# A hybrid collision model for safety collision control

Thibault Noël, Thomas Flayols, Joseph Mirabel, Justin Carpentier, Nicolas Mansard

# A hybrid collision model for safety collision control

T. Noël[1], T. Flayols[1], J. Mirabel[1], J. Carpentier[2], N. Mansard[1,3]

*Abstract*— **Self-collision detection and avoidance are essential for reactive control, in particular for dynamics robots equipped with legs or arms. Yet, only few control methods are able to handle such constraints, and it is often necessary to rely on path planning to define a collision-free trajectory that the controller would then track. In this paper, we introduce a combination of two lightweight, conservative and smooth models to generically handle self-collisions in robot control. For pairs of bodies that are far from one another on average (e.g. segments of distinct legs), we rely on a standard forward kinematics approach, using simplified geometries for which we provide analytical derivatives. For bodies that are moving close to one another, we propose to use a data-driven approach, with datasets generated thanks to a standard collision library. We then build a simple torque-based controller that can be implemented on top of any control law to prevent unexpected self-collision. This controller is meant to be implemented as a low-level protection, directly on the robot hardware. We also provide an open-source library to generate ANSI-C code for any robot model, experimented on the real quadruped Solo.**

## I. Introduction

Collision and self-collision avoidance is an important but yet difficult constraint in robotics [1], [2]. They are now handled at the planning level, by relying on random-based algorithms [3], [4] where only binary detection is needed [5], and efficient implementation are freely available [6]. Two difficulties prevent to dynamically handle collisions at the control level. First, in control, we need to compute the distance to collision, and not only a binary collision test [7]. Yet it has been several times observed that the distance between two general collision objects is a non-differentiable function [8]. This is because the witness points (i.e. points on the surface of each of the two collision objects that are the closest) may *jump* on the surfaces even for small relative motions of the pair. Only the strict convexity of the collision geometry can guarantee smoothness properties of the distance function.

To work with the exact robot geometry, a possible approach is to split the collision volumes into a union of strictly convex objects, and consider all the collision pairs resulting from this union [9], [10]. Yet the number of pairs drastically scales with the geometry complexity. In addition, control implies to compute collision distances and not only binary collision test, which is about 3 times more expensive [6].

[1] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France. [2] INRIA, Ecole Normale Superieure, CNRS, PSL Research University, 75005 Paris, France. [3] Artifical and Natural Intelligence Toulouse Insitute (ANITI), Toulouse, France. This work has been supported by the MEMMO European Union project within the H2020 Program under Grant Agreement No. 780684.

Companion video available at `https://peertube.laas.fr/videos/watch/b6e911ee-728f-4337-a76e-8138a332b583`

Another approach is to accept to approximate the collision geometries to enable better control properties or performances. A very classic and simple geometry is the so-called capsule [11], [12], i.e. a cylinder with spherical extremities [13], which is convex but not stricly convex. Capsules belong to the family of inflated volumes, that we discuss later in the paper. In [14], a strictly convex volume, composed of patches of sphere and torus, is proposed along with an algorithm to build it around a given mesh to be encapsulated, and used in control in [15] or trajectory optimization in [16]. Simplified models of collision can also be learned offline from the robot model or directly from trajectories of the system. Various machine-learning algorithms have been used to learn to predict collision (binary query) using either support vector machine (SVM) [17], neural networks (NN) [18], [19], Gaussian Mixture Models (GMM) [20]. In [21], collision and joint limits are learned together from motion-capture recordings of a human actor.

While computing a single collision distance is faster than machine-learned approximation, especially with a simple geometry, learned representations on all collision pairs of a robot manage to also capture what are the *active* pairs for which the distance should be considered, hence discarding the information and computational burden related to pairs that are too far away. On the other hand, simple geometries can be made as smooth as desired, and distance derivatives are also straight forward to compute. In [22], a hybridization is proposed to take advantage of both approaches. The authors proposed to combine simple encapsulating volumes and collision tables for complex joints. Yet the hybrid model was only considered for detecting collisions.

