Randomized Planning of Dynamic Motions Avoiding Forward Singularities

Ricard Bordalba, Lluís Ros, and Josep M. Porta

Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
08028 Barcelona, Catalonia
{rbordalba,ros,porta}@iri.upc.edu

Abstract. Forward singularities, also known as direct, or actuator singularities, cause many problems to the planning and control of robot motions. They yield position errors and rigidity losses of the robot, and generate unbounded actions in typical control laws. To circumvent these issues, this paper proposes a randomized kinodynamic planner for computing trajectories avoiding such singularities. Given initial and final states for the robot, the planner attempts to connect them by means of a dynamically-feasible, singularity-free trajectory that also respects the force limits of the actuators. The performance of the strategy is illustrated in simulation by means of a parallel robot performing a highly-dynamic task.

Keywords: Kinodynamic planning, singularity, constrained system, RRT.

1 Introduction

The kinodynamic motion planning problem constitutes an important research theme in robotics. The problem entails computing trajectories connecting two given states, both with a prescribed position and velocity of a robot. The trajectories must be free of collisions and have to be consistent with the system dynamics, i.e., they can only involve acceleration profiles compatible with the limited forces deliverable by the actuators. The problem has been extensively studied for open-chain robots, and general practical solutions have been given [14], prominent ones being those based on randomized methods [15]. In comparison, the topic has received much less attention in the case of robots with closed kinematic chains (see [8] and references therein). These robots complicate the problem substantially because their state space is, in general, an implicitly-defined manifold not admitting a global parameterization. As a result, the extension of randomized methods like [15] to the closed-chain case is difficult since (1) it is not easy to come up with a mechanism to uniformly sample an implicitly-defined state space, (2) the numerical integration of the motion equations alone (disregarding the loop closure constraints) generates trajectories that drift away from such space, and (3) the mechanism exhibits forward singularities, whose traversal may compromise the control of the robot at execution time [2,10,5,16]. The first two difficulties were recently addressed by Bordalba et al. [8], who showed

that the iterative construction of manifold atlases provides effective sampling and numerical integration schemes over the state space. The purpose of this paper is to address the third difficulty: to make the planner in [8] robust to the presence of forward singularities.

It is well known that in a forward singularity the velocities of the actuators do not determine the full configuration velocity of the robot [4,6,5]. Such singularities, however, also have an impact on the system dynamics. The robot becomes underactuated, being unable to traverse these configurations under arbitrary accelerations. In their proximity, in fact, the inverse dynamic problem yields very large or unbounded motor actions, which can cause controllability issues or even a breakage of the mechanism. To circumvent these issues, several strategies have been proposed. For example, one can use actuation redundancy to avoid all, or almost all forward singularities [17], or design specific control laws able to traverse the singularities with bounded actions and consistent accelerations [10,12]. Although both approaches are interesting, they also come with a few caveats. The use of additional actuators yields more complex robot designs, and current motion controllers used to traverse singularities tend to be intricate and may not cope with all possible degeneracies of the forward kinematic map [16]. An alternative is to avoid the singularities at the planning stage, but current planners for this purpose cannot cope with dynamic constraints. As described in [3], such planners avoid the singularities by treating them as obstacles, or by defining a singularity-free configuration space. While the former approach requires the configuration space to be globally parametrizable, the latter remains applicable to general robot designs. This justifies the strategy we present in this paper, which extends the work in [3] to also satisfy dynamic constraints.

Our approach relies on a system of equations characterizing the singularity-free state space of a robot (Sec. 2). The solution set of this system is a smooth manifold diffeomorphic to the classic state space, but with all forward singularities removed. We also adapt the dynamic equations of the robot to define an action-varying vector field on this manifold, which we use to steer the robot between two states, while avoiding forward singularities (Sec. 3). We illustrate the planning method on a parallel robot that has to perform a highly dynamic task (Sec. 4) and draw a few conclusions from the study (Sec. 5).

