Motion Planning with Multi-Contact and Visual Servoing on Humanoid Robots

Kevin Giraud–Esclasse, Pierre Fernbach, Gabriele Buondonno, Carlos Mastalli, Olivier Stasse

Motion Planning with Multi-Contact and Visual Servoing on Humanoid Robots

Kevin Giraud-Esclasse[1], Pierre Fernbach[1], Gabriele Buondonno[1],
Carlos Mastalli[1] and Olivier Stasse[1]

***Abstract*— This paper describes the implementation of a canonical motion generation pipeline guided by vision for a TALOS humanoid robot. The proposed system is using a multicontact planner, a Differential Dynamic Programming (DDP) algorithm, and a stabilizer. The multicontact planner provides a set of contacts and dynamically consistent trajectories for the Center-Of-Mass (CoM) and the Center-Of-Pressure (CoP). It provides a structure to initialize a DDP algorithm which, in turn, provides a dynamically consistent trajectory for all the joints as it integrates all the dynamics of the robot, together with rigid contact models and the visual task. Tested on Gazebo the resulting trajectory had to be stabilized with a state-of-the-art algorithm to be successful. In addition to testing motion generated from high specifications to the stabilized motion in simulation, we express visual features at Whole Body Generator level which is a DDP formulated solver. It handles non-linearities as the ones introduced by the projections of visual features expressed and minimized in the image plan of the camera.**

I. Introduction

The expectations for a humanoid robot stand in its capability to navigate in structured and unstructured environments, potentially using other parts of its body than only feet. It should also fullfil tasks of manipulation with its upper body for instance. These expectations imply the use of multiple loops of control on different sensors. Specifically, the exteroceptive sensors like cameras are needed to react to an environment modifications or modelling errors. From the results of the DRC (Darpa Robotic Challenge) it appears that the methods available in the robotic field do not live up to these expectations. Considering assumptions or simplifications in either the environment or robot models, a part of the goal has been reached. Assuming flat floor, absence of obstacle and only biped locomotion (without multicontact), a significant body of work exists to use a representation of the environment, plan foot steps and execute them on humanoid robots of various sizes[1], [2], [3], [4]. An approach could be to use Model Predictive Control (MPC) on a Linear Inverted Pendulum where footsteps, Center-of-Mass (CoM) and Center-of-Pressure (CoP) trajectories are solved together with only the desired CoM velocity as input. Often, to reach online execution of the algorithm, the model of the robot dynamics is simplified as Angular Momentum is not taken into account in the centroidal dynamics (non-linearity). For instance, the model can be kinematically expressed in an Optimal Controller problem (less time consuming) and then rectified with a dynamic filter as in [1] that calculates dynamics errors and corrects the previous solution.

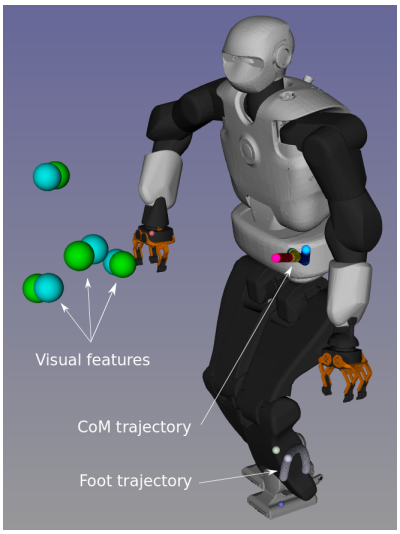


Fig. 1: Resulting position of the robot after simulation with three contacts and visual task. TALOS' right hand (frame represented by a small red point) is equal with the target one. Center of mass trajectories are displayed: each color represents a phase corresponding to a contact change. Big green spheres represent referenced features for the visual task, blue ones for the output last position.