In this paper, we extend this idea of hybrid model, to compute distances and their derivatives and build a controller preventing collisions, illustrated on the quadruped Solo [23]. More precisely, we propose to rely on **simplified geometries to handle pairs of collisions that are far** from each other on average. On the quadruped robot for example, that would correspond to segments of distinct legs. We have chosen to use capsules, for which the distance computation is minimal as quickly reported in section II. We also provide the expression of the distance derivatives which are needed for any controller. The drawback to pay for the simplicity and the derivability is some conservatism (i.e. false positive detections and a reduced workspace) which however remains negligible compared to the total range of motion. Yet this conservatism is not acceptable for pairs of bodies moving very close to one-another. On the quadruped, that would be the upper-leg to torso collision. In general, this case covers any pairs on both sides of a complex joint, like shoulder

or wrist joints with a composed 2- or 3-DoF mechanism. To handle these cases, we propose to **rely on a data-driven representation**, trained on data easily generated from a collision-detection library. We propose in section III several representations, along with their derivatives and the method to train them. The complete self-collision model is then composed of a hybridisation between simple geometries and data-driven approximation. We empirically show in section V that our model is an appropriate approximation, detecting all true collisions and raising only a small ratio of false detections. We also show how to evaluate the model along with its derivatives at a low computational cost. It is then suitable to implement a self-collision safety controller at the lowest level. At the end of this section, experiments on the real robot Solo, presented in [23] and extended here in a 12-DoF version, demonstrate the interest of the approach to prevent collision, for example during fast locomotion where front and rear legs have to cross.

## II. Distance and derivatives for inflated geometries

### A. Mesh-to-capsule approximation

As a first step, relevant pairs can be approximated by simple geometries. As we want to use the collision model in our controllers, we select piece-wise smooth objects. We argue here that inflated volumes (i.e. the level set at equal given distance to a simple geometry) are a good trade-off between smoothness and lightness. For the sake of readability, the equations are described here for capsule pairs, but they can be directly extended to any other inflated volumes, e.g. sphere, swept-sphere rectangles or cubes, etc. Mathematically, the definition of a capsule, based on its generating segment $S$ is as follows :

$$C = \left\{p \in \mathbb{R}^3 \,|\, d(p,S) \leq \rho\right\} \tag{1}$$

The capsule geometry is thus completely characterized by its radius $\rho$ and the length of $S$. Capsule parameters (placement w.r.t. the parent joint, length and radius) can be selected to fit at best the real geometry (i.e. conservatively covering the mesh and minimizing the extra volume) for example by a non-linear optimization program. For Solo, we obtained 2 sets of capsules parameters (the lower legs are thinner than the upper legs), visualized along with the robot model on Fig. 1. The resulting loss of motion range is negligible, as quantified in section V.

### B. Distance

Consider two capsules $C_1$, $C_2$. We denote $\mathcal{F}_1$ and $\mathcal{F}_2$ two frames positioned at the center of the capsules with the Z-axis aligned with the capsule axis. The placement of the frames with respect to the world is computed from the joint configuration $q$ by forward kinematics. The idea of the distance computation is then to solve for the witness points on both segments and use them to get the distance. In order to compute those points, we first compute the closest points on the lines carrying the segments, and then apply the segments constraints before solving again and get

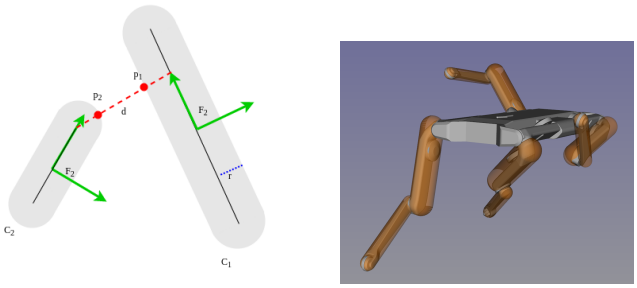


Fig. 1. Capsules notations (left) and approximations for Solo legs meshes (right) : the quadruped legs are properly approximated by capsules

the segments witness points $s_1, s_2$. The capsules distance is then $d(s_1,s_2) - (\rho_1 + \rho_2)$. The capsules witness points are directly derived from the segments ones: $p_1 = s_1 + \rho_1 n$ and $p_2 = s_2 - \rho_2 n$ with $n = \frac{s_2 - s_1}{\|s_2 - s_1\|}$.

### C. Derivatives

The distance between the 2 capsules is the distance between the 2 witness points $p_1$ and $p_2$. As $\overrightarrow{p_1 p_2}$ is normal to both capsules surfaces, then the variation of the distance between the capsules also is the variation of the distance between the witness points. In the following, we denote by resp. $J_{\mathcal{F}_1}$,$J_{\mathcal{F}_2}$ the frame Jacobians, corresponding to the frame movements resp. $\mathcal{F}_1$ and $\mathcal{F}_2$ ($6 \times n_v$ matrices, corresponding to translation and rotation). The Jacobian corresponding to the displacement of the first witness point $p_1$ is:

$$J_{p_1} = \frac{\partial p_1}{\partial q} = J^t_{\mathcal{F}_1} - p_1 \times J^R_{\mathcal{F}_1} \tag{2}$$

with $J^t_{\mathcal{F}_1}$ (resp. $J^R_{\mathcal{F}_1}$) denoting the translation (resp. rotation) part of $J_{\mathcal{F}_1}$. We can then derive the distance gradient which reads:

$$J_{dist} = \frac{\partial \overrightarrow{p_1 p_2}}{\partial q} = \frac{(p_1 - p_2)^T}{d}(J_{p_1} - J_{p_2}) \tag{3}$$

