Manipulation planning: building paths on constrained manifolds

Joseph Mirabel, Florent Lamiraux

Manipulation planning: building paths on constrained manifolds

Joseph Mirabel[1,2] and Florent Lamiraux[1,2]

Abstract— **Constrained motion planning and Manipulation planning, for generic non-linear constraints, highly rely on the ability of solving non-linear equations. The Newton-Raphson method, often used in this context, is discontinuous with respect to its input and point wise path projection can lead to discontinuous constrained paths.**

The discontinuities come from the pseudo-inverse involved at each iteration of the Newton-Raphson algorithm. An interval of continuity for one iteration is derived from an upper bound of the norm of the Hessian of the constraints. Such a bound is easy to compute for constraints involving joint positions and orientations.

We introduce two path projection algorithms. They provide a certificate of continuity of the Newton-Raphson iteration function along the path. The algorithms compare faster with the *Recursive Hermite Projection* on several problems, while having a stronger guaranty.

I. Introduction

Manipulation planning is known to be a difficult problem for several reasons. First, the geometrical structure of the problem is complex: the search is usually performed in the composite configuration space, *i.e.* the Cartesian product of the configuration spaces of the robots and of the objects. The admissible subspace of the composite configuration space, *i.e.* a union of submanifold defined by constraints (placement of objects in stable positions, grasp of objects by grippers). Moreover, in those sub-manifolds, motions are additionally constrained, thus defining foliations of the sub-manifolds. Second, the geometrical structure has to be translated into a graph of states that defines a discrete structure in a continuous problem. Exploring the graph of states implies the choice of transitions between states that adds parameters to the exploration algorithms. The efficiency of exploration algorithms is then very sensitive to parameter tuning. Third, manipulation constraints are diverse and difficult to express in a way both general and efficient.

Recently, we have proposed a formulation of the manipulation planning problem based on implicit numerical constraints of the form $\mathbf{f}(\mathbf{q}) = 0$ where $\mathbf{f}$ is a differentiable mapping from the composite configuration space to a finite-dimensional vector space [1]. To the best of our knowledge, this formulation is the most general ever proposed, and can express constraints as diverse as

- grasping an object, with possible free degrees of freedom (DOF) in the grasps (for cylindrical objects for instance),
- placement of an object on a bounded flat surface,
- quasi-static equilibrium for a humanoid robot,
- and most importantly, any combination of the above.

The above constraints have been implemented and are used by a manipulation planning algorithm in the software platform HPP that we have been developing for the past two years [2]. The core of this manipulation planning platform is thus the notion of implicit numerical constraint. Building paths that satisfy the constraints at any time is based on numerical resolution of those constraints (we also use the word *projection* since we project an initial guess onto the solution sub-manifold) and raises tricky continuity issues when projecting a path on a sub-manifold defined by numerical constraints.

The main contributions of this paper are

- to formulate the problem of path projection in a rigorous way,
- to propose two algorithms that project a linear interpolation on a sub-manifold with a certificate on the continuity of the projection,
- to provide a mathematical proof of the above certificate.

The paper is organized as follows. In Section II, we provide a short state of the art in manipulation planning, showing how original our approach is. In Section III, we give some useful definitions. Section IV describes the main theoretical result and our two path projection algorithms. Finally, the two algorithms are compared to related work in simulations.

II. Related work

Manipulation planning has been first addressed in the 1980's [3], [4], [5] and has given rise to a lot of research work in the 1990's [6], [7], [8]. The first use of roadmap-based random sampling method for the problem of manipulation planning has been reported in [9]. In this later work, constraints are expressed in an explicit way: position of the object computed from position of gripper for grasp positions, position of the gripper for given position of the object computed by inverse kinematics. As such, this pioneering work is not directly extendible to more general problems like humanoid robot in quasi-static equilibrium, or robot arm with more than 6 degrees of freedom. [10] propose an implementation of Navigation Among Movable Obstacles (NAMO) for a humanoid robot manipulating objects rolling on the ground. The geometry of the robot is simplified to a cylinder and the 3D configuration space is discretized. A high level planner searches a path between the initial and goal

*This work has been partially supported by the national PSPC-Romeo 2 project, has received funding from the European CommunityâĂŹs Seventh Framework Programme (FP7/2007-2013) under grant agreement nÂř 609206 and nÂř 608849.

[1]CNRS, LAAS, 7 avenue du colonel Roche, F–31400 Toulouse, France
[2]Univ de Toulouse, LAAS, F–31400 Toulouse, France

configuration that may collide with movable obstacles. Then a manipulation planner plans motion to move objects out of the way. The algorithm is demonstrated on a humanoid robot HRP2. They reduced manipulation planning to a 2D problem. [11] addresses the specific case of dual arm manipulation planning. As in [9], constraints are solved by inverse kinematics. [12] proposes a manipulation planning framework taking into account constraints beyond the classical grasp and placement constraints. As in our case, they need to project configurations and paths on manifolds defined by non-linear constraints. Path projection is however performed by discretization and the continuity issue is not discussed.