2 Dynamics on the singularity-free state space

Let us assume that the configuration of our robot is described by means of a tuple $\boldsymbol{q} = (\boldsymbol{a}, \boldsymbol{r})$ of n_q generalised coordinates, where $\boldsymbol{a}$ contains the n_a actuated degrees of freedom of the robot, and $\boldsymbol{r}$ encompasses the remaining coordinates in $\boldsymbol{q}$. Since the robot contains closed kinematic chains, the components of $\boldsymbol{q}$ will not be independent; they will be subject to a system of n_e equations

$$\boldsymbol{\Phi}(\boldsymbol{q}) = \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{\Phi}(\boldsymbol{q})$ is a differentiable vector function defined by the loop closure constraints. Generically, then, the configuration space $\mathcal{C}$ of the robot will be a manifold of dimension $d_{\mathcal{C}} = n_q - n_e$ formed by all points $\boldsymbol{q}$ satisfying Eq. (1).

Although we have defined $\mathcal{C}$, the planning of dynamic motions is better solved in the state space $\mathcal{X}$ of the robot. In closed-chain systems, this space is traditionally defined as the set of pairs $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ that satisfy Eq. (1) and its time derivative

$$\boldsymbol{\Phi_q} \cdot \dot{\boldsymbol{q}} = \boldsymbol{0}, \tag{2}$$

where $\boldsymbol{\Phi_q} = \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{q}}$. However, since we wish to generate motions avoiding forward singularities, we shall define $\mathcal{X}$ using an additional constraint. Recall here that a forward singularity is a point $\boldsymbol{q}$ for which the $n_e \times n_e$ Jacobian $\boldsymbol{\Phi_r} = \partial \boldsymbol{\Phi} / \partial \boldsymbol{r}$ is rank deficient [4]. To exclude these singularities from $\mathcal{X}$, thus, we need to enforce $\det(\boldsymbol{\Phi_r}) \neq 0$. This can be satisfied by introducing the constraint

$$b \cdot \det(\boldsymbol{\Phi_r}) = 1, \tag{3}$$

where b is a newly-defined auxiliary variable. Note that if we let $d = \det(\boldsymbol{\Phi_r})$, then Eq. (3) can be written as $b \cdot d = 1$, which defines a hyperbola that never crosses the $d = 0$ axis of the (b, d) plane. In our formulation, thus, $\mathcal{X}$ will be the set of points $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}}, b)$ satisfying the system formed by Eqs. (1)-(3). For short, this system will hereafter be written as

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}. \tag{4}$$

Now note that the time evolution of the robot is determined by its motion equations, which can be obtained using the Euler-Lagrange formalism for example. In our case, the fact that $\boldsymbol{\Phi_r}$ is full rank for all $\boldsymbol{x} \in \mathcal{X}$ guarantees that $\boldsymbol{\Phi_q}$ is also full rank on $\mathcal{X}$. This allows the motion equations to be written in the explicit form $\ddot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{u})$ for all $\boldsymbol{x} \in \mathcal{X}$, where $\boldsymbol{u} \in \mathbb{R}^{n_a}$ is the vector of actuator forces of the robot. By defining $\boldsymbol{v} = \dot{\boldsymbol{q}}$, we can then write this equation in the first-order form used in numerical integration schemes:

$$\begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{v}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{u}) \end{bmatrix}. \tag{5}$$

The evolution of b is also governed by a differential equation, which can be obtained by taking the time derivative of Eq. (3) to yield

$$\dot{b} = -\frac{b}{\det(\boldsymbol{\Phi_r})} \cdot \frac{d}{dt}(\det(\boldsymbol{\Phi_r})). \tag{6}$$

By noting that $\dot{\boldsymbol{x}} = (\dot{\boldsymbol{q}}, \dot{\boldsymbol{v}}, \dot{b})$, Eqs. (5) and (6) can be compactly written as

$$\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{q}, \boldsymbol{v}, b, \boldsymbol{u}). \tag{7}$$

