Optimal Motion Planning for Humanoid Robots

Antonio El Khoury, Florent Lamiraux, Michel Taïx

Optimal Motion Planning for Humanoid Robots

Antonio El Khoury*, Florent Lamiraux* and Michel Taïx*
*CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France
Université de Toulouse ; UPS, INSA, INP, ISAE ; UT1, UTM, LAAS ; F-31077 Toulouse Cedex 4, France.
aelkhour@laas.fr

***Abstract*—This paper aims at combining state of the art developments of path planning and optimal control and to create the algorithmic foundations to tackle optimal control problems in cluttered environments. Our contribution is three-fold: first, we describe a simple method to automatically generate minimum bounding capsules around exact robot body geometries represented by meshes. Second, we use the bounding capsules to implement distance constraints for an optimal control problem solver and achieve (self-)collision avoidance. Finally, we propose a complete two-stage framework for optimal motion planning on complex robots. This framework is successfully applied to generate optimal collision-free trajectories both in simulation and on the humanoid robot HRP-2.**

I. Introduction

The generation of the best possible trajectory that does not violate any constraints imposed by the environment is an ubiquitous task in both industrial and humanoid robotics.

Numerous examples of successful robotic applications in the domains of motion planning and optimal control can be encoutered in literature and industry. Very few however, if none, consider the more general problem of optimal motion planning for complex robots evolving in complex environments.

This paper aims at combining state of the art developments of these two domains and to create the algorithmic foundations to tackle optimal control problems in cluttered environments.

II. Related Work

There are two established but still quite separate research areas that both address a part of the optimal motion planning problem, namely path planning and numerical optimal control.

A. Path Planning

Let the configuration q be a vector containing the generalized coordinates of a robot, and let $\mathcal{C}$ be the set of all possible configurations q. Path planning is mainly interested in the determination of a feasible path P connecting a start configuration q_s to a goal configuration q_g in $\mathcal{C}$. P is said to be feasible iff for all $q \in P$, q is valid with respect to both geometric constraints, e.g. (self-)collision avoidance, and kinematic constraints of the robot, e.g. joint limits.

Sampling-based algorithms, such as Rapidly-exploring Random Trees (RRT) [1], are particularly powerful when it comes to solving path planning problems in highly dimensional $\mathcal{C}$ and cluttered environments. In its simplest form, this algorithm expands a tree $\mathcal{T}$ rooted in q_s by first sampling a random configuration q_{rand} in $\mathcal{C}$, then extending an edge e from the nearest configuration q_{near} in $\mathcal{T}$ towards q_{rand} and finding a valid configuration q_{new}. This process is iterative, and it stops once $\mathcal{T}$ and q_g can be connected.

It is interesting to note that all nodes and edges of the final tree $\mathcal{T}$ lies in $\mathcal{C}$. This can lead to infeasible paths if the robot must respect additional kinematic constraints. Such is the case for robots with a floating base, i.e. the configuration contains an unactuated part that can be changed only through maintaining contact with the environment. [2], [3], [4] propose variants of RRT where the extension phase produces an edge that lies in its entirety on a manifold $\mathcal{M}$ of $\mathcal{C}$. $\mathcal{M}$ is defined by kinematic constraints, each defined by a feature value in the operational space, e.g. end-effector position and orientation. The constraint RRT algorithm produces a path P such that for all $q \in P$, q lies on the manifold $\mathcal{M}$.

One of the most important features of sampling-based algorithms is their probabilistic completeness, i.e. their capacity to avoid falling into local minima and find a solution path if it exists. They present however two major drawbacks. First, due to their sampling nature, the configuration q might move in a random fashion along the path P, which could lead to unusually long and unnatural paths. Second, we still need to apply a time parameterization in order to transform the path into a trajectory. This is a non-trivial task in the particular case of a humanoid robot, as we must ensure its dynamic balance along the trajectory. Additional processing is thus needed to provide a reshaped collision-free trajectory that can be executed on the robot.

B. Numerical Optimal Control

Given a dynamic model, let:

- t be the time variable,
- x be the state vector,
- u be the control vector,
- t_f be the trajectory duration,
- ϕ be the scalar objective function to minimize,
- f be the differential equation of the model,
- g be the equality constraint vector function,
- h be the inequality constraint vector function,
- r be the boundary conditions vector function.