The *Recursive Hermite Projection* (RHP) [13] addresses the problem of generating C^1 paths that satisfy a set of non-linear constraints. We consider random exploration of the configuration space. As such, we only consider continuity and not differentiability. We prefer to explore the configuration space of the system and to address differentiability in a post-processing step. This approach is known to be more efficient than kinodynamic motion planning that explores the state space of the system and returns differentiable solutions. As such, the RHP is thus not a suitable solution. Section IV-C gives a more detailed comparison between our method and the RHP.

III. Notation and definitions

We consider a manipulation problem defined by a set of robots and objects. We denote by $\mathcal{C}$ the Cartesian product of the configuration spaces of the robots and of the objects. If the number of robots is 1 and the number of objects is 0, the problem becomes a classical path planning problem. Even in this case, the robot may be subject to non-linear constraints. For instance, static equilibrium constraint for a humanoid robot standing on the ground, or for a wheeled mobile robot moving on a non-flat terrain.

We give the following definitions.

- **Path** p : continuous mapping from an interval $I \subset \mathbb{R}$ to $\mathcal{C}$,
- **Constraint** $\mathbf{f}$: C^1 mapping from $\mathcal{C}$ to vector space $\mathbb{R}^m$, where m is a positive integer. We say that configuration $\mathbf{q} \in \mathcal{C}$ satisfies the constraint iff
$$\mathbf{f}(\mathbf{q}) = 0$$
- **Projector on constraint** $\mathbf{f}$: mapping $proj$ from a subset D_{proj} of $\mathcal{C}$ to $\mathcal{C}$ such that
$$\forall \mathbf{q} \in D_{proj}, \quad \mathbf{f}(proj(\mathbf{q})) = 0.$$

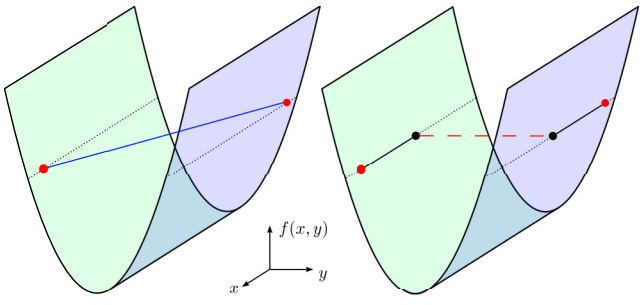


Fig. 1. This 2D example, where (x, y) are the configuration parameters, shows the graph of $\mathbf{f}((x, y)) = y^2 - 1$. The 2 dotted horizontal line are the solutions of $\mathbf{f}((x, y)) = 0$. The 2 red circles are two configurations satisfying $\mathbf{f}(\mathbf{q}) = 0$. On the left, the blue line is $\mathbf{straight}(\mathbf{q}_0, \mathbf{q}_e)$ and on the right, the black solid line is its pointwise projection. The discontinuity is highlighted by the black circles and the red dashed line.

A. Path planning on constrained manifold

When solving a path planning problem where the robot is subject to a numerical constraint, we make use of an operator called *steering method* that takes as input two configurations satisfying the constraint and that returns (in case of success) a path satisfying the constraint and linking the end configurations.

$$\mathcal{SM} : \begin{array}{ccc} \mathcal{C} \times \mathcal{C} \times C^1(\mathcal{C}, \mathbb{R}^m) & \rightarrow & C^1([0,1], \mathcal{C}) \\ (\mathbf{q}_0, \mathbf{q}_e, \mathbf{f}) & \mapsto & p \end{array}$$
$$\text{such that } \forall t \in [0,1], \mathbf{f}(p(t)) = 0$$

We denote by $\mathbf{straight}$ the constraint-free steering method that returns the linear interpolation between the input configurations.

From an implementation point of view, we could discretize the linear interpolation between $\mathbf{q}_0$ and $\mathbf{q}_e$ into N steps, project each sample configuration on constraint $\mathbf{f}$ and make the steering method return linear interpolations between projected sample configurations. However, the point wise projection has two drawbacks. First, in some cases, for instance in Figure 1, it introduces a discontinuity. And second, the resulting path does not satisfy the constraint between samples.

As for collision-checking, discretizing constraints along paths raises many issues, mainly

- discretization step needs to be chosen for each application,
- some algorithms that assume that constraints are satisfied everywhere may fail because the assumption is not satisfied.

Our steering method instead applies the constraint at evaluation:

$$\mathcal{SM}(\mathbf{q}_0, \mathbf{q}_e, \mathbf{f})(t) = proj(\mathbf{straight}(\mathbf{q}_0, \mathbf{q}_e)(t))$$

where $proj$ is a projector on $\mathbf{f}$.