To get closer from the initial expectations, the CoM velocity can be driven by desired visual features as done in [5] where they tackle the sway motion generated by walk oscillations. In addition to all the previous assumptions, the CoM velocity cannot always be achieved due to the constraints of the foot placements and the balance which are of high priority. Another work with exteroceptive sensors copes with dynamical model simplification: in [6] HRP-2 humanoid robot was able to carry a fire hose while walking. The system was using an external localization by motion capture feedback and a real-time pattern generator able to decide by itself foot step locations and generate a balanced Center-of-Mass trajectory.

The final goal is to remove the assumptions and simplifications commonly made, such as assuming a flat floor, convex obstacles, a gaited motion, ignoring the

[1] LAAS, CNRS, Toulouse, France.

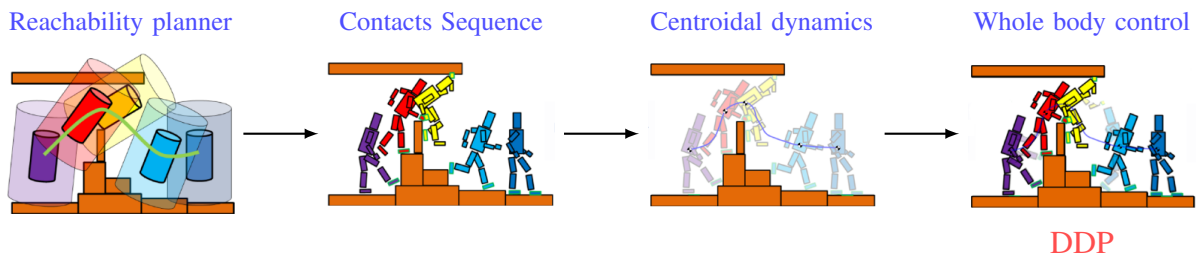


Fig. 2: Overall approach: The reachability planner takes a starting configuration ($\mathcal{C}_S$) and a goal configuration $\mathcal{C}_G$. The motion planner described in section II provides a contact sequence, and a centroidal dynamics trajectory. The DDP described in section III generates a whole body trajectory which is consistent with the contact dynamics and the complete model of the robot.

self-collisions, approximating the dynamic model, ... Being able to leverage such functionalities with a computation time suitable for online execution is quite challenging.

A common approach is to decouple the entire problem in smaller sub-problems solved sequentially as shown in figure 2, that could be handled more efficiently [7]. From a desired final position, the Planner (first and second block) provides steps and contact locations to a Centroidal Dynamics generator (third block) giving CoM dynamical trajectories to a Whole Body Controller (fourth block) that generates joint trajectories. Then, these joint trajectories are sent to a Stabilizer looped on robot motions.

Exteroceptive sensors providing SLAM or visual references could be looped on these blocks at different levels. In this more general setup, combination of these blocks is very tough as highlighted during the DRC [8]. Recent work has proposed a solution to introduce multiple contacts together with vision in [9]. They use this decoupled approach to generate a motion in simulation. The main drawback lies in the use of local Quadratic Programming formulation and the assertion of an axis of forces on CoM dynamics that makes total Centroidal Dynamics linear (even on angular momentum).

The aim of our work, our first contribution, is to build and evaluate this kind of generic motion generation pipeline for a TALOS humanoid robot. We test all the pipeline from the motion planner to the stabilizer block on simulation.

The second contribution of this paper lies in the connection of visual features in this workflow. We decided to use visual tasks in image plan as input of the Whole Body Controller. This choice was motivated by the fact that visual features are directly embedded in the controller that can locally manage modifications of the motion accordingly to those features. Moreover the chosen controller can cope with the non-linearities produced by the projection in the image plan on all the time horizon (contrary to [10] that first order linearizes projections around the first point of the trajectory). Indeed, to tackle multicontact objective and non-linearities of the dynamics such as angular momentum equations, we decided to use Differential Dynamic Programming (DDP) solver that is an Optimal Control algorithm managing non-linearities on state function. It was described in [11]. More recently, it has been used successfully for the DRC in [12], and also proposed for humanoid robots in [13]. In [14], the DDP is used to generate whole body motions (corresponding to the fourth subpart previously mentioned).

