–958, Dec. 2001.

[30] A. Atramentov and S. M. LaValle, "Efficient nearest neighbor searching for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 1–6.

[31] J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proc. Workshop Algorithmic Found. Robot.*, 2000, pp. 995–1001.

[32] P. R. Halmos, *Finite-Dimensional Vector Spaces*. New York: Springer-Verlag, 1974.

[33] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1987, vol. 4, pp. 1152–1159.

[34] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J. P. Laumond, "A local collision avoidance method for non-strictly convex objects," in *Proc. Robot.: Sci. Syst. Conf.*, 2008, pp. 1–8.

[35] T. Yoshikawa, "Manipulability of robotic mechanisms," *Int. J. Robot. Res.*, vol. 4, pp. 3–9, 1985.

[36] H. Hanafusa, T. Yoshikawa, and Y. Nakamura, "Analaysis and control of articulated robot with redundancy," in *Proc. IFAC, 8th Triennal World Congr.*, 1981, vol. 4, pp. 1927–1932.

[37] R. Boulic and P. Baerlocher, "Task-priority formulations for the kinematic control of highly redundant atriculated structures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1998, pp. 323–329.

Optimal Object Configurations to Minimize the Positioning Error in Visual Servoing

Graziano Chesi, *Senior Member, IEEE*

***Abstract*—Image noise unavoidably affects the available image points that are used in visual-servoing schemes to steer a robot end-effector toward a desired location. As a consequence, letting the image points in the current view converge to those in the desired view does not ensure that the camera converges accurately to the desired location. This paper investigates the selection of object configurations to minimize the worst-case positioning error due to the presence of image noise. In particular, a strategy based on linear matrix inequalities (LMIs) and barrier functions is proposed to compute upper and lower bounds of this error for a given maximum error of the image points. This strategy can be applied to problems such as selecting an optimal subset of object points or determining an optimal position of an object in the scene. Some examples illustrate the use of the proposed strategy in such problems.**

***Index Terms*—Image noise, positioning error, visual servoing.**

I. Introduction

Visual servoing consists of steering a robot end-effector toward a desired location by exploiting, in a closed-loop fashion, the information provided by a vision system. Typically, the vision system is a camera mounted on the robot end-effector that observes the scene. The robot is controlled in order to make the projections of some object points in the current view converge to the projections of the same object points in the desired view, which have previously been recorded. Various methods have been proposed to accomplish this task, such as the pioneering image-based visual servoing (IBVS) [13] and position-based visual servoing (PBVS) [21]. Other methods use feedback errors, which are defined in both image and 3-D domains [18], partition of the degrees of freedoms [9], global motion plan via navigation functions [10], control invariant with respect to intrinsic parameters [17], switching control to ensure the visibility constraint [12], and path-planning to take various constraints into account [3]; see also [6] and references therein.

In all these methods, the robot control is terminated whenever the image points in the current view reach the corresponding ones in the desired view. However, this condition does not guarantee that the robot end-effector has accurately reached the sought desired location because the available image points in real visual-servo systems are unavoidably affected by image noise. This means that a positioning error unavoidably occurs.

This paper addresses this problem, in particular, investigating the worst-case robot positioning error that is introduced by image noise. Specifically, a strategy to compute upper and lower bounds of this error for a given maximum error of the image points is proposed. This strategy is based on the construction of suitable optimization problems with linear matrix inequalities (LMIs) and barrier functions that can readily be solved by using existing tools. Then, these bounds

Manuscript received July 28, 2009; revised November 30, 2009; accepted March 15, 2010. Date of publication April 12, 2010; date of current version June 9, 2010. This paper was recommended for publication by Associate Editor D. Kragic and Editor G. Oriolo upon evaluation of the reviewers' comments. This work was supported in part by the Research Grants Council of Hong Kong under Grant HKU711208E. This paper was presented in part at the International Conference on Robotics and Automation, Kobe, Japan, 2009.

The author is with the Department of Electrical and Electronic Engineering, University of Hong Kong, Hong Kong (e-mail: chesi@eee.hku.hk; http://www.eee.hku.hk/˜chesi).

Digital Object Identifier 10.1109/TRO.2010.2046071