IV. Path projection algorithm

In this section, we derive a continuity condition of the Newton-Raphson (NR) algorithm. Then, we introduce two algorithms to check for path continuity.

The NR algorithm iteratively updates the robot configuration so as to decrease the norm of the constraints value $\mathbf{f}(\mathbf{q})$. Let $\alpha > 0$ and $P_\alpha \in \mathcal{F}(\mathcal{C}, \mathcal{C})$ be the NR iteration function:

$$P_\alpha(\mathbf{q}) = \mathbf{q} - \alpha \times \mathbf{J}(\mathbf{q})^\dagger \times \mathbf{f}(\mathbf{q}) \tag{1}$$

where $A^\dagger$ is the Moore-Penrose pseudo-inverse of A and $\mathbf{J}(\mathbf{q})$ is the Jacobian matrix of $\mathbf{f}$ in $\mathbf{q}$. $P_\alpha(\mathbf{q})$ is the configuration obtained after one iteration of the NR algorithm, starting at $\mathbf{q}$.

For a given sequence $(\alpha_n) \in]0,1]^{\mathbb{N}}$ and a given numerical tolerance $\epsilon > 0$, let $\mathcal{P}_N(\mathbf{q}) = P_{\alpha_N}(\cdots(P_{\alpha_0}(\mathbf{q})))$. The projection of a configuration $\mathbf{q}$ is $\mathcal{P}_N(\mathbf{q})$ where N is such that:

- $\forall i \in [\![0, N]\!], P_{\alpha_i}(\cdots(P_{\alpha_0}(\mathbf{q}))) \geq \epsilon$,
- $\mathcal{P}_N(\mathbf{q}) < \epsilon$.

Note that the projection is not always defined as N might not exist.

Denoting $\mathcal{B}(\mathbf{q}, r) = \{\tilde{\mathbf{q}} \in \mathcal{C}, ||\tilde{\mathbf{q}} - \mathbf{q}||_2 < r\}$. The continuity of P_α is expressed as follows.

Lemma 4.1 (Continuity of the NR iteration function): Let $\mathbf{f} \in \mathcal{C}^1(\mathcal{C}, \mathbb{R}^m)$. Let $\mathbf{J}(\mathbf{q})$ be its Jacobian and $\sigma(\mathbf{q})$ be the smallest non-zero singular value of $\mathbf{J}(\mathbf{q})$. Finally, let $r = \max_{\mathbf{q} \in \mathcal{C}}(rank(\mathbf{J}(\mathbf{q})))$.

If $\mathbf{J}$ is a Lipschitz function, of constant K, then, $\forall \mathbf{q} \in \mathcal{C}$

$$rank(\mathbf{J}(\mathbf{q})) = r \Rightarrow P_\alpha \text{ is } C^0 \text{ on } \mathcal{B}\left(\mathbf{q}, \tfrac{\sigma(\mathbf{q})}{K}\right)$$

A. Proof of continuity of the NR iteration function

This section provides a proof of Lemma 4.1. $\mathbf{f}$ is continuously differentiable, K is a Lipschitz constant of its Jacobian, and $r = \max_{\mathbf{q}}(rank(\mathbf{J}(\mathbf{q})))$ is known.

As $\mathbf{f}$ is continuously differentiable, P_α is continuous where the pseudo-inverse application is continuous. The first part of the proof reminds some continuity condition of the pseudo-inverse. The second part proves that the latter condition is satisfied on the interval of Lemma 4.1.

1) Condition of continuity of the pseudo-inverse: Let $\mathbf{q}$ be a *regular point*, *i.e.* $rank(\mathbf{J}(\mathbf{q})) = r$. As the set of *regular points* is open [14] and $\mathbf{J}$ is continuous, there exists a neighborhood $\mathcal{U}$ of $\mathbf{q}$ where the rank of $\mathbf{J}$ is constant. The continuity of the Moore-Penrose pseudo inverse can be expressed as follows [15].

Theorem 4.1 (Continuity of the pseudo inverse): If $(A_n) \in (\mathcal{M}_{m,d})^{\mathbb{N}}, A \in \mathcal{M}_{m,d}$ and $A_n \mapsto A$, then

$$A_n^\dagger \mapsto A^\dagger \Leftrightarrow \exists n_0, \forall n \geq n_0, rank(A_n) = rank(A)$$

Theorem 4.1 proves that $\mathbf{J}^\dagger$ is a continuous function of $\mathbf{q}$ on $\mathcal{U}$. In the following section, we prove that $\mathcal{U} = \mathcal{B}\left(\mathbf{q}, \frac{\sigma}{K}\right)$ is a suitable neighborhood.

2) Interval of continuity of the pseudo-inverse: The norm on $\mathcal{M}_{m,n}(\mathbb{R})$ we consider is the Frobenius norm (L2-norm), denoted $|||.|||_F$.

