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 heuristical path projection algorithms and empirically show that they generate continuous constrained paths. The algorithms are integrated into a motion planning algorithm to empirically assert the continuity of the solution.**

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 submanifolds 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,

*This work has been partially supported by the national PSPC-Romeo 2 project and the ERC-ADV project 340050 Actanthrope.

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

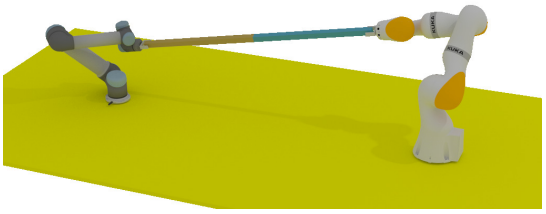


Fig. 1. UR5 and KUKA robots manipulate a beam.

- 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

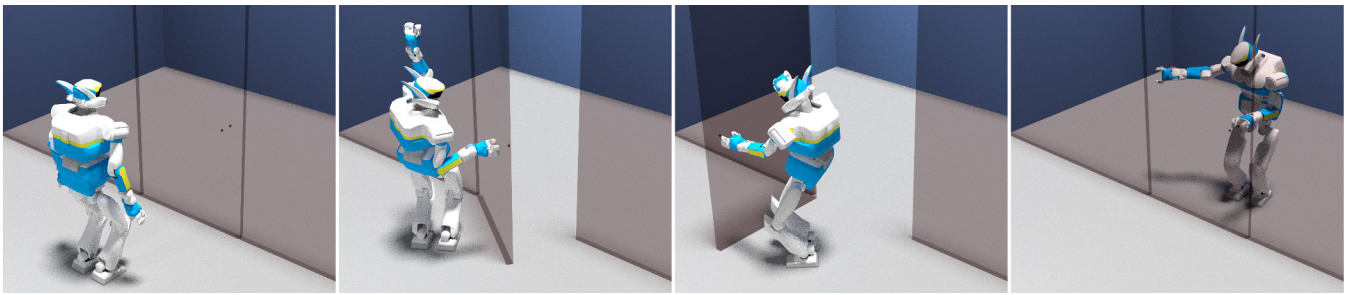


Fig. 2. HRP2 opening a door.

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 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
$$\forall \mathbf{q} \in D_{proj}, \quad \mathbf{f}(proj(\mathbf{q})) = 0.$$

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

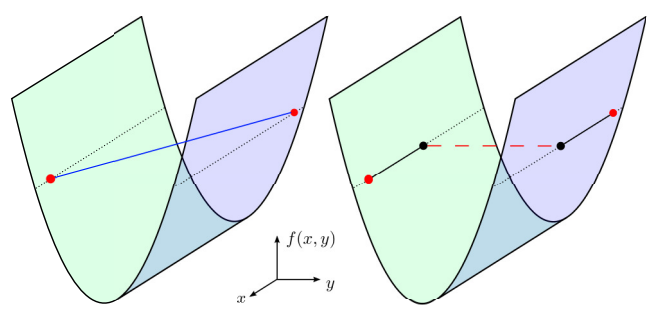


Fig. 3. This 2D example, where (x, y) are the configuration parameters, shows the graph of $\mathbf{f}((x,y)) = y^2 - 1$. The 2 dotted horizontal line are the solutions of $\mathbf{f}((x,y)) = 0$. The 2 red circles are two configurations satisfying $\mathbf{f}(\mathbf{q}) = 0$. On the left, the blue 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.

projected sample configurations. However, the point wise projection has two drawbacks. First, in some cases, for instance in Figure 3, 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 certificate of the Newton-Raphson (NR) algorithm. Then, we introduce two algorithms to generate paths satisfying this certificate.

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 IV.1 (*Continuity certificate*)**:** *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 the continuity certificate

This section provides a proof of Lemma IV.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 IV.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 IV.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 IV.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 IV.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 IV.2: *Let* $(\mathbf{J}, \mathbf{dJ}) \in \mathcal{M}^2_{m \times d}$ *and* σ *be the smallest non-zero singular 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 IV.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 IV.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 IV.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 (IPs) $(\mathbf{q}_1, \cdots, \mathbf{q}_n)$ 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 longest 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 IV.1 ensures the output is a path along which the NR iteration function is continuous.