From (3), we see that the Jacobian is independent from the capsule radius. In practice, we can then directly work with the segment (i.e. the capsule of radius 0). We will see in the description of the proposed controller that we will work with a conservative threshold to activate the collision pair before the collision occurs, which happens to be equivalent to inflate the capsule radius of this threshold. We also see that the capsule distance is only piece-wise $C_1$ (i.e. its Jacobian is differentiable nearly everywhere), which is expected as the capsule is not strictly convex. But this is enough to produce a good controller, as shown in section V.
Distance and derivatives of the capsule are directly derived from the witness points of the segments. Likewise for other inflated volumes, the distance and derivatives can be derived from the witness points returned by any collision library.

## III. Data-driven collision model

Let us now consider the collision pairs for which the distances are small on average, e.g. the case of the shoulder

collision between the base and an upper leg. The previous approach is not suitable anymore here, because the mechanical assembly is such that the pair links are constrained to a very low distance in all configurations. No mesh approximation would be sufficient here. We thus need to define a new collision distance ensuring safe collision detection and affecting the range of motion as little as possible, while keeping good control properties. It might be considered to carefully chose another simplified geometry. We prefer however to rely on machine learning to automatically build a generic geometry representation based on data.

### A. Cartesian distance between close geometries

Let us consider a given pair of robot bodies, connected by a sub-part of the kinematic tree. We denote by $q_R \in \mathbb{R}^{NR}$ the corresponding sub-part of the robot configuration $q$. We recognize here that the dimension of $q_R$ is typically small ($NR = 2$ or $NR = 3$ on most of the robots we considered). For Solo for example, the shoulder articulation has 2 rotations, and the knee adds 1 more in the more advanced model (see Section IV). Thus, the collision of these pairs only occurs in low-dimensions sub-spaces of the robot joint space. From there, we can use a standard collision library [6] to sample the collision distance.

Yet the result is not directly satisfactory because collision algorithms and robot models are typically not designed for the accuracy required to handle such close pairs. The result for the front left shoulder pair of Solo is shown as example in Fig. 2. The obtained distance field clearly displays the collision zone ($d = 0$) corresponding to the angle range along $x$ for which the upper leg intersects with the base, and which depends on the $y$ rotation of the leg (extension towards front or back of the body). We can notice strong discontinuities near the collision zone, and unintuitive variations in the $d > 0$ zone. These properties of the distance make it difficult to approximate as is or to interpret from a control point of view. The most useful data we get from this sampling is actually the shape and localization of the collision surface in joint space (i.e. the submanifold in $\mathbb{R}^{NR}$ separating free space from collision space), which we now characterize further and use to define a new collision distance.

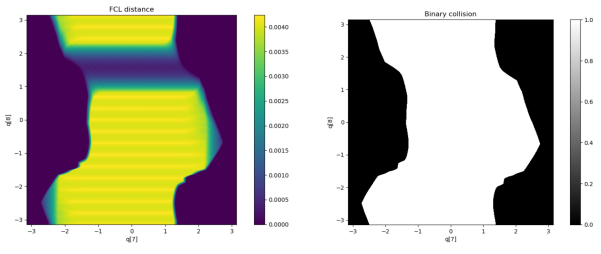


Fig. 2. Example of Solo shoulder collision data subspace ($x$ axis: $q_7$, shoulder $x$ rotation, $y$ axis: $q_8$, shoulder $y$ rotation). Left : Cartesian distance (meters), right : binary view. We observe a large flat area around $q_8 = 1.5$, corresponding to a hook-shape of the complex joint, which is always close to the leg but mechanically never colliding. Cartesian distance is hence not a suitable metric in such cases.

