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]

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

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 is often used in this context. This work tackles the problem of continuity that arises when projecting paths point wise with such method.
A theoretical proof of an interval of continuity for the Newton-Raphson iteration function is given. This interval requires to bound from above the norm of the Hessian of the constraints. A method to compute this bound for constraints involving joint positions and orientations is proposed.
Then, this theoretical result is used in two path projection algorithm to give a certificate of continuity of the continuously projected path.
Finally, simulations are run on several problems.

1 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 2, we provide a short state of the art in manipulation planning, showing how original our approach is. In Section 3, we give some useful definitions. Section 4 describes the main theoretical result and our two path projection algorithms. These algorithms make use of an upper bound on the Hessian operator of the constraint that we compute in Appendix A. Finally, the two algorithms are validated in simulations.

2 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 lever 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. Our contribution is very close to this latter work. The main differences are the following.

1. 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.
2. We only assume that the constraint is C^2 with bounded Hessian, instead of satisfying a Lipschitz condition. We also compute a bound of the Hessian for common constraints.
3. We also perform Newton-Raphson projection at curve evaluation. We thus need to project less intermediate interpolation points to make sure that the constraint is satisfied all along the path up to the precision threshold.
4. From an implementation point of view, we represent the robot configuration space as a manifold (SO(3) rotations, denoted SO_3, are represented by unit quaternions). Defining polynomials on such manifolds is not as straightforward as in a vector space.

Section 4.3 gives a more precise comparison between our method and the RHP.

3 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
$$\forall \mathbf{q} \in D_{proj}, \quad \mathbf{f}(proj(\mathbf{q})) = 0.$$

3.1 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.

$$\begin{aligned} \mathcal{SM} : \mathcal{C} \times \mathcal{C} \times C^1(\mathcal{C}, \mathbb{R}^m) &\rightarrow C^1([0,1], \mathcal{C}) \\ (\mathbf{q}_0, \mathbf{q}_e, \mathbf{f}) &\rightarrow p \end{aligned}$$

$$\forall t \in [0,1], \qquad \mathbf{f}(p(t)) = 0$$

We denote by **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 $\mathbf{f}$.

4 Path projection algorithm

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

The Newton-Raphson algorithm iteratively updates the robot configuration so as to decrease the norm of an error function $\mathbf{f}$. Let $\alpha > 0$ and $P_\alpha \in \mathcal{F}(\mathcal{C}, \mathcal{C})$ be the Newton-Raphson 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 Newton-Raphson algorithm, starting at $\mathbf{q}$.

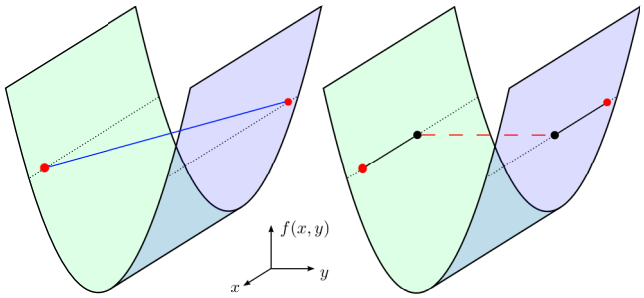

Fig. 1. This 2D example, where (x, y) are the configuration parameters, shows the graph of $\mathbf{f}((x, y)) = y^2 - 1$. The 2 dotted horizontal line are the solutions of $\mathbf{f}((x, y)) = 0$. The 2 red circles are two configurations satisfying $\mathbf{f}(\mathbf{q}) = 0$. On the left, the blue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

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:

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

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

The continuity of P_α is expressed as follows.

Lemma 1 (Continuity of the Newton-Raphson 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,

$$\forall \mathbf{q} \in \mathcal{C}, rank(\mathbf{J}(\mathbf{q})) = r \Rightarrow P_\alpha \text{ is continuous on } Ball\left(\mathbf{q}, \frac{\sigma(\mathbf{q})}{K}\right)$$

4.1 Proof of continuity of the Newton-Raphson iteration function

This section provides a proof of Lemma 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 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 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 1 proves that $\mathbf{J}^\dagger$ is a continuous function of $\mathbf{q}$ on $\mathcal{U}$. In the following section, we prove that $\mathcal{U} = \text{Ball}\left(\mathbf{q}, \frac{\sigma}{K}\right)$ is a suitable neighborhood.

