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-31400 Toulouse Cedex 4, France
Univ de Toulouse, LAAS, F-31400 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 encountered 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 $\mathbf{q}$ be a vector containing the generalized coordinates of a robot, and let $\mathcal{CS}$ be the set of all possible configurations $\mathbf{q}$. Path planning is mainly interested in the determination of a feasible path P connecting a start configuration $\mathbf{q_s}$ to a goal configuration $\mathbf{q_g}$ in $\mathcal{CS}$. P is said to be feasible iff for all $\mathbf{q} \in P$, $\mathbf{q}$ 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 high-dimension $\mathcal{CS}$ and cluttered environments. In its simplest form, this algorithm expands a tree $\mathcal{T}$ rooted in $\mathbf{q_s}$ by first sampling a random configuration $\mathbf{q_{rand}}$ in $\mathcal{CS}$, then extending a valid edge from the nearest configuration $\mathbf{q_{near}}$ in $\mathcal{T}$ towards $\mathbf{q_{rand}}$. This process is repeated until $\mathcal{T}$ and $\mathbf{q_g}$ can be connected.

It is interesting to note that all nodes and edges of the final tree $\mathcal{T}$ lie in $\mathcal{CS}$. This can lead to infeasible paths if the robot must respect additional kinematic constraints. [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 $\mathcal{CS}$. $\mathcal{M}$ is defined by kinematic constraints, each defined by a feature value in the operational space, e.g. end-effector position and orientation. The constrained RRT algorithm produces a path P such that for all $\mathbf{q} \in P$, $\mathbf{q}$ lies on the manifold $\mathcal{M}$.

An important feature of sampling-based algorithms is their probabilistic completeness, i.e. their capacity to avoid falling into local minima and to find a solution path if it exists. They present however three drawbacks. First, due to their sampling nature, the configuration $\mathbf{q}$ might move in a random fashion along the path P, which could lead to unnecessarily long and unnatural paths. Second, we still need to apply a time parametrization 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. Third, the resulting paths are continuous but not C^1. The time-parametrized motion thus needs to stop at each waypoint or to leave the planned path around waypoints. 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,
- $\mathbf{x}$ be the state vector,
- $\mathbf{u}$ be the control vector,
- t_f be the trajectory duration,
- ϕ be the scalar objective function to minimize,
- $\mathbf{f}$ be the differential equation of the model,
- $\mathbf{g}$ be the equality constraint vector function,
- $\mathbf{h}$ be the inequality constraint 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:

$$\begin{aligned} \dot{x}(t) &= f(t,x(t),u(t)) \\ g(t,x(t),u(t)) &= 0, \\ h(t,x(t),u(t)) &\geq 0 \end{aligned} \tag{2}$$

Starting from an initial value of $\mathbf{x}$ and $\mathbf{u}$, 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. Particularly, multiple shooting optimization, implemented in the MUSCOD-II software [6], is a powerful tool that allows solving large-scale optimal control problems thanks to a specially tailored SQP.

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 so far been only applied to simple trajectory optimization problems where obstacle and torque constraints were integrated in the cost function.

Optimal control techniques have been successfully applied on humanoid figures [8] and humanoid robots [9], [10]. The problems take into account complex robot dynamics and actuator limitations. However, current formulations either work under the assumption that the initial guess is collision-free or allow it to be slightly in collision by means of linear interpolation between initial and final configurations. There are then cases where the solver might get stuck in local minima and fail to generate a trajectory without any collisions with either the environment or bodies of the robot. Hence, 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 can express the constraint with the exact distance between the exact polyhedral geometries such as in [11]. While distance computation is precise and efficient thanks to bounding volume hierarchy structures, distance is zero in case of penetration, which forbids the computation of a correct constraint gradient. [12] propose a fast penetration computation algorithm, but computation times are restrictive. Moreover, the distance constraint between two non-strictly convex polyhedra is not C^1, which can cause the optimization solver to behave incorrectly. [13] proposes a nice solution to this problem and introduces Sphere-Torus Patches Bounding Volumes (STPBV), which are strictly convex bounding volumes; 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 [14]. In [15], 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 [16] 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 $\mathcal{CS}$, but the whole state space $\mathcal{S}$ where an element of $\mathcal{S}$ is $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T$. This doubles the dimension of the problem. Another problem arises from choosing the correct metric and local optimal policy. [17] propose a variant uses local linearization of a system to derive a coherent extension method. Results have so far been only obtained on simple systems, and this method has yet to be tested on complex ones.