The maximum number of IPs 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 the path that has been successfully projected, as stated above.

1) Progressive projection: is presented in Alg. 1 and depicted in Figure 4.

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 $\mathbf{q}_0$, it builds a configuration satisfying the constraint, within the continuity interval of $\mathbf{q}_0$. The configuration is chosen towards $\mathbf{q}_e$ (Line 1.7). When $\mathbf{q}_e$ is within the continuity interval of $\mathbf{q}_0$ (Line 1.2), the algorithm succeeds.

From Figure 4a to 4b, 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 1.4-1.6). When $\lambda < \lambda_m$, the projection locally increases the distances more than $\sigma_r(\mathbf{q}_k)/(K\lambda_m)$. The algorithm considers the criteria cannot be satisfied and returns failure. If q is found in $\mathcal{B}(\mathbf{q}_0, \sigma(\mathbf{q}_0)/K)$, the left part satisfies the condition of Lemma IV.1. The right part is then projected using the same procedure.

2) Global projection: method is presented in Alg 2 and depicted in Figure 5.

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: **for all** $\mathbf{q}_k \in \mathbf{Q}$ **do**
4: **if** $||f(\mathbf{q}_k)||_2 > \epsilon$ **then**
5: $\mathbf{q}_k \leftarrow P_\alpha(\mathbf{q}_k)$
6: **for all** Consecutive $\mathbf{q}_k, \mathbf{q}_{k+1} \in \mathbf{Q}$ **do**
7: **if** $\sigma_r(\mathbf{q}_k) < K\lambda_m$ **then**
8: $\mathbf{Q} \leftarrow (\mathbf{q}_0, \cdots, \mathbf{q}_k)$ **and break**
9: $d \leftarrow \sigma_r(\mathbf{q}_k) + \sigma_r(\mathbf{q}_{k+1})$
10: **if** $d < K \times ||\mathbf{q}_k - \mathbf{q}_{k+1}||_2$ **then**
11: $\mathbf{q} \leftarrow$ INTERPOLATE($\mathbf{q}_k, \mathbf{q}_{k+1}, \frac{\sigma_r(\mathbf{q}_k)}{K}$)
12: $\mathbf{Q} \leftarrow (\mathbf{q}_0, \cdots, \mathbf{q}_k, \mathbf{q}, \mathbf{q}_{k+1}, \cdots)$
13: **if** LENGTH($\mathbf{Q}$) $> N_{max} \times ||\mathbf{q}_0 - \mathbf{q}_e||_2$ **then**
14: $\mathbf{Q}$.REMOVELASTELEMENT
15: **go to** Line 3
16: **return** $\mathbf{Q}$

The algorithm starts by computing IPs along the straight path. Then, it works in two steps. First, the IPs are improved in order to decrease the constraint violation, by applying the NR iteration function (Line 2.5). Second, it checks whether the distance between each pair of consecutive IPs $(\mathbf{q}_k, \mathbf{q}_{k+1})$ is within the union of the two continuity balls (Line 2.10). If this check fails, a new IP $\mathbf{q}$ is added at the border of the continuity ball of $\mathbf{q}_k$. Next iteration will consider the two consecutive points $(\mathbf{q}, \mathbf{q}_{k+1})$.

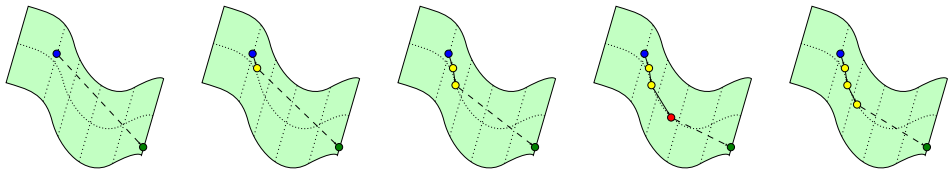
For clarity of the pseudo-code, we omitted to include a limit on the number of iterations of constraint violation reduction loops (Line 2.3). Such a limit must be integrated to avoid infinite loops due to local minima. We set this limit to 40 in our implementation and the counter is reset whenever an IP is added.