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 $|||.|||_F$.

Applying Theorem 6 of [16] to the Frobenius norm, we have:

Theorem 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 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 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,

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

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 Ball $\left(\mathbf{q}, \frac{\sigma}{K}\right)$. By Theorem 1, $\mathbf{J}(\mathbf{q})^\dagger$ is continuous. P_α is the composition of continuous functions so it is continuous on Ball $\left(\mathbf{q}, \frac{\sigma}{K}\right)$. This proves Lemma 1.

4.2 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, a Lipschitz constant must be computed for the Jacobian of the constraint. Appendix A proposes a method to bound from above the norm of the Hessian for constraints involving joint placements. This upper bound is a Lipschitz constant of the Jacobian. This method also extends to constraints involving the center of mass (COM) of the robot as the COM is a weighed sum of joint positions.

A Lipschitz constant might be hard to compute in some cases. Moreover, the larger the upper bound, the smaller the interval of continuity and the greater the number of interpolation points. Because of these two limitations, two versions of each algorithms exist. The strong version makes use of the continuity interval of Lemma 1. A Lipschitz constant of the Jacobian of the constraint must be known. The output is a path along which the Newton-Raphson iteration function is continuous. The weak version, with a weaker guaranty, ensures continuity up to a threshold. It introduces a parameter δ_M, set to 0.02, which limits the maximum distance between interpolation points.

For both algorithms, the limit the number of interpolation points on unit length paths N_{max} is set to 20 and the minimal interpolation distance λ_m is set to 0.001. These parameters ensure our algorithms terminate.

Progressive projection method is presented in Alg 1 in its weak version.

Algorithm 1 Progressive continuous projection

function PROJECT($\mathbf{q}_0, \mathbf{q}_e, \mathbf{f}$)
  ▷ Continuously project the direct path $(\mathbf{q}_0, \mathbf{q}_e)$ onto the submanifold $\mathbf{f}(\mathbf{q}) = 0$
  for $k = 0 \cdots N_{max} \times ||\mathbf{q}_0 - \mathbf{q}_e||_2$ **do**
    if $||\mathbf{q}_k - \mathbf{q}_e||_2 < \delta_M$ **then return** $(\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_k, \mathbf{q}_e)$
    $\lambda \leftarrow \delta_M$
    repeat
      if $\lambda < \lambda_m$ **then return** $(\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_k)$
      $\mathbf{q} \leftarrow$ INTERPOLATE($\mathbf{q}_k, \mathbf{q}_e, \lambda$)
      $\mathbf{q} \leftarrow$ PROJECT($\mathbf{q}, \mathbf{f}$)
      $\lambda \leftarrow \frac{\lambda}{2}$
    until $||\mathbf{q}_k, \mathbf{q}||_2 < \delta_M$ **and** Projection succeeded
    $\mathbf{q}_{k+1} \leftarrow \mathbf{q}$
  return $(\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_k)$

From $\mathbf{q}_0$, it builds a list of configurations satisfying the constraints, within the continuity interval of the previous configuration. The list is iteratively grown towards $\mathbf{q}_e$. When $\mathbf{q}_e$ is within the continuity interval of the last configuration, the algorithm succeeded.

Figure 2 shows the path at some iterations. From 2a to 2b, the path is cut in two at distance λ from the start configuration. λ is decided at lines 7 to 11. λ is chosen so that the projected configuration is at a distance less than δ_M, the discontinuity tolerance, from the previous interpolation point. When $\lambda < \lambda_m$, the projection locally increases the distances more than δ_M/λ_m. The path is considered discontinuous and the algorithm fails. In case of success, the left part, of length λ, is assumed to be continuous. The right part will be projected at next iterations.

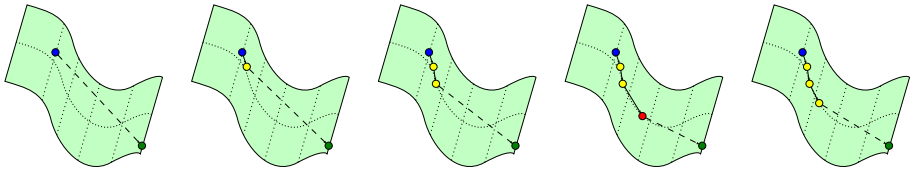

Fig. 2. Progressive projection method. The green curve is $\mathbf{f}(\mathbf{q}) = 0$. 2a shows the input path. Between 3b and 3c, the interpolation point is added because it is close enough from the last point. On 3d, the point is rejected because it is too far from the last point and λ is divided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

Global projection method is presented in Alg 2 in its strong version and depicted in Figure 3.