In this work we report our first tests in integrating a fast multicontact planner used to set a DDP problem which in turns provides reference trajectories to a local whole body instantaneous controller (stabilizer block). It was tested in dynamical simulation (Gazebo/ODE) on the TALOS humanoid robot. In section II we briefly explain how the multicontact planner works. In Section III we give some reminders about DDP algorithm and visual tasks elements to show how it can be integrated. Experiments and results are shown respectively in Section IV and V on a TALOS robot with multicontact and visual tasks.

II. Multicontact planner

As shown in figure 3, the multicontact planner takes as input a model of the environment and the robot and a high level description of a locomotion task: an initial and final pose for the origin of the robot. Optionally some additional constrains may be specified, such as a velocity bounds or a set of initial or final contact positions. The output of this planner is a dynamically consistent and kinematically valid (i.e. which respects collisions with the environment, self-collision and joints limits) joint trajectory.

The connection between the blocks of the multicontact planner is automatic and does not require any manual intervention. The only required expertise is to set the values of the different parameters (cost function weights, constraints, gains ...) used by the various blocks, this step should only be done once per robot and does not depend on the environment nor the task given to the motion planner.

The following paragraphs describe briefly each method of the architecture and refer the interested reader to the papers introducing these methods.

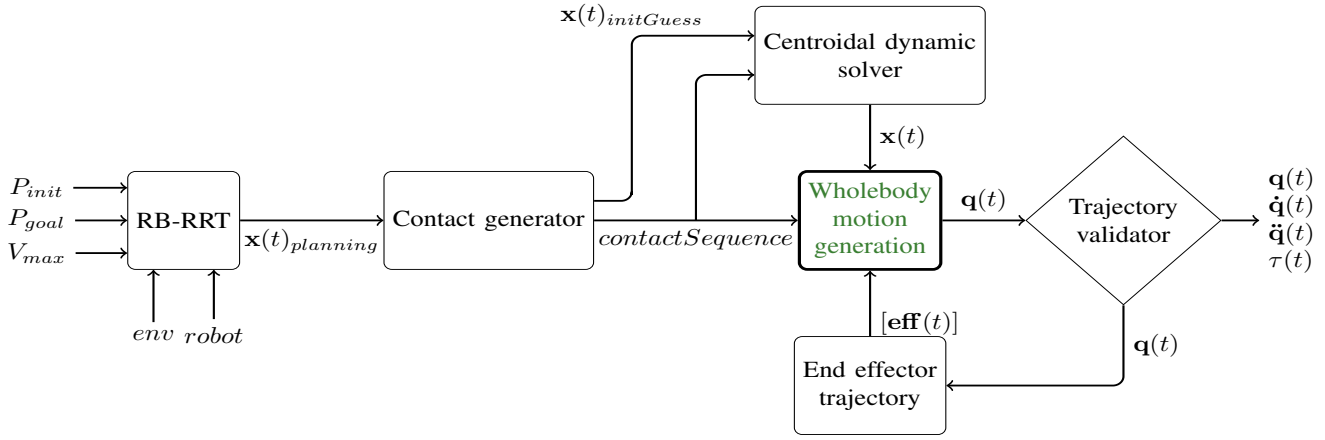


Fig. 3: Multicontact motion planner architecture. All the blocks are implemented in C++ with a Python API, except the wholebody motion generator implemented in Python. The communication between the blocks is made with a dedicated data structure, in Python. The green block is solved by the DDP detailed in section III.

A. Guide path

The first block of the figure 3 produces a rough guide trajectory for the root of the robot. The method RB-RRT was first proposed in [15] and then extended to a kinodynamic version in [16]. This method plan a trajectory for the center of a simplified model of the robot, using a heuristic based on the reachability space of each limb. The goal of this method is to plan a trajectory such that the robot can go from the starting configuration to the goal configuration without collision while maintaining contact with the environment using limbs.

B. Contact sequence

The contact generation method presented in [15] produces a sequence of whole body configurations in contact along the guide previously planned, such that there is only one contact change between two adjacent configurations. The feasibility of the motion between two configurations of the sequence is guaranteed by solving a convex reformulation of the multicontact centroidal dynamic trajectory generation problem [17].