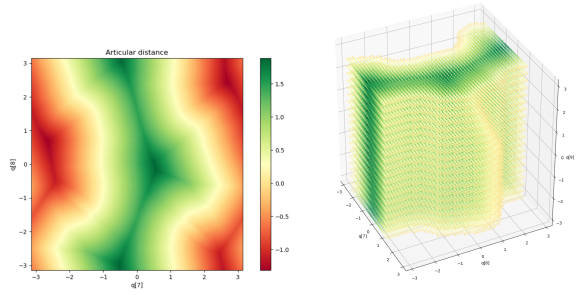


Fig. 3. Left: Shoulder collision distance $d_S$ after redefinition as minimal euclidean distance in 2D joint-space. Right: Taking the knee joint into account adds a third dimension to the distance field

### B. Sampling the distance in joint space

*1) Collision surface characterization:* Let us first define the collision surface in the joint space as the set of configurations for which the collision distance of the pair we consider is exactly 0 :

$$S = \{q_R \in \mathbb{R}^{NR} | \, d(q) = 0\} \tag{4}$$

It seems vain to try to get a closed-form representation of $S$ [1]. We thus proposed to sample it. As $S$ is an implicit submanifold of $\mathbb{R}^{NR}$ (i.e. of measure 0), we cannot directly sample it from $\mathbb{R}^{NR}$ but rather have to project random points from $\mathbb{R}^{NR}$ onto $S$.

We do so by sampling many random pairs of configurations in free space $q_{free}$ and in collision space $q_{coll}$. We then use dichotomy to compute the intersection of the segment $[q_{free}, q_{coll}]$ with the surface $S$ with a given precision. This method can lead to miss some parts of the collision surface if $S$ does not numerically contain enough points; however, the sampling of the configurations can also be chosen more carefully than at random when the shape of the collision surface is easy to interpret in 2 or 3D for example.

*2) Collision distance in joint space:* As the distance in Cartesian space is not satisfactory to represent the proximity of the two bodies, we propose to rather represent it by the distance in joint space of the configuration to the surface $S$:

$$d_S(q) = \min_{q_S \in S} \|q_R - q_S\| \tag{5}$$

In our case, we used the standard euclidean distance : $\|q_S - q_R\| = \sqrt{(q_R - q_S)^T (q_R - q_S)}$ We can naturally extend this definition to the collision domain, i.e. handle positive and negative distance. The complete definition of the collision distance is thus :

$$d_S(q_R) = \min_{q_S \in S} \{s_{coll} \|q_R - q_S\|\} \tag{6}$$

with $s_{coll}(q) = 1$ when $q_R$ is in free space and $-1$ otherwise. Note that $d_S$ is properly defined also when $S$ is a union of manifold, as it is the case for example with Solo, as shown in Fig. 3.

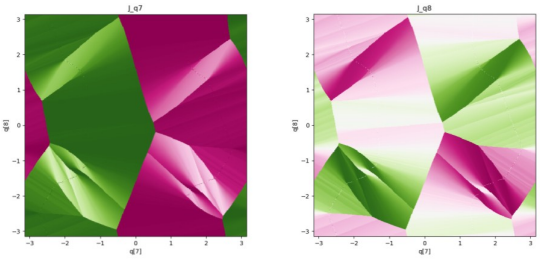

Fig. 4. $x$ and $y$ shoulder rotations components of the collision distance derivatives for Solo shoulder (i.e. $\frac{\partial d_S}{\partial q_7}$, $\frac{\partial d_S}{\partial q_8}$)

*3) Distance jacobian:* The gradient of $d_S$ is quite similar to the jacobian expression of section II:

$$J_S(q_R) = \frac{\partial d_S}{\partial q} = \frac{q_R - q_w}{d_S(q_R)} \tag{7}$$

where $q_w = \underset{q_S \in S}{\arg\min} \|q_R - q_S\|$ is the witness configuration. The distance derivatives for the shoulder of Solo are represented in Fig. 4. The visible irregularities, along the line of maximal distance for example, are due to the *jump* of the witness configuration when $q_R$ crosses this line. Similarly to Section II, the distance field is only piecewise $C_1$ on each side of such configurations surfaces where the witness configuration *jumps*.

### C. Distance approximation

We now have a properly defined collision distance $d_S$ and its derivatives, but its evaluation is inefficient at the moment. In order to use it in a real-time controller, we thus need a fast-evaluated approximation of this distance. The formulation we have chosen to introduce $d_S$ directly leads to a method to sample $d_S$ in the joint space $\mathbb{R}^{n_R}$, it can now be used to generate a training dataset for a data-driven approximation. We introduce three approximation models to represent $d_S$, that will be benchmarked in Section V.