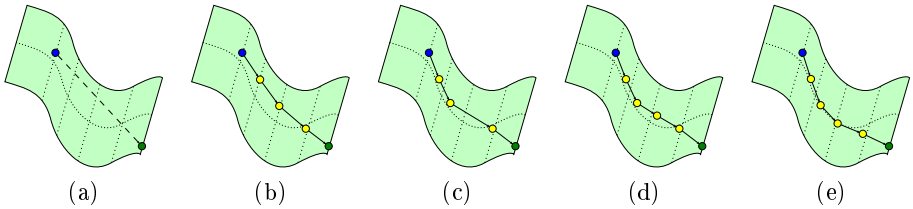

Fig. 3. Global projection method. The green curve is $\mathbf{f}(\mathbf{q}) = 0$. 3a shows the input path. Between 3b and 3c, each interpolation points has been updated to decrease the constraint violation. Between 3c and 3d, an interpolation point is added because the distance between two adjacent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

The algorithm starts by computing interpolation points along the straight path. Then, it works in two steps. First, the interpolation points are improved

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** $\mathbf{Q} \leftarrow (\mathbf{q}_0, \cdots, \mathbf{q}_k)$ **and break**
12: $d \leftarrow \sigma_r(\mathbf{q}_k) + \sigma_r(\mathbf{q}_{k+1})$
13: **if** $d < K \times ||\mathbf{q}_k - \mathbf{q}_{k+1}||_2$ **then**
14: $\mathbf{q} \leftarrow$ INTERPOLATE$(\mathbf{q}_k, \mathbf{q}_{k+1}, \frac{\sigma_r(\mathbf{q}_k)}{K})$
15: $\mathbf{Q} \leftarrow (\mathbf{q}_0, \cdots, \mathbf{q}_k, \mathbf{q}, \mathbf{q}_{k+1}, \cdots)$
16: $repeat \leftarrow True$
17: **if** LENGTH$(\mathbf{Q}) > N_{max} \times ||\mathbf{q}_0 - \mathbf{q}_e||_2$ **then** $\mathbf{Q}$.REMOVELASTELEMENT
18: **return** $\mathbf{Q}$

in order to decrease the constraint violation, by applying the Newton-Raphson 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 13). 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).

4.3 Discussion

The strong version of the two algorithms presented here have the following guaranties. They provide a path with interpolation points satisfying the constraints. Moreover, they ensure that the Newton-Raphson 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 Newton-Raphson 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 strict continuity, at the cost of being computationally less efficient. Indeed, RHP generates a lot more interpolation points than us. 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.

5 Simulations

In this section, each version of both algorithms 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 4 algorithms have been implemented.

Quadratic problems We first compare the two versions of each algorithm for various parameter in the following problems.

- Circle: the configuration space is $\mathbb{R}^2$, subject to constraint $f_1(x, y) = x^2 + y^2 - 1 = 0$. A Lipschitz constant of the 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 $\mathbb{R}^2$, subject to constraint $f_1(x, y) = x^2 - 1 = 0$. A Lipschitz constant of the Jacobian is $K = 2$. We project line segments between $(1, 0)$ and $(-1, \tau)$ for $\tau \in [0, 2]$. All the algorithms were able to detect the discontinuity.

Results are presented in Table 1. The global projection method outperforms the progressive method on these quadratic problems.

Manipulation planning

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. A Lipschitz constant of the Jacobian, using Eq. (7) with $L = 1m$, is $K = \sqrt{36L^2 + 15} < 8$. Table 2 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.