Figure 5 shows the path after some iterations. From 5b and 5c, the projection loop (Line 2.3) reduces the constraint violation point-wise. Between 5c and 5d, an IP is added (Line 2.12).

C. Discussion

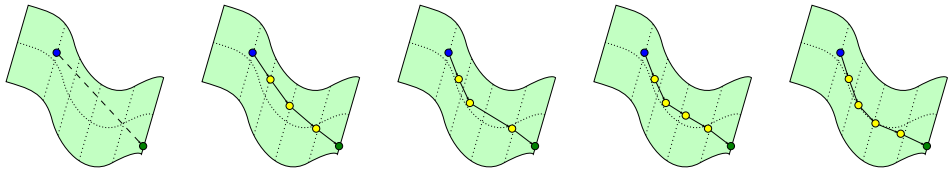
The two algorithms have the following guarantees. They provide a path with IPs satisfying the constraints. Moreover, they ensure that the NR iteration function is continuous along the lines connecting consecutive IPs. The piecewise straight interpolation is closer to constraint satisfaction than the input path and one iteration of NR is continuous. This leads to good chances to have the resulting path continuous. This is empirically verified, as shown in the next section.

Compared to our method, the RHP can prove the continuity of a motion, at the cost of being first, computationally less efficient and second, unable to return the continuously



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

Fig. 4. Progressive projection method. The green surface is $\mathbf{f}(\mathbf{q}) = 0$. 4a shows the input path. Between 5b and 5c, the interpolation point is added because it is close enough from the last point. On 5d, the point is rejected because it is too far from the last point and λ is divided by two. It results in 5d and the interpolation point is finally added.



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

Fig. 5. Global projection method. The green surface is $\mathbf{f}(\mathbf{q}) = 0$. 5a shows the input path. Between 5b and 5c, each interpolation points has been updated to decrease the constraint violation. Between 5c and 5d, an interpolation point is added because the distance between two adjacent points is bigger than the threshold.

TABLE I

BENCHMARKING SCENARIOS OF TABLE II.

	$\mathcal{C}$	Constraint	M	K
Circle	$(x, y) \in [-1, 1]^2$	$x^2 + y^2 - 1 = 0$	$2\sqrt{2}$	$2\sqrt{2}$
Parabola	$(x, y) \in [-1, 1]^2$	$y^2 - 1 = 0$	2	2
UR5	$[-2\pi, 2\pi]^6$	EE on line	6	7.14

projected part of the path. The two approaches are thus complementary. Our approach can be used at a high rate, for instance each time a path is created in a motion planner, while the RHP can be used to assert that the solution path is continuous.

The efficiency of our method, compared to RHP, comes from the expected distances between IPs. The distance between IPs generated by the RHP 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. With the proposed methods, 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 areas 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 this result.

V. SIMULATIONS

In this section, both algorithms and the RHP are compared to each other. Then, we empirically show that paths satisfying the certificate are in practice continuous. Finally, we show how to adapt constrained motion planners to generate continuous path.

The benchmarks uses the HPP software framework [2], in which the 3 algorithms have been implemented.

TABLE II

BENCHMARKS OF THE 3 ALGORITHMS. T ARE THE AVERAGE AND MAXIMUM COMPUTATION TIME OVER 10 RUNS. DIST. ARE THE MINIMUM, AVERAGE AND MAXIMUM DISTANCE, IN mm, BETWEEN CONSECUTIVE IPS. N_{IP} IS THE NUMBER OF INTERPOLATION POINTS.

		Global	Progressive	Hermite
Circ	T (ms)	0.016/0.09	0.078/0.134	503/900
	Dist.	–	284/462/512	0.05/0.075/0.1
	N_{IP}	0	4.75	28946
Para	T (µs)	201/231	173/207	17/30
	Dist.	0.2/5.1/316	1/71/316	–
	N_{IP}	10	14	0
UR5	T (ms)	85/746	6.5/9.5	160/175
	N_{IP}	123	72	3403
	Success	46%	100%	92%

A. Comparison with RHP

We compared our two algorithms and the RHP in the following problems, summarized in Table I.

1) Circle: 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.

2) Parabola: we project line segments between $(-1, 1)$ and $(\tau, -1)$ for $\tau \in [-1, 1]$. 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.