*1) Fourier transform approximation:* Dealing with revolute joints and their natural periodicity, a reasonable idea is to use the $n$D Fourier transform to approximate the distance field sampled on a grid. For example in 2D, the Fourier transform of the distance on a $n \times m$ grid is :

$$D(u,v) = \sum_{j=-n/2}^{n/2} \sum_{k=-m/2}^{m/2} d(j,k)\, e^{-2i\pi(u\frac{j}{n} + v\frac{k}{m})} \tag{8}$$

Inverting this transform requires the evaluation of $2nm$ trigonometric functions. A classical approximation is thus to filter it and neglect the smallest coefficients to only evaluate the trigonometric functions for the coefficients of significant amplitude. The approximation of the derivatives of $d_S$ is also directly obtained from the same coefficients.

The main limit of this method is that for a given resolution, the number of Fourier coefficients needed to represent the distance (and thus the execution time), filtered or not, grows exponentially with the number of dimensions in the relevant subspace. Efficiency at runtime is problematic.

*2) Datasets generation:* For the following methods, we do not need grid-ordered data. The training datasets are generated with the method described in III-B. We also include the set $S$ of surface configurations in the training data, because this is the zone where the distance should be best approximated. Derivatives of the distance are also part of the dataset.

*3) Neural network approximation:* We now present a standard multi-layer perceptron approximation. The regular shape of the distance field and the relatively low number of inputs allow for good results with a small, quickly evaluated architecture (one hidden layer). The activation function can also be chosen to optimize computational efficiency. Another advantage is the possibility to use the same inference model to obtain a simple evaluation expression for both the distance and its Jacobian. The periodicity of the distance field is accounted for by using , instead of the shoulder configuration $q_R$, $x = [\cos q_R, \sin q_R]$. The loss used during the training is the usual RMSE of the distance. Concerning architecture, we used models with a tanh activation, and a hidden layer containing 6 to 48 neurons, while still easily satisfying the real-time evaluation constraint. A Sobolev training [24] was also tested, i.e. where the loss also inlcudes the error between the network derivatives and a groundtruth reference; however, the results were not included here because the method brought no significant improvement in our case.

*4) Support Vector Machine:* The datasets generated to train the neural network models were also used to train a SVM model. We trained it with a gaussian kernel, first as a classifier, expecting a distance representation to naturally appear. As the result was not satisfactory, we finally trained the SVM as a regressor with the same RMSE loss as the neural network. This approach is compared with the other approximations in Section V.

## IV. Controller

We used the proposed hybrid model to implement a torque-based collision controller as an additional safety controller on top of the main torque-based controller driving the robot. To that end, we first quickly describe the controller, based on the classical formulation of a repulsive potential field. This simple formulation allows us to implement it as a default safety layer directly on the robot hardware. We then describe the implementation using code generation.

### A. Control torque

We chose to mimic a visco-elastic spring-damper model to implement the virtual repulsive force applied along the collision normal. The amplitude of the force corresponds to:

$$f_r(q, v_q) = \begin{cases} 0 & \text{if } d \geq d_0 \\ -k_e(d(q) - d_0) - k_v \dot{d}(q, v_q) & \text{if } d < d_0 \end{cases} \tag{9}$$

where $d_0$ is the pair-dependant distance threshold under which the virtual force should be active, $k_e$ (respectively $k_v$) is the elastic (resp. viscous) gain characterizing the dynamics of the virtual spring, and $\dot{d}$ the time derivative of $d$ which can be re-expressed as $\dot{d}(q, v_q) = J_{coll}(q) v_q$. This choice for

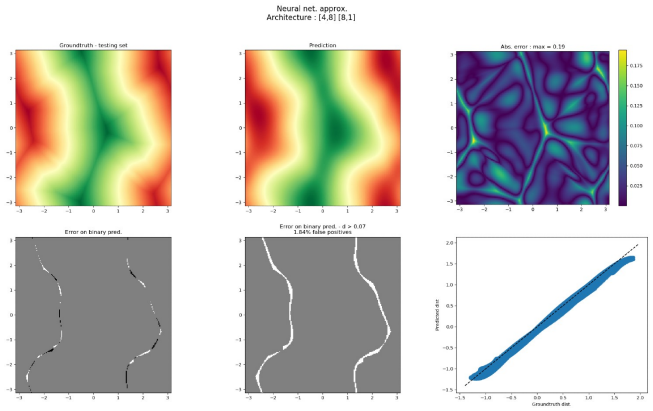


Fig. 5. Metrics illustrated in the case of a $[4, 8, 1]$-architecture, tanh activation neural network. From left to right and top to bottom : ground-truth distance field, approximated distance field, absolute error field, false positives and negatives, false positives after false negatives removal, predicted distance w.r.t. ground-truth distance