Global proj.	Circle		Parabola	
K	-	2	-	$2\sqrt{2}$
t_{avg}/t_{max} (µs)	877/1617	43/289	420/478	569/610
$d_{min}/d_{avg}/d_{max}$(mm)	8/44/99	-	11/27/45	0/25/316
N_{ip}	38	0	6.5	9.5

Progressive proj.	Circle		Parabola	
K	-	2	-	$2\sqrt{2}$
t_{avg}/t_{max} (µs)	739/1399	1541/2189	995/1838	594/1003
$d_{min}/d_{avg}/d_{max}$ (mm)	7/67/100	35/90/100	1/11/45	1/45/94
N_{ip}	33	22	39	11

Table 1. Quadratic problems benchmarks. K is the Lipschitz constant of the Jacobian, where "-" for the weak version is used. 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}.

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.

Projection method	Global		Progressive	
	strong	weak	strong	weak
t_{avg}/t_{max} (ms)	1313/2475	243/613	426/449	146/169
N_{ip}	149	216	267	261
False negative	71%	71%	0%	0%

Table 2. Results of UR5 case. The rows have the same meaning as in Table 1. The number of false negative corresponds to the ratio of rejected path over all tests, while a continuous path exists.

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 valid end of the path. And second, check the projected path for collision.

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 [17]. 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.

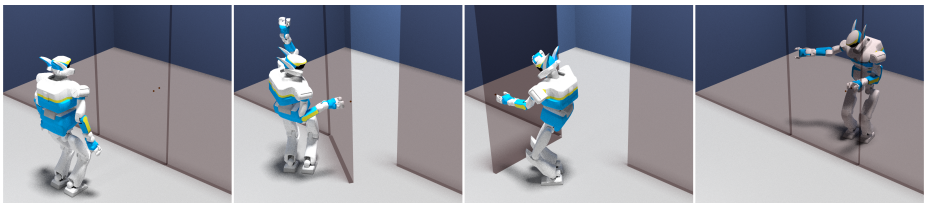

Fig. 4. HRP2 opening a door.

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.

6 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. 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

A Lipschitz constant of the Jacobian matrix

A.1 Notation

We consider a tree of joints with N joints and d DOF. Each joint can have one or several DOF. $p(n)$ denotes the parent joint of joint n and $p^k(n) = p(p^{k-1}(n))$ is its k-th ancestor. O_n, resp. $\mathbf{n_n}$, is the center, resp. normal, of joint n. Let L be the longest possible distance between two centers of joint.

Let $I(n)$ be the set of joint indexes of the chain between the root joint and joint n. Let $I_R(n)$ (resp. $I_T(n)$, $I_{SO_3}(n)$) be the subset of indexes of rotation (resp. translation, SO_3) joint in $I(n)$. They are such that $I(n) = I_R(n) \cup I_T(n) \cup I_{SO_3}(n)$ and $I_R(n) \cap I_T(n) = I_T(n) \cap I_{SO_3}(n) = I_{SO_3}(n) \cap I_R(n) = \emptyset$.

$[\mathbf{u}]_X$ is the cross matrix, *i.e.* $[\mathbf{u}]_X \mathbf{v} = \mathbf{u} \wedge \mathbf{v}$. $[[\mathbf{M}_{3,n}]]_X$ is the cross tensor, *i.e.* $[[\mathbf{M}_{3,n}]]_X \mathbf{x}_n = [\mathbf{M}_{3,n}\mathbf{x}_n]_X$.

A.2 Jacobian

The Jacobian of the placement of joint n is ${}^n\mathbf{J} \in \mathcal{M}_{6\times d}(\mathbb{R})$. ${}^n\mathbf{J}^C_j$ is the block corresponding to joint j in ${}^n\mathbf{J}$. ${}^n\mathbf{J}^v_j$ denotes the velocity part of ${}^n\mathbf{J}^C_j$. ${}^n\mathbf{J}^\omega_j$ denotes the angular velocity part of ${}^n\mathbf{J}^C_j$. The following list explicity the different block of ${}^n\mathbf{J}$. The unlisted block are matrices of zeros.