C. Centroidal trajectory

The centroidal trajectory is generated with the method proposed in [18]. This method takes as input the sequence of contacts and produces a centroidal trajectory satisfying the centroidal dynamic constraints for the given contact points and minimizing a tailored cost function. This method can generate centroidal trajectory for multicontact scenario in real-time thanks to a convex relaxation of the problem.

D. Wholebody motion generation

From a contact sequence and a reference centroidal trajectory and end-effectors trajectories, this block should produce a wholebody trajectory. This trajectory must satisfy the dynamic constraints applied to the robot and leads to a feasible motion (i.e. the robot does not fall during the execution of the motion).

In our previous work, this problem was solved with a Task Space Inverse Dynamics method implemented in TSID [1]. In this paper, we propose to solve this problem with a DDP algorithm as detailed in section III.

E. Trajectory validation

As the solver used for the wholebody motion generation does not include hard constraints on the (self-)collision neither on the joints limits of the robot, the trajectory need to be validated a posteriori. For that purpose we use the tools provided by the Humanoid Path Planner framework [19], the collisions are verified with the collision library FCL [20].

If the produced trajectory is not kinematically valid because of one of the moving limbs, the reference trajectory of the end effector is automatically and iteratively modified until a valid motion is found [21]. In the rare case where no kinematically valid motion can be found at this step, the complete framework is started again. Thanks to the probabilistic nature of the guide and contact planning, a different solution will be found.

III. DDP AND VISUAL TASKS

In this section we describe our visual tasks based approach under multicontact events based on DDP. For that, we first introduce our DDP algorithm tailored to mutiphase rigid dynamics [14]. And later, we explain the visual task formulation within our multicontact DDP. This work is based on the DDP solver implemented in Crocoddyl [22], which computes efficiently the rigid body dynamics and its derivatives using Pinocchio [23].

A. Differential dynamic programming

DDP belongs to the family of Optimal Control (OC) and Trajectory Optimization (TO) [11]. It locally approximates

[1]https://github.com/stack-of-tasks/tsid

the optimal flow (feedback gains), and as a consequence, the OC problem is split into simpler and smaller subproblems (sparse structure). The DDP promises to handle whole-body MPC on a humanoid thanks to its sparse structure [13]. However, the main drawback lies on the fact that it poorly handles constraints. [24] manages to implement constrained DDP, increasing the computation time due to quadratic programs used in inner loops instead of matrix inversions.

Let us consider a generic multicontact OC problem as follows:

$$\begin{aligned} \mathbf{X}^*, \mathbf{U}^* &= \arg\min_{\mathbf{X},\mathbf{U}} l_T(\mathbf{x}_N) + \sum_{k=0}^{T-1} l_k(\mathbf{x}_k, \mathbf{u}_k) \\ &\text{s. t.} \quad \mathbf{x}_0 = \tilde{\mathbf{x}}_0, \\ &\quad \mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k), \end{aligned} \tag{1}$$

where T is the given horizon, the state $\mathbf{x} = (\mathbf{q}, \mathbf{v})$ lies in a Lie manifold with $\mathbf{q} \in SE(3) \times \mathbb{R}^{n_j}$ and $\mathbf{v} \in T_{\mathbf{x}}\mathcal{Q}$, $\tilde{\mathbf{x}}_0$ is the initial condition, the system is underactuated $\mathbf{u} = (\mathbf{0}, \boldsymbol{\tau})$ with $\boldsymbol{\tau}$ the torque commands, the discrete dynamics $\mathbf{f}_k(\cdot)$ describes different contact phases, and $l_k(\mathbf{x}_k, \mathbf{u}_k)$ describes the different tasks (or running costs) and $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_T\}$ and $\mathbf{U} = \{\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{T-1}\}$ are the sequences of states and controls along the defined horizon. Note that both – cost and dynamics – are often time varying functions.