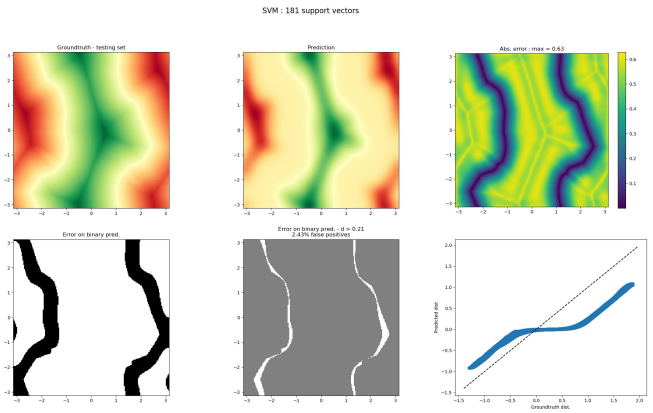


Fig. 6. Same metrics as in figure 5 for a gaussian kernel SVM with 181 support vectors. We observe a high numerical sensitivity around the collision surface.

the amplitude of the repulsive force can lead to a discontinuous control when $k_v \neq 0$; however, these discontinuities are numerically small compared to the maximum amplitude produced. Finally, we have to project the computed virtual force in joint space using the collision Jacobian. The final expression of the collisions avoidance control torque for one pair is :

$$\tau(q) = J_{coll}^T(q) f_r(q, v_q) \tag{10}$$

To compute the total control torque, we finally loop on all the relevant collision pairs in the model and sum the corresponding torques. The controller presented here is very simple by design choice, to allow for an implementation on the lowest-level control hardware. Our collision model could also be integrated with task-space inverse dynamics [25], [26], [27] or an optimal control solver [28], [29], [30].

### B. ANSI-C code generation

The computation of the control torque only requires the distance and derivatives provided by the collision model. The model itself also relies on a fast forward kinematics implementation, and an offline neural network training library, both open-source [31], [32]. In an effort to produce a fully-embedded module able to run on the robot hardware, we used the code generation capabilities offered by open-source libraries to re-implement the collision model, initially written in C++, as an agnostic ANSI-C library with reduced development effort. An additional step of code generation from Python to C++ is needed for the neural network model. The code will be released with the final version of the paper.

## V. Experimental results

In order to evaluate the proposed model and its efficiency to control the robot, we first adapt the data-driven model for conservatism. Then, we measure the quality of the approximation, and score the efficiency of its evaluation. Finally, we report the behavior of the controller on the real Solo.

### A. Joint-space approximation benchmark

We want to compare the shoulder approximation methods on a common basis to chose the best one. First, we want to adjust the parameters of each approximation model to make them conservative, i.e. ensure no false negatives in terms of collision detection. We do so by defining a modified collision distance $\tilde{d_S}$ so that $\tilde{d_S}(q) = d_S(q) - d_0$. This is equivalent to shifting the collision surface along its local normal direction. The distance offset $d_0$ can be chosen for each learning model to minimize the number of false positives while forbidding false negatives. After training, the different methods are evaluated with the following metrics :

- False positives : after removing the false negatives, the false positives, for a homogeneous dataset sampling, are a very good proxy for the loss of range of motion due to the approximation. This metric reflects the quality of

TABLE I

Metrics comparison for the different joint-space distance approximation methods. Execution times are also given for the capsule approximation and the complete model.

| **Approx. method** | Test dataset avg. RMSE | Lost range of motion (%) | Avg. exec. time (s) |
|---|---|---|---|
| **SVM** | | | |
| 313 support vectors | - | 1.5 | - |
| 47 support vectors | - | 7.61 | - |
| **Fourier transform** | | | |
| 112 coeffs | - | 3.6 | 11.3 |
| 47 coeffs | - | 3.8 | - |
| 26 coeffs | - | 11.2 | 2.7 |
| **Neural network - 2D** | | | |
| [4,8,1] | 0.0358 | 2.475 | 1.1 |
| [4,20,1] | 0.010 | 0.78 | - |
| **Neural network - 3D** | | | |
| [6,18,1] | 0.034 | 1.18 | - |
| [6,48,18,1] | 0.012 | 0.68 | 8.4 |
| **Caps. geom. model** | | | |
| 1 caps. pair | - | - | 1.3 |
| 20 caps. pairs | - | - | 5.5 |
| Full model (NN + caps.) | - | - | *9.8 - 38.9* |