Note that for each value of $\boldsymbol{u}$, Eq. (7) defines a vector field over $\mathcal{X}$, which can be used in conjunction with Eq. (4) to integrate the motion of the robot forward in time, using proper numerical methods [19]. Given two states of $\mathcal{X}$, $\boldsymbol{x}_s$ and $\boldsymbol{x}_g$, the goal of our planner is then to find an action trajectory $\boldsymbol{u}(t)$ such that the trajectory $\boldsymbol{x}(t)$ determined by Eqs. (4) and (7) for $\boldsymbol{x}(0) = \boldsymbol{x}_s$ satisfies $\boldsymbol{x}(t_f) = \boldsymbol{x}_g$ for some time $t_f > 0$. For any time $t \in [0, t_f]$, moreover, we shall require $\boldsymbol{x}(t)$ to stay inside a region $\mathcal{X}_{\text{feas}} \subset \mathcal{X}$ of non-collision states, and we shall confine $\boldsymbol{u}(t)$ to a bounded subset $\mathcal{U} \in \mathbb{R}^{n_a}$ given by the actuator force limits.

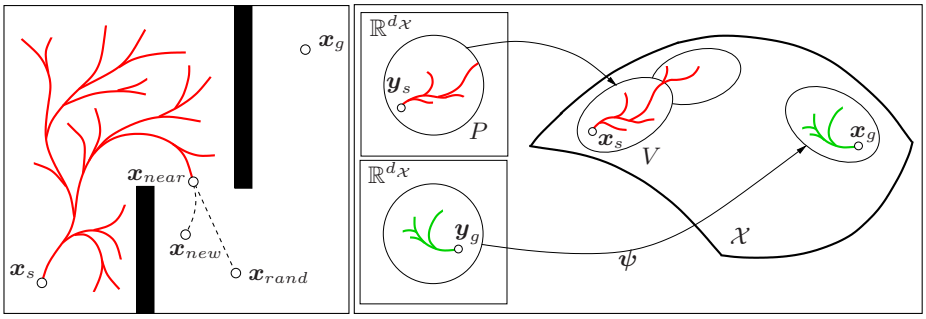

Fig. 1: Left: Extension process of an RRT. Right: Construction of an RRT on an implicitly-defined state space manifold.

3 Planning in the singularity-free state space

To explain the planner in an easy way, we follow [7]. Suppose initially that the variables in $\boldsymbol{x}$ are independent; i.e., that Eq. (4) does not apply, so that $\mathcal{X}$ can be thought of as $\mathbb{R}^{2n_q+1}$. In this situation, we can look for a trajectory connecting $\boldsymbol{x}_s$ with $\boldsymbol{x}_g$ by constructing a rapidly-exploring random tree (RRT) over $\mathcal{X}$ [15]. The RRT is rooted at $\boldsymbol{x}_s$ and it is grown incrementally towards $\boldsymbol{x}_g$ while staying inside $\mathcal{X}_{\text{feas}}$. Every tree node stores a feasible state $\boldsymbol{x} \in \mathcal{X}_{\text{feas}}$, and every edge stores the action $\boldsymbol{u} \in \mathcal{U}$ needed to move between the connected states. This action is assumed to be constant during a fixed time Δt. The expansion of the RRT proceeds by applying three steps repeatedly (Fig. 1, left). First, a state $\boldsymbol{x}_{rand} \in \mathcal{X}$ is randomly selected; then, the RRT state $\boldsymbol{x}_{near}$ that is closest to $\boldsymbol{x}_{rand}$ is computed according to some metric; finally, a movement from $\boldsymbol{x}_{near}$ towards $\boldsymbol{x}_{rand}$ is performed by applying an action $\boldsymbol{u} \in \mathcal{U}$ during Δt seconds. The movement from $\boldsymbol{x}_{near}$ towards $\boldsymbol{x}_{rand}$ is simulated by integrating Eq. (7) numerically, which yields a new state $\boldsymbol{x}_{new}$ that may or may not be in $\mathcal{X}_{\text{feas}}$. In the former case, $\boldsymbol{x}_{new}$ is added to the RRT, and in the latter it is discarded. To test whether $\boldsymbol{x}_{new} \in \mathcal{X}_{\text{feas}}$, $\boldsymbol{x}_{new}$ is checked for collisions by using standard algorithms [13], and the joint positions are computed to check whether they stay within bounds. The action $\boldsymbol{u}$ applied is typically chosen as the one from $\mathcal{U}$ that brings the robot closer to $\boldsymbol{x}_{rand}$. One can either try all possible values in $\mathcal{U}$ (if it is a discrete set) or only those of n_s random points on $\mathcal{U}$ (if it is continuous). To force the RRT to extend towards $\boldsymbol{x}_g$, $\boldsymbol{x}_{rand}$ is set to $\boldsymbol{x}_g$ once in a while, stopping the whole process when a RRT leaf is close enough to $\boldsymbol{x}_g$. Usually, however, a solution trajectory can be found more rapidly if two RRTs respectively rooted at $\boldsymbol{x}_s$ and $\boldsymbol{x}_g$ are grown simultaneously towards each other. The expansion of the tree rooted at $\boldsymbol{x}_g$ is based on the integration of Eq. (7) backwards in time.