Theorem 6 of [16], restricted to the Frobenius norm, is:

Theorem 4.2 (Mirsky): If $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ and $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_n$ are the singular values of two matrices of the same size, $\mathbf{B}$ and $\tilde{\mathbf{B}}$, then

$$|||diag(\tilde{\sigma}_i - \sigma_i)|||_F \leq |||\tilde{\mathbf{B}} - \mathbf{B}|||_F$$

.

Lemma 4.2: Let $(\mathbf{J}, \mathbf{dJ}) \in \mathcal{M}^2_{m \times d}$ and σ be the smallest non-zero sigular value of $\mathbf{J}$. Then,

$$|||\mathbf{dJ}|||_F < \sigma \Rightarrow rank(\mathbf{J}) \leq rank(\mathbf{J} + \mathbf{dJ})$$

Proof: Let p, resp. q, be $rank(\mathbf{J})$, resp. $rank(\mathbf{J} + \mathbf{dJ})$. Let $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p > 0$, resp. $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_q > 0$, be the non-zero singular values of $\mathbf{J}$, resp. $\mathbf{J} + \mathbf{dJ}$. We apply Theorem 4.2 with $\mathbf{B} = \mathbf{J}$ and $\tilde{\mathbf{B}} = \mathbf{J} + \mathbf{dJ}$.

$$\begin{aligned} |||\mathbf{dJ}|||_F < \sigma_p &\Rightarrow |||diag(\tilde{\sigma}_i - \sigma_i)|||_F < \sigma_p \\ &\Rightarrow \forall i \leq p, \tilde{\sigma}_i > \sigma_i - \sigma_p \\ &\Rightarrow \forall i \leq p, \tilde{\sigma}_i > 0 \\ &\Rightarrow p \leq q \end{aligned}$$

■

Note that the ball has to be open. At this point, we have an interval for the Jacobian in which the rank does not decrease. We use the Lipschitz constant K to have an interval in the configuration space.

$$\forall (\mathbf{q}, \tilde{\mathbf{q}}) \in \mathcal{C}^2, |||\mathbf{J}(\tilde{\mathbf{q}}) - \mathbf{J}(\mathbf{q})|||_F \leq K||\tilde{\mathbf{q}} - \mathbf{q}||_2$$

Let $\mathbf{q} \in \mathcal{C}$ and σ be the smallest non-zero singular value of $\mathbf{J}(\mathbf{q})$. Then,

$$\begin{aligned} \tilde{\mathbf{q}} \in \mathcal{B}\left(\mathbf{q}, \tfrac{\sigma}{K}\right) &\Rightarrow |||\mathbf{J}(\tilde{\mathbf{q}}) - \mathbf{J}(\mathbf{q})|||_F \leq K||\tilde{\mathbf{q}} - \mathbf{q}||_2 < \sigma_p \\ &\Rightarrow rank(\mathbf{J}(\tilde{\mathbf{q}})) \geq rank(\mathbf{J}(\mathbf{q})) \end{aligned}$$

If $\mathbf{q}$ is a *regular point*, $rank(\mathbf{J}(\mathbf{q}))$ has rank $r = \max_{\mathbf{q}}(rank(\mathbf{J}(\mathbf{q})))$. Thus $\mathbf{J}(\tilde{\mathbf{q}})$ has a constant rank r on $\mathcal{B}\left(\mathbf{q}, \frac{\sigma}{K}\right)$. By Theorem 4.1, $\mathbf{J}(\mathbf{q})^\dagger$ is continuous. P_α is the composition of continuous functions so it is continuous on $\mathcal{B}\left(\mathbf{q}, \frac{\sigma}{K}\right)$.

This proves Lemma 4.1.

B. Algorithms

This section presents two path projection algorithms with continuity certificate. From an initial constrained path $\mathcal{SM}(\mathbf{q}_0, \mathbf{q}_e, \mathbf{f})$, the algorithm generates a set of interpolation points $(\mathbf{q}_1, \cdots, \mathbf{q}_n)$ where $\mathbf{f}(\mathbf{q}_i) = 0$ and n is decided by the algorithm. The resulting path is the concatenation of $\mathcal{SM}(\mathbf{q}_i, \mathbf{q}_{i+1}, \mathbf{f}), \forall i \in [0, n[$. When the algorithms succeed, $\mathbf{q}_n = \mathbf{q}_e$. When they fail to project a path, they return the longuest part along the path, starting at $\mathbf{q}_0$, that has been validated.

To benefit from the continuity interval of P_α, an upper bound of the norm of the Hessian of the constraints for constraints involving joint placements is computed [17]. This upper bound is a Lipschitz constant of the Jacobian. This method extends to constraints involving the center of mass (COM) of the robot as the COM is a weighed sum of joint positions. Lemma 4.1 ensures the output is a path along which the NR iteration function is continuous.