An optimal control problem can be written as follows:

$$\min_{x(\cdot),u(\cdot),t_f} \int_0^{t_f} \phi(x(t),u(t))dt \tag{1}$$

subject to:

$$\begin{array}{rcl} \dot{x}(t) & = & f(t,x(t),u(t)) \\ g(t,x(t),u(t)) & = & 0, \\ h(t,x(t),u(t)) & \geq & 0, \\ r(x(0),x(t_f)) & = & 0. \end{array} \quad (2)$$

Starting from an initial value of x and u, numerical optimal control techniques are capable of iteratively converging towards a trajectory that locally minimizes ϕ while taking into account the dynamics of the system and verifying equality and inequality constraints given by Eq. 2.

There exists several classes of optimization solvers, among which gradient-based methods such as interior point optimizers implemented notably in IPOPT [5], and Sequential Quadratic Program (SQP) solvers. More particularly, multiple shooting optimization, implemented in the software package MUSCOD-II [6], is a very powerful tool that allows to solve large-scale optimal control problems thanks to a specially tailored SQP.

It is interesting to note that for all gradient-based solvers, the objective and constraint functions can be nonlinear, but they must be at least C^1, i.e. continuously differentiable. This requirement can be alleviated by the use of non gradient-based solvers such as STOMP [7], but this method has been only applied to simple problems with no equality constraints so far.

Optimal control techniques have been successfully applied on humanoid figures [8] and humanoid robots [9], [10]. The problems take into account complex robot dynamics, non-coplanar contact points and actuator limitations. However, current formulations work under the assumption the initial guess is collision-free; this initial guess is usually set by hand by the user, which can be a non-trivial task in complex environments. Furthermore, only some analytical constraints are added to prevent collisions. This is obviously not enough if we want to guarantee that the final trajectory never leads to collisions with either the environment or bodies of the robot. Hence, to the authors knowledge, initial trajectory generation and collision avoidance have yet to be treated properly in the optimal control formulation.

C. (Self-)Collision Avoidance Constraints

There are several ways to express collision avoidance constraints between two bodies; one could express the constraint with the exact distance between the exact polyhedral geometries such as in [11]. While distance computation is precise and relatively efficient thanks to bounding volume tree structures, the returned distance is zero in case of penetration, which forbids the computation of a correct constraint gradient. Moreover the distance constraint between two non-strictly convex polyhedra is not C^1, which can cause the optimization solver to behave incorrectly. [12] proposes a nice solution to this problem and introduces Sphere-Torus Patches Bounding Volumes (STPBV), which are strictly convex bounding volumes of the robot bodies; in this case the distance between a STPBV and a convex mesh is always C^1. These constraints are successfully used for self-collision avoidance in real-time control of a humanoid robot [13]. In [14], also in a control context, self-collision constraints are expressed as the distance between bounding capsules, i.e. cylinders capped by half-spheres.

D. Sampling-Based Optimal Motion Planning

Recently [15] introduced RRT*, a new variant of RRT that has the property of asymptotically converging towards the global minimum solution path. This is a very interesting approach as only one algorithm is needed to achieve optimal motion planning. But if we want to generate an optimal trajectory, we need to explore not only the configuration space $\mathcal{C}$, but the whole state space $\mathcal{S}$ where an element of $\mathcal{S}$ is $x = [q, \dot{q}]^T$. This doubles the dimension of the problem. Another problem arises from choosing the correct metric and local optimal policy for the extension phase. [16] propose a variant called LQR-RRT*. This variant uses local linearization of a system to derive a coherent extension method. Results have so far been only obtained on simple systems. Whether this approach can reasonably scale up to complex systems such as humanoid robots remains to be tested.

III. Contribution

To the authors best knowledge there is no algorithmic approach available that allows to address the problem of optimal motion planning for very complex dynamic robot systems in cluttered environments.