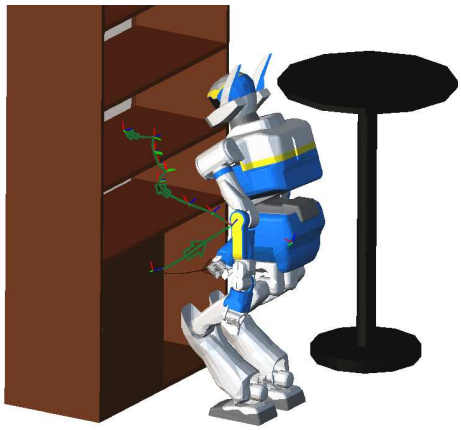


Fig. 1: Path found by the path planner in a shelves environment.

III. Contribution

To the authors' best knowledge there is no available algorithmic approach that addresses the problem of optimal motion planning for complex 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. Second, we use the bounding capsules to implement distance constraints for the solver and achieve (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 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™[18]. This planner allows generating 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 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 $\mathbf{q_g}$, but instead define a goal task $\mathbf{x_g}$. 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 as the intersection between $\mathcal{M}$ and the manifold defined by the following stack of equality constraints:

1) Gripper has the same 3D position as the object to grab.
2) Gripper thumb is oriented vertically.

Given a start configuration $\mathbf{q_s}$ a planning manifold $\mathcal{M}$ and a goal sub-manifold $\mathcal{M}_g$, we first create a set of goal configurations $\mathbf{q_g}$ by sampling a fixed number of configurations in $\mathcal{M}_g$, then we solve the path problem from $\mathbf{q_s}$ to $\mathbf{q_g}$. The constrained planner diffuses trees from $\mathbf{q_s}$ and each configuration of $\mathbf{q_g}$, and stops once at least one of the goal configurations is in the same connected component as $\mathbf{q_s}$. A shortcut optimizer can then prune unnecessary waypoints and smooth the solution path. Figure 1 shows an 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, Boolean collision queries are used to validate or invalidate configurations and hence the whole path. This is 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 the means to get out of collision.

We choose to use bounding capsules, like in [15]. Capsules are sphere swept segments; the inter-capsule distance can be simply found by computing the distance between the capsule axes, then subtracting their radii. Similarly, we can compute the distance between a capsule and a polyhedron by 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 [15], bounding capsules parameters, i.e. the 2 endpoints $\mathbf{e_1}, \mathbf{e_2}$ 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:

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

subject to:

$$r - d(\mathbf{p}, \mathbf{e_1}\mathbf{e_2}) \quad \geq \quad 0, \text{ for all } \mathbf{p} \in \mathcal{P} \tag{4}$$

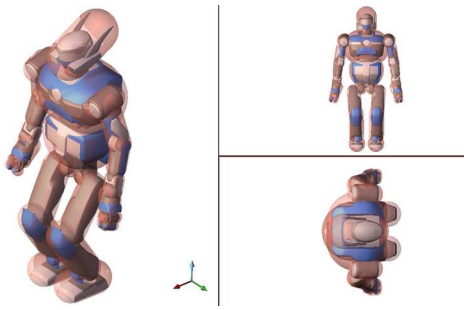


Fig. 2: Best fitting capsules for HRP-2

where $d(\mathbf{p}, \mathbf{e_1}\mathbf{e_2})$ is the distance of $\mathbf{p}$ to line segment $\mathbf{e_1}\mathbf{e_2}$. 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 [19] and the IPOPT solver [5] in less than 5s. Figure 2 shows the best fitting bounding capsules 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 [20] 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 [18] for an efficient computation. For the environment in figure 1, the distance for one capsule-environment pair takes about 500 μs to be computed, since the environment is assumed to be perfectly modeled by polyhedron meshes.