3) 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 switch between inverse kinematic solutions. A continuous solution always exists. The success

TABLE III

RESULTS OF UR5 AND KUKA MANIPULATION CASE. S, F AND N MEAN RESPECTIVELY SUCCESS, FAILURE, OCCURENCE FREQUENCY. N^c IS THE RATIO OF PATHS COMPUTED BY THE PROGRESSIVE METHOD THAT ARE CONTINUOUS.

			Progressive		RHP
Prog.	RHP	N	Time	N^c	Time
S	S	12.8%	0.019	100%	8.30
S	F	13.5%	0.074	100%	1.73
F	S	1.3%	0.328	100%	14.4
F	F	72.5%	0.082	100%	2.18

TABLE IV

INTEGRATION IN A MOTION PLANNING ALGORITHM. IN ROW "SUCCESS", "BOTH" MEANS BOTH ALGORITHM SUCCEEDED WHILE "ALONE" MEANS ONLY THE ALGORITHM OF THE COLUMN SUCCEEDED. PLANNING FAILS WHEN THE NUMBER OF ITERATION GO BEYOND 20.

	RRT		RRT + Progressive	
Success	Both	Alone	Both	Alone
Nb occurence	20	35	20	6
Planning time	0.48	0.44	1.49	1.62
Nb nodes in roadmap	11.6	12.57	5.5	7.83
% of continuous	70%	0%	100%	100%

rate, on Table II, shows the ratio of paths validated by each method. To our best knowledge, it would not be possible to compute the same motions merely using inverse kinematics.

Results are presented in Table II. On the two simple scenarios, the global method has, on average, better results than the progressive method. However, on a more realistic scenario, the progressive method outperforms the global method. Except for the special case of the parabola, the RHP is much longer than the progressive method.

B. Continuity

We consider the setup of Figure 1. UR5 and KUKA robots must cooperate to move a beam. For this experience, We only consider the manifold $\mathcal{M}$ where the two robots grasp the beam. For random placements of the beam, we generate two sets of configuration S_i and S_f in $\mathcal{M}$. Then, for each pair $(\mathbf{q}_i, \mathbf{q}_f) \in S_i \times S_f$, we apply the progressive projection method, resp. the RHP, to the straight line $(\mathbf{q}_i, \mathbf{q}_f)$ onto $\mathcal{M}$ to get, in case of success, p_{prog}, resp. p_{RHP}. When the progressive projection method succeeds, we check that the path is continuous by applying the RHP to p_{prog}.

Table III summarizes the results. Three important facts must be noted. First, the RHP failed on about 85% of the paths, which means it is rather unlikely that a randomly generated path is continuous. Second, the Progressive algorithm almost never fails where the RHP succeeds (column N). Third, in our experiments, paths returned by the progressive method were all continuous (column N^c).

C. Motion planning

Table II and III suggest the RHP is time consuming. This makes it unsuitable for a use at a high rate in a motion planner. To assert the continuity of the solution returned by a randomized motion planning algorithm, such as Randomly exploring Random Trees (RRT), we proceed in two steps. First, we adapt RRT so that all the generated paths are validated using the progressive algorithm. Then, we check the continuity of the solution using the RHP. In the scenario of Figure 1, we compute a path using, first, a standard RRT algorithm and, second, a RRT where each path is validated using the progressive algorithm, before being checked for collision. Table IV summarizes the results. RRT with the progressive algorithm always computes continuous path while being three to four times slower.

Using the manipulation framework of HPP [1], the algorithms has also been tried on real, more challenging, scenarios on Figure 1 and 2. The results are shown in the accompanying video.

Figure 2 shows a case where the planner must infer that the robot has to manipulate the door to achieve its goal. 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 can be post-processed to obtain a dynamically-feasible walking trajectory.

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.

REFERENCES

[1] Mirabel, J., Lamiraux, F.: Manipulation planning: addressing the crossed foliation issue. In: IEEE International Conference on Robotics and Automation 2017, Singapour, Singapore (May 2017)

[2] Mirabel, J., Tonneau, S., Fernbach, P., Seppälä, A.K., Campana, M., Mansard, N., Lamiraux, F.: HPP: a new software for constrained motion planning. In: International Conference on Intelligent Robots and Systems (IROS 2016), Daejeon, South Korea (October 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