Our contribution is three-fold: first, we describe a simple method to automatically generate minimum bounding capsules around exact robot body geometries represented by meshes. Second, we use the bounding capsules to implement distance constraints for the solver and achieve collision and self-collision avoidance. Finally, we propose a complete two-stage framework for optimal motion planning on complex robots. Given a start and goal task or configuration, the constrained path planner produces a collision-free path where kinematic constraints have been enforced, thus avoiding geometric local minima. This path is given as an initial guess to the MUSCOD-II solver, which generates a locally optimal trajectory while enforcing dynamic constraints and (self-)collision avoidance. This framework is successfully applied to generate optimal collision-free trajectories both in simulation and on the humanoid robot HRP-2.

Section IV describes the path planning stage. The collision avoidance constraints are tackled in section V, and are used in the optimal control problem described in section VI. Finally, section VII showcases preliminary simulation and experimental results on the robot HRP-2.

IV. Constrained Path Planning

We use the constrained planner in [2], which is implemented with the motion planning library KineoWorks™[17]. This planner allows to generate a collision-free path, while guaranteeing that the solution path lies on a manifold of the configuration space. We want to generate for HRP-2 a

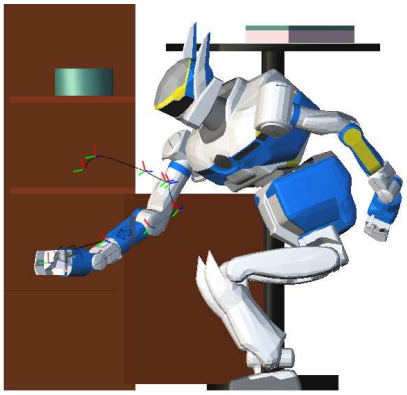


Fig. 1: Goal config generated on the goal submanifold $\mathcal{M}_g$ and a path found by the constrained path planner in a shelves environment.

collision-free path that guarantees its quasi-static balance when it is standing on both feet. We then define the manifold $\mathcal{M}$ with the following stack of equality constraints:

1) Right foot has a fixed 6D transformation,
2) Left foot has a fixed 6D transformation,
3) Center of mass vertical projection lies in the center of the support polygon.

Additionally, we would like to avoid choosing a single goal configuration q_g, but instead define a goal task x_g. This task can be defined by a sub-manifold $\mathcal{M}_g$ of the planning manifold $\mathcal{M}$. For a simple object manipulation task, $\mathcal{M}_g$ can be defined with the following stack of equality constraints:

1) Right foot has a fixed 6D transformation,
2) Left foot has a fixed 6D transformation,
3) Center of mass vertical projection lies in the center of the support polygon.
4) Gripper has the same 3D position as the object to grab.
5) Gripper thumb is oriented vertically.

Given a start configuration q_s a planning manifold $\mathcal{M}$ and a goal sub-manifold $\mathcal{M}_g$, we first create a set of goal configurations q_g by sampling a fixed number of configurations in $\mathcal{M}_g$, then we solve the path problem from q_s to q_g. The constrained planner diffuses exploration trees from q_s and each configuration of q_g, and stops once at least one of the goal configurations is in the same connected component as q_s. A shortcut optimizer can then be called to prune unnecessary waypoints and smooth the solution path.

Figure 1 shows a concrete example where HRP-2 has to grab an object on the lower shelf and place it on the upper shelf.

V. (Self-)Collision Avoidance Constraints

In path planning, collision queries are used to validate or invalidate configurations and hence the whole path. This is obviously not enough in optimal control, especially for gradient-based solvers where constraint must be C^1. Furthermore we need to return negative values of the distance in case of collision to provide the solver with the means to get out of collision.

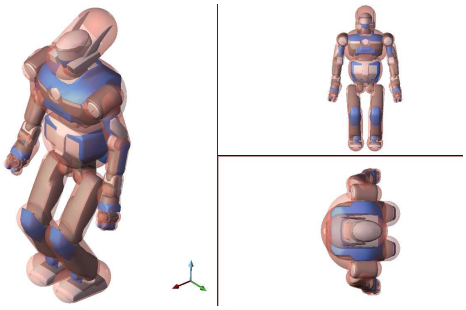


Fig. 2: Best fitting capsules for HRP-2

We must first define a bounding geometry of the real geometry of the robot bodies, and we choose to use bounding capsules, like in [14]. Capsule are basically sphere swept segments; to compute the distance between two capsules, we simply need to compute the distance between the capsule axes, then subtract their radii. Similarly, we can compute the distance between a capsule and a polyhedron in the environment by first computing the distance between the capsule axis and the polyhedron, then subtracting the radius. The returned distance can then become negative in case of collision.