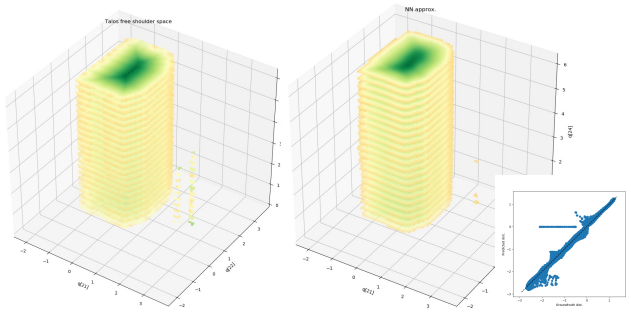


Fig. 7. Left : Joint-space distance (negative part masked) for the Talos shoulder joint. The free configurations outside the main free zone correspond to mechanically forbidden configurations with the shoulder contained inside the torso. Right : (top) neural network appoximation of the distance field and (bottom) corresponding distance predictions w.r.t groundtruth distances.

the boolean collision detection.

- RMSE : loss used for neural network training, which reflects the relative quality of the collision distance approximation in the whole space
- Execution time

Graphical representations of those metrics in the case of the Solo shoulder in 2D are shown for the NN on Fig. 5 and for the SVM on Fig. 6. From the more extensive results presented in Table I, we can conclude that the trade-off between approximation quality and fast evaluation is best handled by the neural network approach, which will be implemented in our final controller. We also validated the model by approximating the geometry of the shoulder (3D) of a Talos humanoid robot [33]. Due to lack of space, only partial results are reported in Fig. 7.

### B. Solo experiments

We present the experiments conducted with the controller integrated in the robot control loop on the Solo platform. The parameters of the collision avoidance control torque chosen for the experimental setup are $k_e = 60$ N.m$^{-1}$, $k_v = 0$ N.m$^{-1}$.s$^{-1}$ and $d_0 = 0.05$ m. The video attached shows the live experiments on the robot along with the collision data visualized in a virtual environment. The results are summarized in the video `https://peertube.laas.fr/videos/watch/b6e911ee-728f-4337-a76e-8138a332b583`.

In order to better demonstrate the capability of our controller as a safety feature for additional control strategies, we first run it as the only active controller and manipulate the legs manually with the robot on a stand (motion 1 of the video). This also allows us to reach configurations for which we know the collision detection can be more difficult and easily test the limits of the controller. We then use a PD tracking a reference trajectory which contains colliding configurations. The reference trajectory for the PD-tracking is an ellipse described by the robot feet in the $(x, y)$ Cartesian plane. To test the capsule model case, we set the two front feet to follow such intersecting ellipses (motion 2 of the video). For the shoulder, we can force a collision

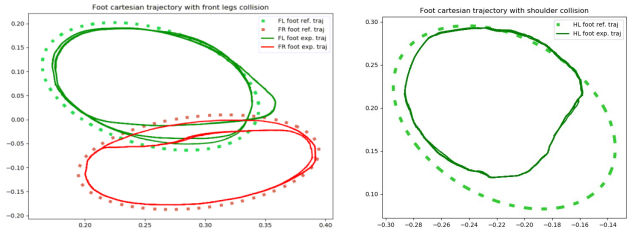


Fig. 8. Reference trajectory tracking with active collision avoidance : the dashed line represents the reference trajectory, and the full line is the experimental result on Solo for (left) a capsule pair collision i.e. motion 2 of the video, (right) a shoulder collision i.e. motion 3 of the video.

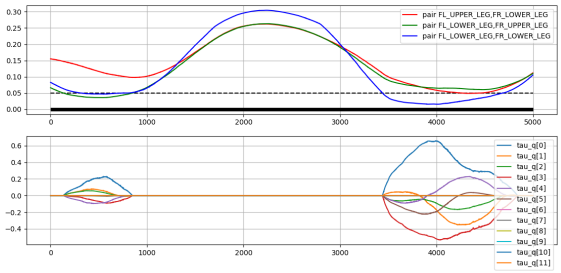


Fig. 9. (Top) Distance threshold violation and (bottom) avoidance torque for a capsule collision (motion 2 of the video).

by setting the ellipse plane close to the body plane for one foot (motion 3 of the video). The reference trajectories in Cartesian space and the experimental trajectories with active collision avoidance are shown in Fig. 8. We also show an example of distance threshold violation and corresponding repulsive torques for the capsule model in Fig. 9.