The maximum number of interpolation points on unit length paths N_{max} is set to 20 and the minimum interpolation distance λ_m is set to 0.001. These parameters ensure our algorithms terminate. When a limit is reached, the algorithm returns the left part of a path that has been successfully projected, as stated above.

1) Progressive projection: is presented in Alg 1 and depicted in Figure 2.

From $\mathbf{q}_0$, it builds a configuration satisfying the constraint, within the continuity interval of $\mathbf{q}_0$. The configuration is chosen towards $\mathbf{q}_e$ (Line 7). When $\mathbf{q}_e$ is within the continuity interval of $\mathbf{q}_0$ (Line 2), the algorithm succeeds.

Algorithm 1 Progressive continuous projection

1: **function** PROJECT($\mathbf{q}_0, \mathbf{q}_e, depth$)
▷ Continuously project the direct path $(\mathbf{q}_0, \mathbf{q}_e)$ onto the submanifold $\mathbf{f}(\mathbf{q}) = 0$
2: **if** $K \times ||\mathbf{q}_0 - \mathbf{q}_e||_2 < \sigma_r(\mathbf{q}_0)$ **then return** $(\mathbf{q}_0, \mathbf{q}_e)$
3: **if** $depth > N_{max} \times ||\mathbf{q}_0 - \mathbf{q}_e||_2$ **then return** $(\mathbf{q}_0)$
4: $\lambda \leftarrow \sigma_r(\mathbf{q}_0)/K$
5: **repeat**
6: **if** $\lambda < \lambda_m$ **then return** $(\mathbf{q}_0)$
7: $\mathbf{q} \leftarrow \mathcal{SM}(\mathbf{q}_0, \mathbf{q}_e, f)(\frac{\lambda}{||\mathbf{q}_0 - \mathbf{q}_e||_2})$
8: $\lambda \leftarrow \frac{\lambda}{2}$
9: **until** $K||\mathbf{q} - \mathbf{q}_0||_2 < \sigma_r(\mathbf{q}_0)$
10: **return** $\{\mathbf{q}_0\} \cup$ PROJECT($\mathbf{q}, \mathbf{q}_e, depth + 1$)

From 2a to 2b, the path is cut in two at distance λ from the start configuration. λ is gradually reduced so that the projected configuration is within the continuity ball of $\mathbf{q}_0$ (Lines 4-9). When $\lambda < \lambda_m$ (Line 6), the projection locally increases the distances more than $\sigma_r(\mathbf{q}_k)/(\lambda_m K)$. The path is considered discontinuous and the algorithm fails. If q is found in $\mathcal{B}\left(\mathbf{q}_0, \frac{\sigma(\mathbf{q}_0)}{K}\right)$, the left part satisfies the condition of Lemma 4.1. The right part then projected using the same procedure.

2) Global projection: method is presented in Alg 2 and depicted in Figure 3.

Algorithm 2 Global continuous projection

1: **function** PROJECT($\mathbf{q}_0, \mathbf{q}_e, \mathbf{f}$)
▷ Continuously project the direct path $(\mathbf{q}_0, \mathbf{q}_e)$ onto the submanifold $\mathbf{f}(\mathbf{q}) = 0$
2: $\mathbf{Q} \leftarrow (\mathbf{q}_0, \mathbf{q}_e)$
3: $repeat \leftarrow True$
4: **while** $repeat$ **do**
5: $repeat \leftarrow False$
6: **for all** $\mathbf{q}_k \in \mathbf{Q}$ **do**
7: **if** $||f(\mathbf{q}_k)||_2 > \epsilon$ **then**
8: $\mathbf{q}_k \leftarrow P_\alpha(\mathbf{q}_k)$
9: $repeat \leftarrow True$
10: **for all** Consecutive $\mathbf{q}_k, \mathbf{q}_{k+1} \in \mathbf{Q}$ **do**
11: **if** $\sigma_r(\mathbf{q}_k) < K\lambda_m$ **then**
12: $\mathbf{Q} \leftarrow (\mathbf{q}_0, \cdots, \mathbf{q}_k)$ **and break**
13: $d \leftarrow \sigma_r(\mathbf{q}_k) + \sigma_r(\mathbf{q}_{k+1})$
14: **if** $d < K \times ||\mathbf{q}_k - \mathbf{q}_{k+1}||_2$ **then**
15: $\mathbf{q} \leftarrow$ INTERPOLATE($\mathbf{q}_k, \mathbf{q}_{k+1}, \frac{\sigma_r(\mathbf{q}_k)}{K}$)
16: $\mathbf{Q} \leftarrow (\mathbf{q}_0, \cdots, \mathbf{q}_k, \mathbf{q}, \mathbf{q}_{k+1}, \cdots)$
17: $repeat \leftarrow True$
18: **if** LENGTH($\mathbf{Q}$) $> N_{max} \times ||\mathbf{q}_0 - \mathbf{q}_e||_2$ **then**
19: $\mathbf{Q}$.REMOVELASTELEMENT
20: **return** $\mathbf{Q}$

The algorithm starts by computing interpolation points along the straight path. Then, it works in two steps. First, the interpolation points are improved in order to decrease the constraint violation, by applying the NR iteration function (Line 8). Second, it checks whether the distance between each pair of consecutive interpolation points $(\mathbf{q}_k, \mathbf{q}_{k+1})$ is within the union of the two continuity balls (Line 14). If this check fails, a new interpolation point $\mathbf{q}$ is added at the border of the continuity ball of $\mathbf{q}_k$. Next iteration will consider the two consecutive points $(\mathbf{q}, \mathbf{q}_{k+1})$.