A. Computing minimum bounding capsules

In [14], bounding capsules parameters, i.e. the 2 endpoints e_1, e_2 and the radius r, were set by hand to obtain the best fitting capsules around the body geometries. We propose here to automatically find these parameters by solving, offline and once for each body of the robot, the following optimization problem:

$$\min_{e_1,e_2,r} \quad \|e_2 - e_1\|\pi r^2 + \frac{4}{3}\pi r^3 \tag{3}$$

subject to:

$$r - d(p, e_1e_2) \quad \geq \quad 0, \text{ for all } p \in \mathcal{P} \tag{4}$$

where $d(p, e_1e_2)$ is the distance of p to line segment e_1e_2. Equations 3 and 4 mean we want to find the minimum-volume capsule while ensuring all points p of the underlying polyhedron $\mathcal{P}$ lie inside the capsule. HRP-2 has 41 bodies, each body polyhedron containing about 1000 points. We solve all 41 optimization problems with RobOptim [18] and the IPOPT solver [5] in less than 5s. Figure 2 shows the best fitting bounding capsule superimposed on the robot geometries.

B. Distance computation

As mentioned previously, for capsule-capsule distance pairs, we need to compute the distance between the two capsule axes then subtract the radii to obtain the real distance. We rely on the Wild Magic geometric library [19] to compute this distance in an average time of 2 μs.

Concerning capsule-polyhedron pairs, we rely on an implementation of OBB-Trees in the Kineo Collision Detection (KCD) library [17] for an efficient computation. For the environment in figure 1, the distance for one capsule-environment pair takes about 500 μs to be computed.

C. Distance pairs computation

If we take into account all capsule-capsule pairs of the robot, we end up with $\frac{n(n-1)}{2}$ possible pairs, with n the number of bodies. This means that for a robot like HRP-2, we can have up to 780 pairs and it can be very costly to evaluate the distance for all of them. Luckily, some bodies are either always colliding because they are adjacent in the kinematic tree, or never colliding due to the joint limits. This means that the pairs corresponding to those bodies can be safely pruned. We use the tool [20], which relies on finely exploring the configuration space and keeping track of colliding bodies, to save 510 useful pairs. Furthermore, since we are in the particular case of double-support motion, we can be sure that most of the leg bodies cannot collide with each other due to the additional kinematic constraints. We finally end up with 327 capsule-capsule pairs that must be all evaluated to guarantee self-collision avoidance. Similarly, we can prune some of the capsule-environment pairs and keep 23 pairs.

D. Distance gradient computation

The MUSCOD-II evaluates gradients through internal numerical differentiation, so gradients do not need to be explicitly provided to the solver. Nevertheless constraints must be C^1 to ensure proper convergence. As pointed out in [12], non-C^1 distance constraints can be a problem in the case of posture optimization. Capsules are convex but not strictly convex geometries; the distance for a capsule-capsule pair is C^1 everywhere except when the two capsule axes are exactly parallel, where a discontinuity in the gradient is seen. Due to the rarity of such an event in the context of trajectory optimization, we have not witnessed any misbehavior of the solver when using capsule geometries.

Regarding capsule-environment distance pairs, since we make no assumption about the obstacles convexity, we can be sure that the distance will not be C^1 whatever the bounding geometry of the robot body. Again, in optimal control, the solver still seems to behaves correctly.

VI. Description of Optimal Motion Planning Framework

Now that we have properly set distance pairs, we can establish a complete formulation of the optimal control problem for the second stage of our optimal motion planning framework.

A. Optimal Control Problem Formulation

1) Objective function: We choose to minimize, for a fixed duration, the integral over time of the sum of square jerks, as this criterion leads to smooth and natural trajectories.