## Conclusion

We presented a hybrid self collision model capable of estimating complex-joint limits and body-to-body collisions of a rigid multi-body articulated robot in real time. The proposed method is able to handle collisions of bodies which are naturally distant from one another as well as collisions between links forming hook-shaped joints. Our approach is based both on Cartesian distances between conservative inflated-volume approximations of body geometries and joint-space distances of complex joints via data-driven neural networks. The smooth evaluation of collision distances and their derivatives allow us to use them in a collision avoidance control algorithm while marginally reducing the robot workspace. Those quantities are of major importance for high-level controllers but can also be a crucial information for last-resort safety controllers. We proposed a simple collision avoidance controller which applies a visco-elastic repulsive force, moving away the robot links from imminent collision. Using automatic code generation, our hybrid collision model can be embedded on limited low-level robot controller hardware. We demonstrated and evaluated the safety controller capabilities on the quadruped platform Solo.

While this model can already serve as a low-level safety

controller preserving the robot hardware from a destructive high-level control output, we would like to use it in higher-level controller such as whole-body controllers, and model predictive controllers to better enforce feasibility of the computed motions.

## REFERENCES


[1] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[2] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[3] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *The International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[4] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[5] M. Lin, D. Manocha, and J. Canny, "Fast contact determination in dynamic environments," in *IEEE International Conference on Robotics and Automation*, 1994, pp. 602–608.

[6] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 3859–3866.

[7] C. J. Ong and E. G. Gilbert, "Fast versions of the gilbert-johnson-keerthi distance algorithm: Additional results and comparisons," *IEEE transactions on robotics and automation*, vol. 17, no. 4, pp. 531–539, 2001.

[8] A. Escande, S. Miossec, and A. Kheddar, "Continuous gradient proximity distance for humanoids free-collision optimized-postures," in *IEEE-RAS International Conference on Humanoid Robots*, 2007.

[9] J. Kuffner, K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue, "Self-collision detection and prevention for humanoid robots," in *IEEE International Conference on Robotics and Automation*, 2002.

[10] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond, "A local collision avoidance method for non-strictly convex polyhedra," *Proceedings of robotics: science and systems IV*, p. 33, 2008.

[11] C. Dube, M. Tsoeu, and J. Tapson, "A model of the humanoid body for self collision detection based on elliptical capsules," in *IEEE International Conference on Robotics and Biomimetics*, 2011.

[12] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[13] M. Lin and S. Gottschalk, "Collision detection between geometric models: A survey," in *Proc. of IMA conference on mathematics of surfaces*, vol. 1, 1998, pp. 602–608.

[14] A. Escande, S. Miossec, M. Benallegue, and A. Kheddar, "A strictly convex hull for computing proximity distances with continuous gradients," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 666–678, 2014.

[15] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, 2012.

[16] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1104–1119, 2013.

[17] J. Pan and D. Manocha, "Efficient configuration space construction and optimization for motion planning," *Engineering*, vol. 1, no. 1, pp. 046–057, 2015.

[18] I. García, J. D. Martín-Guerrero, E. Soria-Olivas, R. J. Martínez, S. Rueda, and R. Magdalena, "A neural network approach for real-time collision detection," in *IEEE International Conference on Systems, Man and Cybernetics*, 2002.

[19] Y. J. Heo, D. Kim, W. Lee, H. Kim, J. Park, and W. K. Chung, "Collision detection for industrial collaborative robots: a deep learning approach," *IEEE Robotics and Automation Letters*, 2019.

[20] J. Huh and D. D. Lee, "Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees," in *IEEE International Conference on Robotics and Automation*, 2016.

[21] Y. Jiang and C. K. Li, "Data-driven approach to simulating realistic human joint constraints," in *IEEE International Conference on Robotics and Automation*, 2018.

[22] K. Okada and M. Inaba, "A hybrid approach to practical self collision detection system of humanoid robot," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 3952–3957.

[23] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wuthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, and et al., "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, p. 36503657, 2020. [Online]. Available: http://dx.doi.org/10.1109/LRA.2020.2976639

[24] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. wirszcz, and R. Pascanu, "Sobolev training for neural networks," 2017.

[25] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

[26] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 981–988.

[27] A. Del Prete and N. Mansard, "Robustness to joint-torque-tracking errors in task-space inverse dynamics," *IEEE transactions on Robotics*, vol. 32, no. 5, pp. 1091–1105, 2016.

[28] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[29] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[30] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2536–2542.

[31] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *SII 2019 - International Symposium on System Integrations*, Paris, France, Jan. 2019. [Online]. Available: https://hal.laas.fr/hal-01866228

[32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[33] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, J.-P. Laumond, L. Marchionni, H. Tome, and F. Ferro, "TALOS: A new humanoid research platform targeted for industrial applications," in *International Conference on Humanoid Robotics, ICHR, Birmingham 2017*, ser. IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids),. Birmingham, United Kingdom: IEEE, Nov. 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01485519