For clarity of the pseudo-code, we omitted to include a limit on the number of iterations of constraint violation reduction loops (Line 6). Such a limit must be integrated to avoid infinite loops due to local minimas. We set this limit to 40 in our implementation and the counter is reset whenever a interpolation point is added.

Figure 3 shows the path after some iterations. From 3b and 3c, the projection loop (Line 6) reduces the constraint violation point-wise. Between 3c and 3d, an interpolation point is added (Line 18).

C. Discussion

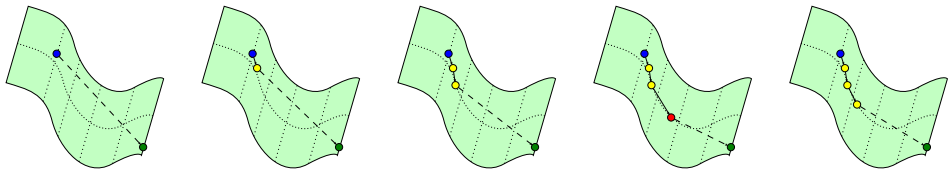
The two algorithms have the following guaranties. They provide a path with interpolation points satisfying the constraints. Moreover, they ensure that the NR iteration function is continuous along the lines connecting consecutive interpolation points. The piecewise straight interpolation is closer to constraint satisfaction than the input path and one iteration of NR is continuous. This leads to good chances to have the resulting path continuous. In practice, no discontinuity has been encountered.

Compared to our method, the RHP gives continuity, at the cost of being first, computationally less efficient, second, unable to return the continuously projected part of the path and third requires to introduce velocities. The efficiency of our method, compared to RHP, comes from the expected distances between interpolation points. Indeed, RHP generates a lot more interpolation points than us. The reason is the following. The distance between interpolation points is less than $\epsilon/K_{\mathbf{f}}$ where ϵ is the constraint satisfaction tolerance and $K_{\mathbf{f}}$ is a Lipschitz constant of the constraint. In our case, this distance is around $\sigma/K_{\mathbf{J}}$, where σ is the smallest singular value of the Jacobian and $K_{\mathbf{J}}$ is a Lipschitz constant of the Jacobian of the constraint. In part of the configuration space far from singularities, σ is orders of magnitude bigger that ϵ, set to 10^{-4} in our experiments. The comparison of RHP and our work in next section emphasizes. Next section comparison of both approaches regarding this result. This results are

V. SIMULATIONS

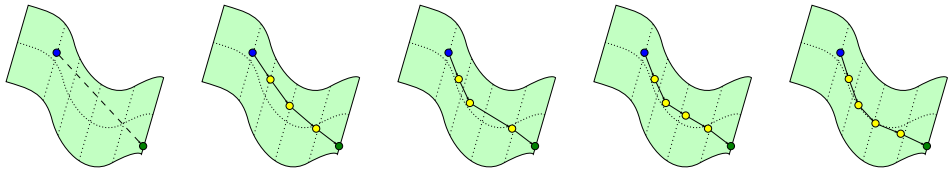
In this section, both algorithms and the RHP are compared to each other in two settings, each described in the two following paragraph. The benchmarks are run using the HPP software framework, in which the 3 algorithms have been implemented[1].

[1]https://github.com/jmirabel/hpp-core/tree/hermite_projection/src/path-projector



(a) (b) (c) (d) (e)

Fig. 2. Progressive projection method. The green surface is $\mathbf{f}(\mathbf{q}) = 0$. 2a shows the input path. Between 3b and 3c, the interpolation point is added because it is close enough from the last point. On 3d, the point is rejected because it is too far from the last point and λ is divided by two. It results in 3d and the interpolation point is finally added.



(a) (b) (c) (d) (e)

Fig. 3. Global projection method. The green surface is $\mathbf{f}(\mathbf{q}) = 0$. 3a shows the input path. Between 3b and 3c, each interpolation points has been updated to decrease the constraint violation. Between 3c and 3d, an interpolation point is added because the distance between two adjacent points is bigger than the threshold.

1) Quadratic problems: We first compare our two algorithms and the RHP for various parameter in the following problems.