C. Selection of pairs of bodies for distance 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 820 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; the pairs corresponding to those bodies can be safely pruned. We use the tool described in [21], 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 41 capsule-environment pairs that do not need to be checked due to the particular kinematic structure of the HRP-2. For instance, if both waist and chest are not in collision with the environment, we can be sure that it will be the same for the intermediate body linking them. We thus keep 23 capsule-environment pairs.

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:

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

and we define the state and control variables to be:

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

2) Equality and inequality constraints:

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

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

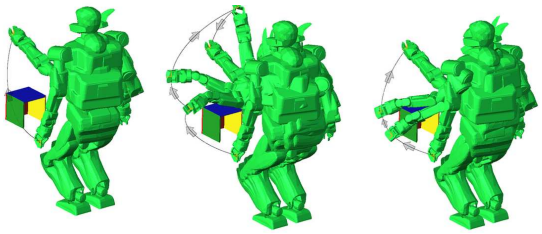
b) Dynamic balance: The robot is submitted in our case to multiple coplanar contact reaction forces from the ground. We can then express the dynamic balance constraint using the Zero-Moment Point (ZMP) [22], which has to remain inside the robot support polygon defined by its feet.

These constraints can be written as for any $t \in [0, t_f]$:

$$\begin{array}{rcl} \mathbf{p_{lf}}(\mathbf{q}(t)) & = & \mathbf{p_{lf}}(\mathbf{q}(0)) \\ \mathbf{p_{rf}}(\mathbf{q}(t)) & = & \mathbf{p_{rf}}(\mathbf{q}(0)) \\ \mathbf{zmp}(\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t)) & \in & \mathcal{P}_{sup}, \end{array} \tag{8}$$

where $\mathbf{p_{lf}}$, $\mathbf{p_{rf}}$ are respectively the 6D positions of the left and right feet, **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 $\mathbf{q}$ 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 each single body and define as an inequality constraint, the minimum distance of the body to other bodies and to the obstacles being positive.



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

Fig. 3: 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 $\mathbf{q}(t)$ and $\mathbf{u}(t)$ 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 parametrization before feeding it to the optimization solver. We want to minimize the sum of square jerks; [23] 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 path waypoints, assuming they start and end at zero velocity and acceleration. This ensures that the configuration $\mathbf{q}(t)$ follows exactly the solution path and that collision avoidance constraints are not violated. The optimization solver will then reshape this initial guess while enforcing all constraints, leading to a smooth motion without intermediate stops.

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.

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 2.53 GHz Intel® Core™2 Duo processor.

A. Test Case

Figure 3a 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 3b shows a possible solution path found with constrained RRT. This path can be shortened with a shortcut optimizer, as in figure 3c.

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

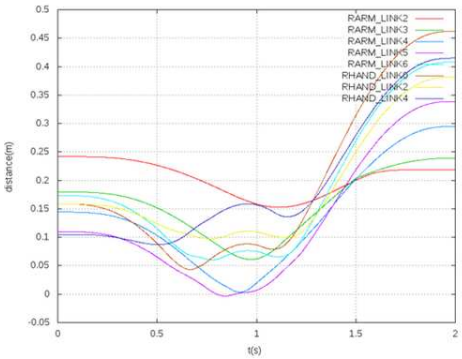


Fig. 4: Plots of left arm body distance constraints.

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 3a, 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 enclosed inside the obstacle and some distance constraints are violated; the solver fails to determine a clear direction which would remove 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 3b, but the maximum number of iterations is reached before reaching convergence. It is achieved when starting with the shortcut path in 3c. Note that about 70% of the optimization time is spent in evaluating the distance constraints and their gradients; this significant overhead can be explained by the fact that MUSCOD-II relies on internal numerical differentiation to compute Jacobians. Relying on analytical gradient expressions would thus accelerate the optimization process. Regarding the distance constraints enforcement, figure 4 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 bounding capsule geometries which already define conservative volumes around the exact geometries.