When the coordinates in $\boldsymbol{x}$ are constrained by Eq. (4), $\mathcal{X}$ becomes an non-linear manifold of dimension $d_\mathcal{X} = 2d_\mathcal{C}$. This fact complicates the generation of RRTs over $\mathcal{X}$ because there is no straightforward way to randomly select points $\boldsymbol{x}$ satisfying Eq. (4), and the numerical integration of Eq. (7) easily drifts away

from $\mathcal{X}$ when standard methods for ordinary differential equations are used. However, these two issues can be circumvented by constructing an atlas of $\mathcal{X}$ in parallel to the RRT [8].

An atlas is a collection of charts mapping $\mathcal{X}$ entirely, where each chart is a local diffeomorphism $\boldsymbol{\psi}$ from an open set $P \subseteq \mathbb{R}^{d_{\mathcal{X}}}$ of parameters to an open set $V \subset \mathcal{X}$ (Fig. 1, right). The V sets can be thought of as partially-overlapping tiles covering $\mathcal{X}$, in a way that every $\boldsymbol{x} \in \mathcal{X}$ lies in at least one set V. Assuming that an atlas is available, the problem of sampling $\mathcal{X}$ boils down to generating random values $\boldsymbol{y}$ in the P sets, since these values can always be projected to $\mathcal{X}$ using $\boldsymbol{x} = \boldsymbol{\psi}(\boldsymbol{y})$. Also, the atlas allows the conversion of the vector field defined on $\mathcal{X}$ by Eq. (7) into one in the coordinate spaces P, which permits the integration of Eq. (7) using local coordinates [19]. As a result, the RRT motions satisfy Eq. (4) by construction, eliminating any drift from $\mathcal{X}$ to machine precision.

The construction of the atlas is incremental. The atlas is initialized with two charts covering $\boldsymbol{x}_s$ and $\boldsymbol{x}_g$, respectively (Fig. 1, right). Then, these charts are used to pull the expansion of the RRT, which in turn adds new charts to the atlas as needed, until $\boldsymbol{x}_s$ and $\boldsymbol{x}_g$ become connected. To be able to construct the charts, the method requires $\mathcal{X}$ to be smooth, i.e., with a well-defined tangent space at all of its points. However, using the formulation in Section 2, we can see that this property holds. Observe that all functions defining $\boldsymbol{F}(\boldsymbol{x})$ are differentiable, so that the Jacobian of $\boldsymbol{F}(\boldsymbol{x})$,