DDP breaks the dynamic problem into simpler subproblems thanks to the "Bellman's principle of optimality". Indeed, moving backward in time, the approximated value function $V(\cdot)$ can be found by minimizing the local policy for a given time k (problem formulated and all data relative to that time form a node), i.e.

$$V_k(\delta\mathbf{x}_k) = \min_{\delta\mathbf{u}_k} l_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k) + V_{k+1}(\mathbf{f}_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k)), \tag{2}$$

and this is locally approximated by a quadratic function (as a Gauss-Newton approximation) as follows:

$$\begin{aligned} &\delta\mathbf{u}_k^*(\delta\mathbf{x}_k) = \\ &\arg\min_{\delta\mathbf{u}_k} \frac{1}{2} \begin{bmatrix} 1 \\ \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \end{bmatrix}^T \begin{bmatrix} 0 & \mathbf{q}_{\mathbf{x}_k}^T & \mathbf{q}_{\mathbf{u}_k}^T \\ \mathbf{q}_{\mathbf{x}_k} & \mathbf{q}_{\mathbf{x}\mathbf{x}_k} & \mathbf{q}_{\mathbf{x}\mathbf{u}_k} \\ \mathbf{q}_{\mathbf{u}_k} & \mathbf{q}_{\mathbf{x}\mathbf{u}_k}^T & \mathbf{q}_{\mathbf{u}\mathbf{u}_k} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \end{bmatrix}, \end{aligned} \tag{3}$$

where $\delta\mathbf{x} = \bar{\mathbf{x}} \ominus \mathbf{x}$ is the deviation with respect to the local linearization $\bar{\mathbf{x}}$ and belongs to the tangential space ($\in T_{\mathbf{x}}\mathcal{Q}$), and the Jacobian and Hessian of the Hamiltonian are defined as:

$$\begin{aligned} \mathbf{q}_{\mathbf{x}_k} &= \mathbf{l}_{\mathbf{x}_k} + \mathbf{f}_{\mathbf{x}_k}^T V_{\mathbf{x}_{k+1}}, \\ \mathbf{q}_{\mathbf{u}_k} &= \mathbf{l}_{\mathbf{u}_k} + \mathbf{f}_{\mathbf{u}_k}^T V_{\mathbf{x}_{k+1}}, \\ \mathbf{q}_{\mathbf{x}\mathbf{x}_k} &= \mathbf{l}_{\mathbf{x}\mathbf{x}_k} + \mathbf{f}_{\mathbf{x}_k}^T V_{\mathbf{x}\mathbf{x}_{k+1}} \mathbf{f}_{\mathbf{x}_k}, \\ \mathbf{q}_{\mathbf{x}\mathbf{u}_k} &= \mathbf{l}_{\mathbf{x}\mathbf{u}_k} + \mathbf{f}_{\mathbf{x}_k}^T V_{\mathbf{x}\mathbf{x}_{k+1}} \mathbf{f}_{\mathbf{u}_k}, \\ \mathbf{q}_{\mathbf{u}\mathbf{u}_k} &= \mathbf{l}_{\mathbf{u}\mathbf{u}_k} + \mathbf{f}_{\mathbf{u}_k}^T V_{\mathbf{x}\mathbf{x}_{k+1}} \mathbf{f}_{\mathbf{u}_k}. \end{aligned} \tag{4}$$

We obtain the local policy by solving the Quadratic Programming (QP) (3) as:

$$\delta\mathbf{u}_k^*(\delta\mathbf{x}_k) = \mathbf{k}_k + \mathbf{K}_k \delta\mathbf{x}_k \tag{5}$$

where $\mathbf{k}_k = -\mathbf{q}_{\mathbf{u}\mathbf{u}_k}^{-1} \mathbf{q}_{\mathbf{u}_k}$ and $\mathbf{K}_k = -\mathbf{q}_{\mathbf{u}\mathbf{u}_k}^{-1} \mathbf{q}_{\mathbf{u}\mathbf{x}_k} \delta\mathbf{x}$ are the feedforward and feedback terms, respectively. And for the next node, we update the quadratic approximation of the value function by injecting $\delta\mathbf{u}_k^*$ expression into (3):