- Circle: the configuration space is $[-1, 1]^2$, subject to constraint $f(x, y) = x^2 + y^2 - 1 = 0$. A Lipschitz constant of f is $M = 2\sqrt{2}$ and a Lipschitz constant of its Jacobian is $K = 2\sqrt{2}$. We project line segments between $(1, 0)$ and $(\cos\theta, \sin\theta)$ for $\theta \in [\pi/2, \pi]$. None of the algorithms were able to find a continuous path for the singular case $\theta = \pi$. The Global projection method did not need any interpolation points to return an answer.
- Parabola: the configuration space is $[-1, 1] \times [0, 2]$, subject to constraint $f(x, y) = y^2 - 1 = 0$. This constraint has two disjoint sets of solution: $y = -1$ and $y = 1$. Fig. 1 illustrates it. A Lipschitz constant of f is $M = 2$ and a Lipschitz constant of its Jacobian is $K = 2$. We project line segments between $(0, 1)$ and $(\tau, -1)$ for $\tau \in [0, 2]$. No continuous path can both connect any pair of these points and satisfy the constraints at all time. All the algorithms were able to detect the discontinuity.

Results are presented in Table I. The global projection method outperforms the progressive method on these quadratic problems.

Global proj.	Circle	Parabola
t_{avg}/t_{max} (µs)	16/90	201/231
$d_{min}/d_{avg}/d_{max}$(mm)	-	0.2/51/316
N_{ip}	0	10

Progressive proj.	Circle	Parabola
t_{avg}/t_{max} (µs)	78/134	173/207
$d_{min}/d_{avg}/d_{max}$ (mm)	284/462/512	1/71/316
N_{ip}	4.75	14

Recursive hermite proj.	Circle	Parabola
t_{avg}/t_{max} (ms)	503/900	0.017/0.03
$d_{min}/d_{avg}/d_{max}$ (µm)	50/75/100	-
N_{ip}	28946	0

TABLE I

QUADRATIC PROBLEMS BENCHMARKS. THE ROWS CORRESPONDS TO THE NUMBER OF INTERPOLATION POINTS N_{ip}, THE AVERAGE, MINIMUM AND MAXIMUM DISTANCE BETWEEN CONSECUTIVE WAYPOINTS d_{avg}, d_{min}, d_{max}, AND THE AVERAGE AND MAXIMUM COMPUTATION TIME OVER 10 RUNS t_{avg}, t_{max}.

2) Manipulation planning:

a) UR5: We constrain the end-effector of the UR5 robot along a line, its orientation being fixed. We project a motion where the robot must move along this line and switch between inverse kinematic solutions. Using [17], a Lipschitz constant of the constraint is $M = 6$ and a Lipschitz constant of its Jacobian is $K = 7.14$. Table II summarizes the obtained results for various line segment. The first part of the accompanying video shows one of the computed motions. Note that, in this case, we do not do any motion planning. To our best knowledge, it would not be possible to compute the same motions merely using inverse kinematics.

When the projection method returns a false negative, the longest validated part of the input path is returned. In the context of randomized motion planning, the high rate of false negatives of global methods does not block the search. The expected effect is a increase of the number of nodes generated.

b) Integration in a manipulation planner: The continuous projection can easily be integrated in randomized constrained motion planners. Path projection must be done before collision checking as it modifies the path. To ensure their validity, paths can be created in two steps. First, continuously project the straight interpolation onto the constraint satisfaction manifold. Optionally, keep one continuous end of the path. And second, check the projected path for collision.

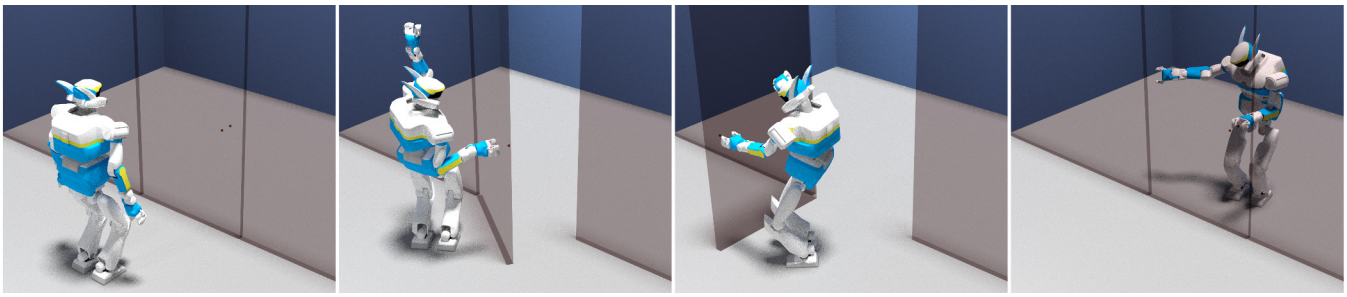


Fig. 4. HRP2 opening a door.