$$\boldsymbol{F}_{\boldsymbol{x}} = \boldsymbol{F}_{\boldsymbol{q},\dot{\boldsymbol{q}},b} = \left[\begin{array}{c|c|c} \boldsymbol{\Phi}_{\boldsymbol{q}} & \mathbf{0} & \mathbf{0} \\ \hline \frac{\partial \boldsymbol{\Phi}_{\boldsymbol{q}}}{\partial \boldsymbol{q}} \cdot \dot{\boldsymbol{q}} & \boldsymbol{\Phi}_{\boldsymbol{q}} & \mathbf{0} \\ \hline b \cdot \frac{\partial \det(\boldsymbol{\Phi}_{\boldsymbol{r}})}{\partial \boldsymbol{q}} & \mathbf{0} & \det(\boldsymbol{\Phi}_{\boldsymbol{r}}) \end{array}\right], \tag{8}$$

is well defined at every $\boldsymbol{x} \in \mathcal{X}$. This Jacobian, moreover, is full rank for all $\boldsymbol{x} \in \mathcal{X}$ because each of the diagonal blocks contains a non-vanishing minor of maximal size (since $\det(\boldsymbol{\Phi}_{\boldsymbol{r}}) \neq 0$ for all $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{\Phi}_{\boldsymbol{q}}$ is guaranteed to be full rank on $\mathcal{X}$). By the implicit function theorem, these facts imply that $\mathcal{X}$ is smooth.

4 A weight throwing task

The previous planner has been implemented in C and it has been integrated into the CUIK Suite [18]. We next illustrate its performance on planning a weight throwing task for a 5-bar robot with the geometry shown in Fig. 2. The angles q_1 and q_5 are actuated, allowing to control the (x, y) position of point Q, which is taken as the end effector. Various versions of this mechanism have been used both in research and the industry, including the DEXTAR and RAPI-MOD models from

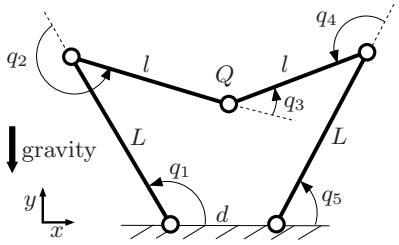

Fig. 2: A 5-bar robot.

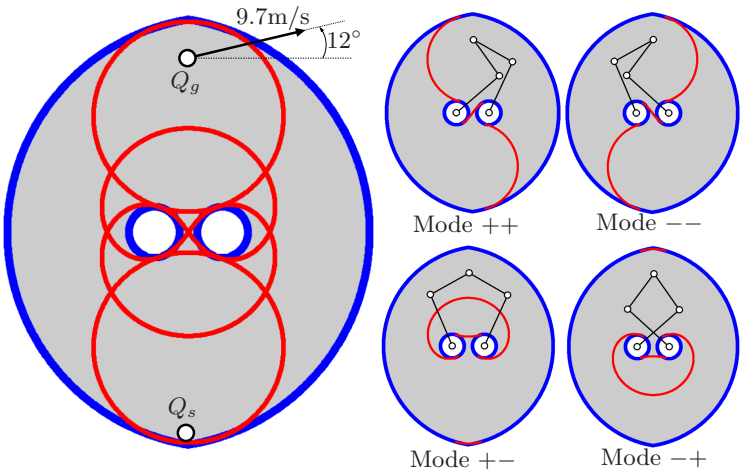

Fig. 3: Left: Forward singularities of the 5-bar robot (in red). Right: the same singularities, classified according to the different working modes.