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-2. In the first scenario, HRP-2 executes a kind of martial art figure where it crosses its arms rapidly while bending its knees, changes the arms configuration, then moves back to a rest posture. The motion must be executed while ensuring the arms do not collide with each other, and the robot does not fall. This is quite a difficult task as the 3 trajectories durations are fixed to 1, 2, and 2 seconds respectively. Particularly, the second motion where one arm goes from being behind the other arm to being ahead of it proved to be impossible to generate without a prior planning phase as proposed in out approach.

TABLE II: Computation Times for Martial Arts Scenario

Phase	1	2	3
Planning time (s)	4	13	2
Shortcut time (s)	4	6	1
SQP iterations	32	73	25
Optimization time (s)	346	1130	278
Constraints evaluation time (s)	124	356	83

TABLE III: Computation Times for the Shelves Scenario

Phase	1	2
Planning time (s)	13	38
Shortcut time (s)	6	23
SQP iterations	74	80
Optimization time (s)	1745	5020
Constraints evaluation time (s)	1396	2640

In the second scenario, we add a complex environment that contains shelves with different levels; HRP-2 first bends its knees to grab a ball located deep on the lower shelf, then moves it to an upper shelf to release it between two other objects. The trajectories last respectively 2 and 5 seconds. Here, both collision and self-collision constraints need to be enforced in order to obtain a valid trajectory. Again, the ball transfer motion cannot be generated using an optimal control solver and a simple initial guess; a prior planning phase is needed to find a collision-free transfer path.

We successfully apply our framework to generate feasible motions for both scenarios as seen in figures 5 and 6. Computation times are shown in Tables II and III. Videos of both scenarios are available in the attached file.

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 to 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. The optimization solver then can reshape this trajectory in order to minimize the objective function while enforce constraints such as joint limits and dynamic balance. Note that with this method, locally optimal trajectories are found; in order to find global optima, it might be interesting to plan in $\mathcal{CS}$ using RRT* instead of RRT in the first stage.

The trajectory duration is for the moment fixed. This means that if it is not properly set, the optimization solver might fail

Fig. 5: HRP-2 does a quick martial arts motion while avoiding self-collision.

Fig. 6: HRP-2 bends down quickly to grab a ball in the lower shelf and transfers it to the upper shelf.

as some constraints such as velocity limits would never be enforced. If it is included as a free variable in the optimal control problem, we will have a guarantee of success for the second stage. The complete framework would then always succeed in generating optimal trajectories.

VIII. Conclusion

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 constrained 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 framework can benefit from improvements to increase its usability. Future work will consider non-coplanar contacts, as well as release the robot from its fixed support constraints in order to accomplish optimal locomotion planning.

Acknowledgment

The authors would like to thank Katja Mombaur and Martin Felis from University of Heidelberg for their strong support with MUSCOD-II, as well as Olivier Stasse and Nicolas Mansard from LAAS-CNRS for their valuable help during the experiments. 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. 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. IEEE International Conference*.

[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. IEEE International Conference*.

[5] L. Biegler and V. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers and Chemical Engineering*, 2009.

[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." *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 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] M. Toussaint, M. Gienger, and C. Goerick, "Optimization of sequential attractor-based movement for compact behaviour generation," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, 2007.

[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] Y. J. Kim, M. A. Otaduy, M. C. Lin, and D. Manocha, "Fast penetration depth computation for physically-based animation," in *2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2002.

[13] 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*.

[14] 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. International Conference*.

[15] O. Kanoun, "Real-time prioritized kinematic control under inequality constraints for redundant manipulators," in *Robotics: Science and Systems*, 2011.

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

[17] 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. IEEE International Conference on*.

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

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

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

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

[22] M. Vukobratović and B. Borovac, "Zero-moment pointthirty five years of its life," *International Journal of Humanoid Robotics*, 2004.

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