The objective function can then be written as:

$$J = \int_0^{t_f} \dddot{q}(t)^T \dddot{q}(t) dt \tag{5}$$

and we define the state and control variables to be:

$$\begin{array}{rcl} x(t) & = & [q(t), \dot{q}(t), \ddot{q}(t)]^T \\ u(t) & = & [\dddot{q}(t)]^T \end{array} \tag{6}$$

2) Equality and inequality constraints:

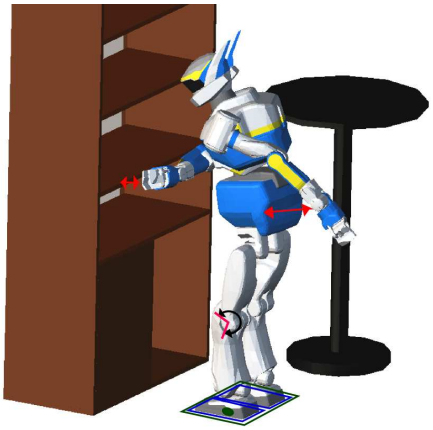


Fig. 3: Constraints added in OCP: support foot positions (blue), joint limits (purple), ZMP constraint (green), (self)-collision avoidance (red).

a) Joint constraints: Each actuated joint is subject to physical limitations of its underlying actuator and mechanical structure. Box constraints on angular, speed and torque limits are then added as:

$$\begin{array}{rcccl} q_{min} & \leq & q & \leq & q_{max} \\ \dot{q}_{min} & \leq & \dot{q} & \leq & \dot{q}_{max} \\ \tau_{min} & \leq & \tau & \leq & \tau_{max} \end{array} \tag{7}$$

b) Dynamic balance: The robot is submitted in our case to multiple coplanar contact reaction forces from the ground. We can then express the dynamic balance constraint thanks to the Zero-Moment Point (ZMP) [21], which has to remain inside the robot support polygon defined by its feet.

These constraints can be written as:

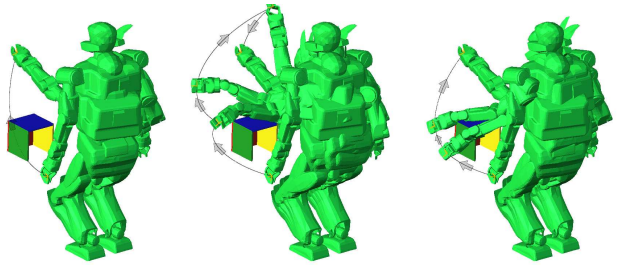
$$\begin{array}{rcl} p_{\mathcal{B}_c}(t) & = & p_{\mathcal{B}_c}(0) \\ zmp & \in & \mathcal{P}_{sup}, \end{array} \tag{8}$$

where $p_{\mathcal{B}_c}$ is the 6D position of the contact body $\mathcal{B}_c$, zmp and $\mathcal{P}_{sup}$ are the ZMP coordinates and the support polygon respectively.

c) Collision avoidance constraints: We use the capsule-capsule and capsule-environment pairs defined in V. Given a configuration q of the robot, we check that distances for pairs of bodies and pairs of body and obstacle are positive to ensure (self-)collision avoidance. We first tried to add one constraint per pair, which added up to $(327+23)*n_{ms}$ constraints, where n_{ms} is the number of multiple shooting nodes in MUSCOD-II. This led to poor performance as the solver systematically went beyond the threshold number of iterations. We hence propose to group all pairs for a single body and compute the minimum distance over those pairs as follows:

$$\min_{d_{\mathcal{B}} \in \mathcal{D}_{\mathcal{B}}} d_{\mathcal{B}} \quad \geq \quad 0, \text{ for all } \mathcal{B} \tag{9}$$

where $d_{\mathcal{B}}$ is the distance for one distance pair of the body $\mathcal{B}$.



(a) Initial invalid path. (b) RRT path. (c) Shortcut path.

Fig. 4: Paths for the test case.

B. Considerations for the Solver

With a complete formulation of the optimal control problem, the solver starts from an initial value of q(t) and u(t) and converges iteratively towards the locally optimal solution. It is then obvious that the initial guess plays an important role in the successful determination of the final solution and the convergence speed.

The constrained path planner generates a collision-free path where the kinematic constraints are enforced, but we still need to apply a time parameterization before feeding it to the optimization solver. We want to minimize the sum of square jerks; [22] shows that an unconstrained minimum-jerk trajectory is a polynomial of degree 5 which can be explicitly computed if the initial and final states are known. We choose then to place minimum-jerk trajectories between each pair of waypoints of the path, assuming they start and end at zero velocity and acceleration. This allows us to make sure that the configuration $q(t)$ follows exactly the solution path and that collision avoidance constraints are not violated in the initial guess.