ÉTS de Montréal [9] and Gridbot Technologies [1]. In our version, the base distance is $d = 0.6472$ [m], and the proximal and distal link lengths are $L = 1.14$ [m] and $l = 0.9$ [m], respectively. With these dimensions, the workspace of Q is the grey area shown in Fig. 3, left. The task to be planned consists in picking an object of 1 [kg] located at point Q_s, to later throw it from point Q_g with the shown velocity. To increase the difficulty of the task, we limit the motor torques to ± 15 [Nm], forcing the planner to generate pendulum-like motions that increase the kinetic energy progressively, until the required velocity at Q_g is achieved. The mass of each link is 0.5 [kg] and the moments of inertia of the proximal and distal links (with respect to their center of gravity) are of 0.0541 [kg·m^2] and 0.0338 [kg·m^2], respectively.

We recall from [9] that the forward singularities of this robot occur when the distal links get aligned, i.e., when q_3 is either 0 or π. The positions of Q for which such an event occurs are depicted in red in Fig. 3, left. From this figure it appears that the workspace is severely limited by the presence of forward singularities, but note that each position of Q can be attained by up to four inverse kinematic solutions of the robot. Each solution corresponds to one working mode identified by the signs of q_4 and q_2 in the range $[-\pi, \pi]$. If we separate the singularities according to these signs, larger singularity-free regions arise (Fig. 3, right).

Some existing planners require the use of only one working mode [11], but the full motion range of the robot can only be exploited when working mode changes are allowed. This is especially true in highly dynamic motions like the ones we require, in which the rapid movements of the robot, and its inertia, will frequently favor the mode changes. Our planner is beneficial in this respect, because it works in the state space manifold of the robot, on which the changes will occur in a natural way.

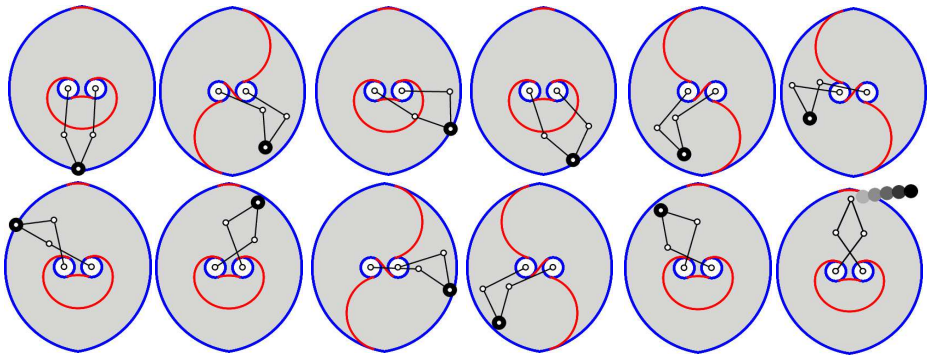

Fig. 4: Trajectory computed by the planner. Each snapshot depicts the forward singularities of the current working mode. See also youtu.be/ENKX0wrZAKM.

Fig. 4 shows the trajectory obtained by the planner, which lasts for 7.5 seconds, and takes 150 CPU seconds to be computed on average, on a Macbook Pro with an Intel i7 processor. Note how the robot begins to oscillate to the right, to later go left, climb to the top-most position, and finally complete a full revolution about the base to gain momentum before throwing the load. See the video in youtu.be/ENKX0wrZAKM for an animation. As we appreciate, many working mode changes occur during the move and the robot never crosses the forward singularity locus (the sign of q_3 is always kept constant). The video also shows that, if we plan the same task without singularity avoidance constraints (Eqs. (3) and (6)), we obtain trajectories that easily cross the singularities. As we mentioned in the introduction, this fact would hinder the control of the robot during task execution.

5 Conclusions

This paper has proposed a randomized planner to compute dynamic trajectories avoiding forward singularities. The planner relies on a system of equations characterizing the singularity-free state space of the robot, and on an ordinary differential equation describing the robot dynamics in such space (Eqs. (4) and (7), respectively). A great advantage of the planned trajectories is that they can be stabilized by means of simple computed-torque control laws [2], because the inverse dynamic problem always yields bounded actions in their vicinity. A second advantage is that, when executing the trajectories, it suffices to sense the positions and velocities of the actuators in order to keep track of the full robot state. The remaining coordinates can always be recovered with the forward kinematic maps, which never degenerate in non-singular configurations. Our current efforts are focused on validating the planned trajectories in real robots.