Projection method	Global	Progressive	RHP
t_{avg}/t_{max} (ms)	85/746	6.5/9.5	160/175
N_{ip}	123	72	3403
False negative	54%	0%	8%

TABLE II

RESULTS OF UR5 CASE. THE ROWS HAVE THE SAME MEANING AS IN TABLE I. THE NUMBER OF FALSE NEGATIVE CORRESPONDS TO THE RATIO OF REJECTED PATH OVER ALL TESTS, WHILE A CONTINUOUS PATH EXISTS.

The proposed algorithms have been integrated in a manipulation planner. We wish to compute a path for the HRP2 robot opening a door. The planning is split in two phases [18]. A quasi-static full-body motion for the sliding robot is first computed. Additionally to the manipulation rules, quasi-static constraints are taken into account. Then, the motion is post-processed to obtain a dynamically-feasible walking trajectory.

The accompanying video and Figure 4 shows the result of the first phase, obtained with and without continuous path projection. No optimization were run. As one can see, the motion without continuous path projection has several discontinuity. This demonstrates both the necessity to check motion continuity and that our algorithms perform as expected.

VI. CONCLUSION

This work has shown that it is possible to deal with generic non-linear implicit constraints and still have a certificate of continuity for motions projected inside the constraint satisfaction submanifold. Our main focus has been to derive a theoretical condition of continuity which can be exploited by a computer, and to design algorithms using this condition.

Efficiency of the proposed algorithms has not been the main focus and is left for future work. They can be improved by better organising the computation as computing singular values and evaluating the Newton-Raphson iteration function can be factorized.

REFERENCES

[1] Mirabel, J., Lamiraux, F.: Constraint graphs: Unifying task and motion planning for navigation and manipulation among movable obstacles. submitted to IROS 2016

[2] Mirabel, J., Tonneau, S., Fernbach, P., Seppälä, A.K., Campana, M., Mansard, N., Lamiraux, F.: Hpp: a new software for constrained motion planning. submitted to IROS 2016

[3] Mason, M.: Manipulation by grasping and pushing operations. PhD thesis, MIT, Artificial Intelligence Laboratory (1982)

[4] Cutkosky, M.: Robotic Grasping and Fine Manipulation. Kluwer, Boston (1985)

[5] Peshkin, M., Sanderson, A.: Planning robotic manipulation strategies for workpieces that slide. IEEE Transactions on Robotics and Automation **4**(5) (1988) 524–531

[6] Koga, Y., Latombe, J.C.: On multi-arm manipulation planning. In: International Conference on Robotics and Automation, San Diego (USA), IEEE (May 1994) 945–952

[7] Bicchi, A.: Hands for dexterous manipulation and powerful grasping. In Giralt, G., Hirzinger, G., eds.: International Symposium on Robotics Research. Springer, London (1996) 2–15

[8] Alami, R., Laumond, J., Siméon, T.: Two manipulation planning algorithms. In Goldberg, K., Halperin, D., Latombe, J.C., Wilson, R., eds.: Algorithmic Foundations of Robotics, Wellesley, MA, A K Peters, Ltd. (1995) 109–125

[9] Siméon, T., Laumond, J.P., Cortés, J., Sahbani, A.: Manipulation planning with probabilistic roadmaps. International Journal of Robotics Research **23**(7/8) (July 2004)

[10] Stilman, M., Nishiwaki, K., Kagami, S., Kuffner, J.J.: Planning and executing navigation among movable obstacles. Advanced Robotics **21**(14) (2007) 1617–1634

[11] Harada, K., Tsuji, T., Laumond, J.P.: A manipulation motion planner for dual-arm industrial manipulators. In: IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China (May 2014)

[12] Berenson, D., Srinivasa, S., Ferguson, D., Kuffner, J.: Manipulation planning on constraint manifolds. In: IEEE International Conference on Robotics and Automation. (May 2009)

[13] Hauser, K.: Fast interpolation and time-optimization on implicit contact submanifolds. In: Proceedings of Robotics: Science and Systems, Berlin, Germany (June 2013)

[14] Lewis, A.: Semicontinuity of rank and nullity and some consequences

[15] Rakočević, V.: On continuity of the moore-penrose and drazin inverses. Matematički Vesnik **49**(209) (1997) 163–172

[16] Stewart, M.: Perturbation of the {SVD} in the presence of small singular values. Linear Algebra and its Applications **419**(1) (2006) 53 – 77

[17] Schwarzer, F., Saha, M., Latombe, J.C. In: Exact Collision Checking of Robot Paths. Springer Berlin Heidelberg, Berlin, Heidelberg (2004) 25–41

[18] Dalibard, S., El Khoury, A., Lamiraux, F., Nakhaei, A., Taïx, M., Laumond, J.P.: Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach. The International Journal of Robotics Research **32**(9-10) (2013) 1089–1103