In order to solve a finite-dimension optimal problem, we need to discretize both controls and constraints. We thus define the control u(t) as a continuous piecewise linear function over 20 sub-intervals of the whole trajectory duration. In the case of MUSCOD, the same discretization applies for the constraints, which means they are only enforced on each of the sub-intervals ends. However, this does not appear to be a problem for the constraints we use in our particular formulation, as shown in the next section.

VII. Results

We demonstrate the effectiveness of our optimal motion planning framework by first using it in a a simple test case example, then applying it to generate feasible motions on the robot HRP-2. All tests were run on a computer with a 1.6 GHz Intel® Core™2 Duo processor.

A. Test Case

Figure 4a shows the motion planning problem to be solved: HRP-2 starts from its rest position and moves to a goal configuration by raising its left arm. A concave object is placed such that the left hand is at one point enclosed in it if the initial path connecting the start to goal configuration is executed. This is a typical example of a problem with a local minimum defined by the environment, where a real-time control approach in task-space might fail. Figure 4b shows a possible solution path found with constrained RRT. This path can be shortened with a shortcut optimizer, as in figure 4c.

TABLE I: Test Case Computation Times

Initial guess	Initial path	RRT path	Shortcut path
Planning time (s)	–	5	5
Shortcut time (s)	–	–	4
Optimization status	ERROR	MAX_ITER	OK
SQP iterations	–	200	70
Optimization time (s)	–	3068	1186
Constraints evaluation time (s)	–	2176	847

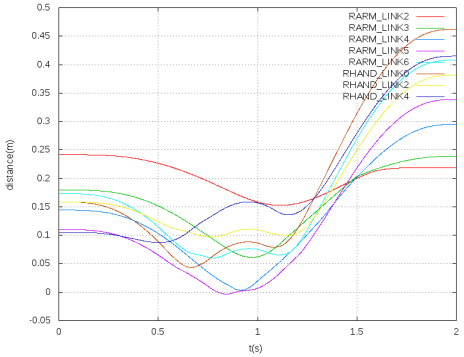


Fig. 5: Plots of left arm body distance constraints.

To showcase the usefulness of our approach, we try to solve the optimal control problem defined in Section VI starting from the different paths, and put all results in Table I. When starting with the initial path from figure 4a, the solver failed to achieve a single iteration. This can be explained by the fact that in the middle of this path, the robot left hand is clearly enclosed inside the obstacle and some distance constraints are violated; the solver fails to determine a clear direction which would lessen this violation due to the geometric local minimum. Since the constrained RRT avoids it and generates a collision-free path, the solver behaves correctly when starting with the path in 4b, but the maximum number of iterations is reached before reaching convergence. It is achieved when starting with the shortcut path in 4c. Note that about 70% of the optimization time is spent in evaluating the distance constraints and their gradients. Regarding their enforcement, figure 5 shows the evaluation of the left arm distances: due to the constraints time discretization, one constraint is violated during less than 100ms. This violation does not however exceed 4mm, and this is considered as acceptable as all distance constraints are computed with the bounding capsule geometry.

B. Fast Trajectory Generation on HRP-2

We also use our approach to generate fast optimal collision-free trajectories and execute them on the humanoid robot HRP-

Fig. 6: HRP-2 crosses its arms quickly while avoiding self-collision.

Fig. 7: HRP-2 bends down quickly to grab a ball in the lower shelf.

TABLE II: Scenarios Computation Times

Scenario	Arms-crossing	Shelves
Planning time (s)	4	13
Shortcut time (s)	4	6
SQP iterations	82	74
Optimization time (s)	636	1745
Constraints evaluation time (s)	248	1396

2. In the first scenario, HRP-2 crosses its arms while bending its knees. The motion must be executed while ensuring the arms do not collide whith each other, and ensuring that the robot does not fall. This is quite a difficult task as the trajectory duration is fixed to 1 second. In the scenario, we add a complex environment that contains shelves with different levels; HRP-2 bends it knees to grab a ball on the lower shelf. The trajectory lasts 2 seconds. We successfully apply our framework to generate feasible motions for both scenarios as seen in figures 6 and 7. Computation times are shown in table II. Videos are available on http://homepages.laas.fr/aelkhour/video/12humanoids/scenarios.mp4.