Acknowledgements Work partially funded by the Spanish Ministry of Economy and Competitiveness under projects DPI2014-57220-C2-2-P and DPI2017-88282-P.

References

1. RAPI-MOD: Ultrafast scara robot. URL http://gridbots.com/rapi_mov.html
2. Aghili, F.: A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: applications to control and simulation. IEEE Transactions on Robotics **21**(5), 834–849 (2005)
3. Bohigas, O., Henderson, M.E., Ros, L., Manubens, M., Porta, J.M.: Planning singularity-free paths on closed-chain manipulators. IEEE Transactions on Robotics **29**(4), 888–898 (2013)
4. Bohigas, O., Manubens, M., Ros, L.: Singularities of non-redundant manipulators: A short account and a method for their computation in the planar case. Mechanism and Machine Theory **68**, 1–17 (2013)
5. Bohigas, O., Manubens, M., Ros, L.: Singularities of robot mechanisms: Numerical computation and avoidance path planning. Springer (2016)
6. Bohigas, O., Zlatanov, D., Ros, L., Manubens, M., Porta, J.M.: A general method for the numerical computation of manipulator singularity sets. IEEE Transactions on Robotics **30**(2), 340–351 (2014)
7. Bordalba, R., Porta, J.M., Ros, L.: Randomized kinodynamic planning for cable-suspended parallel robots. In: Cable-Driven Parallel Robots, pp. 195–206. Springer (2018)
8. Bordalba, R., Ros, L., Porta, J.M.: Randomized kinodynamic planning for constrained systems. In: IEEE International Conference on Robotics and Automation (2018). To appear.
9. Bourbonnais, F., Bigras, P., Bonev, I.A.: Minimum-time trajectory planning and control of a pick-and-place five-bar parallel robot. IEEE/ASME Transactions on Mechatronics **20**(2), 740–749 (2015)
10. Briot, S., Arakelian, V.: Optimal force generation in parallel manipulators for passing through the singular positions. The International Journal of Robotics Research **27**(8), 967–983 (2008)
11. Cortés, J., Siméon, T., Laumond, J.P.: A random loop generator for planning the motions of closed kinematic chains using PRM methods. In: IEEE International Conference on Robotics and Automation, pp. 2141–2146 (2002)
12. Hill, R.B., Six, D., Chriette, A., Briot, S., Martinet, P.: Crossing type 2 singularities of parallel robots without pre-planned trajectory with a virtual-constraint-based controller. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 6080–6085 (2017)
13. Jiménez, P., Thomas, F., Torras, C.: 3D collision detection: A survey. Computers & Graphics **25**(2), 269–285 (2001)
14. LaValle, S.M.: Planning Algorithms. Cambridge University Press, New York (2006)
15. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. International Journal of Robotics Research **20**(5), 378–400 (2001)
16. Özdemir, M.: Removal of singularities in the inverse dynamics of parallel robots. Mechanism and Machine Theory **107**, 71–86 (2017)
17. Parsa, S.S., Boudreau, R., Carretero, J.A.: Reconfigurable mass parameters to cross direct kinematic singularities in parallel manipulators. Mechanism and Machine Theory **85**, 53–63 (2015)
18. Porta, J.M., Ros, L., Bohigas, O., Manubens, M., Rosales, C., Jaillet, L.: The Cuik Suite: Analyzing the motion of closed-chain multibody systems. IEEE Robotics and Automation Magazine **21**(3), 105–114 (2014)
19. Potra, F.A., Yen, J.: Implicit numerical integration for Euler-Lagrange equations via tangent space parametrization. J. of Struct. Mechanics **19**(1), 77–98 (1991)