- Joint $p^k(n)$ is a rotation:

$$ {}^n\mathbf{J}^v_{p^k(n)} = \mathbf{O_n O_{p^k(n)}} \wedge \mathbf{n_{p^k(n)}}, {}^n\mathbf{J}^\omega_{p^k(n)} = \mathbf{n_{p^k(n)}} $$

- Joint $p^k(n)$ is a translation:

$$ {}^n\mathbf{J}^v_{p^k(n)} = \mathbf{n_{p^k(n)}}, {}^n\mathbf{J}^\omega_{p^k(n)} = 0 $$

- Joint $p^k(n)$ is a SO_3:

$$ {}^n\mathbf{J}^v_{p^k(n)} = \left[\mathbf{O_n O_{p^k(n)}}\right]_X, {}^n\mathbf{J}^\omega_{p^k(n)} = I_3 $$

Thus, when $p^k(n)$ is a rotation, we have $||{}^n\mathbf{J}^v_{p^k(n)}||_2 \leq L$ and $||{}^n\mathbf{J}^v_{p^k(n)}||_2 \leq 1$. Otherwise, we have $||{}^n\mathbf{J}^{v,w}_{p^k(n)}||_2 \leq 1$.

A.3 Hessian

The Hessian matrix is defined by ${}^n\mathbf{H}_{i,j,k} = \frac{\partial {}^n\mathbf{J}_{i,j}}{\partial \dot{\mathbf{q}}_k}$. Similarly to ${}^n\mathbf{J}^{v,\omega}_j$, we denote ${}^n\mathbf{H}^{v,\omega}_{j,k} = \frac{\partial {}^n\mathbf{J}^{v,\omega}_j}{\partial \dot{\mathbf{q}}_k}$.

Element of the Hessian matrix

- Joint $p^j(n)$ is a rotation:

$$ \begin{aligned}
j > k, {}^n\mathbf{H}^v_{p^j(n),p^k(n)} &= \left[\mathbf{n_{p^j(n)}}\right]_X {}^n\mathbf{J}^v_{p^k(n)} \\
{}^n\mathbf{H}^\omega_{p^j(n),p^k(n)} &= 0 \\
j \leq k, {}^n\mathbf{H}^v_{p^j(n),p^k(n)} &= \left[\mathbf{n_{p^j(n)}}\right]_X \left({}^n\mathbf{J}^v_{p^k(n)} - {}^{p^j(n)}\mathbf{J}^v_{p^k(n)}\right) \\
&\quad - \left[\mathbf{O_n O_{p^j(n)}}\right]_X \left(\left[\mathbf{n_{p^j(n)}}\right]_X {}^{p^j(n)}\mathbf{J}^\omega_{p^k(n)}\right) \\
{}^n\mathbf{H}^\omega_{p^j(n),p^k(n)} &= -\left[\mathbf{n_{p^j(n)}}\right]_X {}^{p^j(n)}\mathbf{J}^\omega_{p^k(n)}
\end{aligned} $$

- Joint $p^j(n)$ is a translation:

$$ {}^n\mathbf{H}^v_{p^j(n),p^k(n)} = -\left[\mathbf{n_{p^j(n)}}\right]_X {}^{p^j(n)}\mathbf{J}^\omega_{p^k(n)} $$
$$ {}^n\mathbf{H}^\omega_{p^j(n),p^k(n)} = 0 $$

- Joint $p^j(n)$ is a SO_3:

$$ {}^n\mathbf{H}^v_{p^j(n),p^k(n)} = \left[\left[{}^n\mathbf{J}^v_{p^k(n)} - {}^{p^j(n)}\mathbf{J}^v_{p^k(n)}\right]\right]_X $$
$$ {}^n\mathbf{H}^\omega_{p^j(n),p^k(n)} = 0 $$

Bounds By the mean value theorem, an upper bound of $||||{}^n\mathbf{H}(\mathbf{q})||||_F$ on $\mathcal{C}$ is a suitable Lipschitz constant for ${}^n\mathbf{J}$. An explicit upper bound is computed in this section.

We are interested in joint trees, which makes the Hessian matrix sparse. Joints not in $I(n)$ are not influencing the placement of joint n so:

$$||||{}^n\mathbf{H}||||_F^2 = \sum_{j\in I(n),k\in I(n)} ||{}^n\mathbf{H}^v_{j,k}||_2^2 + ||{}^n\mathbf{H}^\omega_{j,k}||_2^2 \tag{2}$$

It gives the following bound:

$$||||{}^n\mathbf{H}||||_F^2 \leq |I(n)|^2(\max(9L^2,(L+2)^2)+1) \tag{3}$$

We denote $\sigma(m,\chi,\kappa) = \sum_{j\in I_\chi(n),k\in I_\kappa(n)} ||{}^n\mathbf{H}^m_{j,k}||_2^2$. Bounds for σ are summarized in Table 3.