C. Discussion

In subsection VII-A, we demonstrate in a simple example the influence of the initial guess of the optimal control problem on the solver success and performance. In fact, due its probabilistic completeness, the usage of the constrained planner in a first stage guarantees that an initial collision-free and quasi-statically feasible trajectory can be found. All the optimization solver then has to do is to reshape this trajectory to minimize the objective function, i.e. the sum of jerks, and enforce additionnal constraints, i.e. joint limits and dynamic balance.

The trajectory duration is for the moment fixed. This means that if it is not properly set, the optimization solver might fail as some constraints such as velocity limits would never be enforced. If it was to be included as a free variable in the optimal control problem, we would have a guarantee of success for the second stage. The complete framework would then have both properties of probalistic completeness and soundness.

VIII. Conclusion and Future Work

In this paper we propose a novel approach to tackle optimal control problems in cluttered environments. Our approach combines, in a two-stage framework, a consrained path planning algorithm and an optimal control problem solver. We generate optimal feasible trajectories for the humanoid robot HRP-2 and successfully execute them.

Our approach can benefit from improvements to increase its usability, such as taking into account movable obstacles, and considering non-coplanar contacts. We are currently working to include these features.

Acknowledgment

The authors would like to thank Katja Mombaur and Martin Felis from University of Heidelberg for their active support in using MUSCOD-II, and Olivier Stasse and Nicolas Mansard from LAAS-CNRS for fruitful discussions. This work was supported by the European Commission under the FP7 project ECHORD (grant 231143).

References

[1] J. Kuffner, J.J. and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*.

[2] S. Dalibard, A. Nakhaei, F. Lamiraux, and J.-P. Laumond, "Whole-body task planning for a humanoid robot: a way to integrate collision avoidance," in *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*.

[3] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*.

[4] M. Stilman, "Task constrained motion planning in robot joint space," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*.

[5] L. Biegler and V. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers and Chemical Engineering*, selected Papers from the 17th European Symposium on Computer Aided Process Engineering held in Bucharest, Romania, May 2007.

[6] D. B. Leineweber, I. Bauer, A. Schfer, H. G. Bock, and J. P. Schlder, "An efficient multiple shooting based reduced sqp strategy for large-scale dynamic process optimization. parts 1 and 2," *Computers and Chemical Engineering*, 2003.

[7] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*.

[8] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running," *Mechatronics, IEEE/ASME Transactions on*, 2010.

[9] S. Miossec, K. Yokoi, and A. Kheddar, "A method for trajectory optimization of robots having contacts or motion constraints," in *Robotics, Automation and Mechatronics, 2006 IEEE Conference on*.

[10] S. Lengagne, P. Mathieu, A. Kheddar, and E. Yoshida, "Generation of dynamic motions under continuous constraints: Efficient computation using b-splines and taylor polynomials," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*.

[11] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*.

[12] A. Escande, S. Miossec, and A. Kheddar, "Continuous gradient proximity distance for humanoids free-collision optimized-postures," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*.

[13] O. Stasse, A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. Kheddar, "Real-time (self)-collision avoidance task on a hrp-2 humanoid robot," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*.

[14] O. Kanoun, "Real-time prioritized kinematic control under inequality constraints for redundant manipulators," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, 2011.

[15] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, 2011.

[16] A. Perez, R. J. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *Robotics and Automation, 2012. ICRA '12. IEEE International Conference on*.

[17] J.-P. Laumond, "Kineo cam: a success story of motion planning algorithms," *Robotics Automation Magazine, IEEE*, 2006.

[18] T. Moulard, "Roboptim," https://github.com/laas/roboptim.

[19] D. H. Eberly, "Wild magic," http://www.geometrictools.com/.

[20] I. Sucan, "planning environment," http://www.ros.org/wiki/planning_environment.

[21] M. Vukobratovic and B. Borovac, "Note on the article "zero-moment point - thirty five years of its life"," *I. J. Humanoid Robotics*, 2005.

[22] T. Flash and N. Hogans, "The coordination of arm movements: An experimentally confirmed mathematical model," *Journal of neuroscience*, 1985.