Eq. 2 becomes

$$||||{}^n\mathbf{H}||||_F^2 = \sum_{m\in\{v,\omega\},(\chi,\kappa)\in\{R,T,SO_3\}^2} \sigma(m,\chi,\kappa)$$

The element of the Hessian matrix given above gives, we have:

$$\begin{aligned}
\forall\kappa\in\{R,T,SO_3\}, \quad & \sigma(\omega,T,\kappa)=0\\
\forall\kappa\in\{R,T,SO_3\}, \quad & \sigma(\omega,SO_3,\kappa)=0\\
& \sigma(\omega,R,T)=0\\
& \sigma(\omega,R,SO_3)\leq |I_R(n)||I_{SO_3}(n)|\\
& \sigma(v,T,T)=0\\
& \sigma(v,R,T)\leq 4|I_R(n)||I_T(n)|\\
& \sigma(v,T,R)\leq 2|I_T(n)||I_R(n)|\\
& \sigma(v,T,SO_3)\leq 2|I_T(n)||I_{SO_3}(n)|
\end{aligned}$$

Moreover, as $\forall(j,k)\in I_R(n)\times I(n)|j\geq k$, ${}^n\mathbf{H}^\omega_{j,k}=0$ and $||{}^n\mathbf{H}^\omega_{j,k}||_2\leq 1$,

$$\sigma(\omega,R,R)\leq \sum_{j,k\in I_R(n)^2,j<k} 1 = \frac{|I_R(n)|(|I_R(n)|-1)}{2} \tag{4}$$

$\forall j\in I_{SO_3}$, we have:

$$||{}^n\mathbf{H}^v_{j,k}||_2^2 = 2(||{}^n\mathbf{J}^v_k - {}^j\mathbf{J}^v_k||_2^2) \leq \begin{cases} 2L^2 \text{ if } k\in I_R(n)\\ 2 \text{ if } k\in I_T(n)\\ 2L^2 \text{ if } k\in I_{SO_3}(n)\end{cases}$$

So we have:

$$\sigma(v,SO_3,\kappa)\leq 2|I_{SO_3}(n)|\times\begin{cases} 2L^2|I_R(n)| \text{ if } \kappa=R\\ 2|I_T(n)| \text{ if } \kappa=T\\ 2L^2(|I_{SO_3}(n)|-1) \text{ if } \kappa=SO_3\end{cases}$$

The Jacobi identity of cross product on ${}^{n}\mathbf{H}^{v}_{j,k}$ where $j \in I_R(n)$ gives:

$$\sigma(v, R, R) \leq |I_R(n)|^2 L^2 \tag{5}$$
$$\sigma(v, R, SO_3) \leq 2|I_R(n)||I_{SO_3}(n)|L^2 \tag{6}$$

$\chi \backslash \kappa$	R	T	SO_3
R	$\|I_R\|^2 L^2$	$4\|I_R\|\|I_T\|$	$2\|I_R\|\|I_{SO_3}\|L^2$
T	$2\|I_R\|\|I_T\|$	0	$2\|I_T\|\|I_{SO_3}\|$
SO_3	$4\|I_R\|\|I_{SO_3}\|L^2$	$4\|I_T\|\|I_{SO_3}\|$	$4\|I_{SO_3}\|(\|I_{SO_3}\|-1)L^2$

$\chi \backslash \kappa$	R	SO_3
R	$\|I_R\|\left(\|I_R\|-1\right)/2$	$\|I_R\|\|I_{SO_3}\|$

Table 3. Upper bound for $\sigma(v, \chi, \kappa)$ (left) and $\sigma(\omega, \chi, \kappa)$ (right). Omitted combination are null.

The above inequalities put together give the following bound.

$$\begin{aligned} ||||{}^{n}\mathbf{H}||||_F^2 \leq & \left(|I_R|^2 + 6|I_R||I_{SO_3}| + 4|I_{SO_3}|(|I_{SO_3}|-1)\right) \times L^2 \\ & + 6|I_T||I_R| + 6|I_T||I_{SO_3}| + \frac{|I_R|(|I_R|-1)}{2} + |I_R||I_{SO_3}| \end{aligned} \tag{7}$$