$$\begin{aligned} \Delta V(i) &= -\frac{1}{2} \mathbf{q}_{\mathbf{u}_k} \mathbf{q}_{\mathbf{u}\mathbf{u}_k}^{-1} \mathbf{q}_{\mathbf{u}_k} \\ V_{\mathbf{x}_k} &= \mathbf{q}_{\mathbf{x}_k} - \mathbf{q}_{\mathbf{u}_k} \mathbf{q}_{\mathbf{u}\mathbf{u}_k}^{-1} \mathbf{q}_{\mathbf{u}\mathbf{x}_k} \\ V_{\mathbf{x}\mathbf{x}_k} &= \mathbf{q}_{\mathbf{x}\mathbf{x}_k} - \mathbf{q}_{\mathbf{u}\mathbf{x}_k} \mathbf{q}_{\mathbf{u}\mathbf{u}_k}^{-1} \mathbf{q}_{\mathbf{u}\mathbf{x}_k} \end{aligned} \tag{6}$$

This backward pass allows us to compute the search direction during the numerical optimization. Then DDP runs a nonlinear rollout (a.k.a. forward pass) of the dynamics to try the computed direction along a step length α, i.e.

$$\begin{aligned} \hat{\mathbf{x}}_0 &= \tilde{\mathbf{x}}_0 \\ \hat{\mathbf{u}}_k &= \mathbf{u}_k + \alpha\mathbf{k}_k + \mathbf{K}_k(\hat{\mathbf{x}}_k \ominus \mathbf{x}_k) \\ \hat{\mathbf{x}}_{k+1} &= \mathbf{f}_k(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k) \end{aligned} \tag{7}$$

in which we perform a typical *backtracking* line search by trying first the full step ($\alpha = 1$).

The DDP solver iterates on these two phases – backward and forward passes – until convergence to the result (gradient approximately equals zero).

B. Handling tasks

A task is usually formulated as a regulator:

$$\mathbf{h}_{task_k}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}_{task}^d - \mathbf{s}_{task}(\mathbf{x}_k, \mathbf{u}_k), \tag{8}$$

where $\mathbf{s}_{task}^d$ and $\mathbf{s}_{task}(\mathbf{x}_k, \mathbf{u}_k)$ are the desired and the current feature vectors, respectively. As one wants to regulate each feature to the origin (i.e. $\lim_{t\to+\infty} \mathbf{h}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$), the task at each node is implemented as a penalty:

$$l_k(\mathbf{x}_k, \mathbf{u}_k) = \sum_{j \in tasks} \mathbf{w}_{j_k} \|\mathbf{h}_{j_k}(\mathbf{x}_k, \mathbf{u}_k)\|^2, \tag{9}$$

with $\mathbf{w}_{j_k}$ the weight assigned at time k to task j. Note that the DDP requires the derivative of the regulator functions, which are needed to compute the Jacobians and Hessians of the cost functions.

In our case we have the following $tasks \subseteq \{CoM, RH_{SE(3)}, RF_{SE(3)}, LF_{SE(3)}, EE_{SE(3)}^{eeName}, VT\}$: 1) the CoM tracking computed by the centroidal TO [18] (CoM), 2) the SE(3) tracking of the right-hand pose ($RH_{SE(3)}$), 3) the SE(3) tracking of the right- and left-feet pose ($RF_{SE(3)}, LF_{SE(3)}$), 4) the impact end-effector velocity ($EE_{SE(3)}^{eeName}$), and 5) the visual task expressed in the image plane (VT).

C. Handling dynamical constraints

Although the DDP does not handle constraints, it is possible to analytically derive the forward dynamics under rigid contact constraints (or holonomic constraints) and to penalize deviation from the contact forces [14] provided by the centroidal TO [18]. The numerical optimization problem

of this dynamics can be formulated by using the Gauss principle of least squares:

$$\begin{aligned} \dot{\mathbf{v}} &= \arg\min_{\mathbf{a}} \frac{1}{2}\|\dot{\mathbf{v}} - \dot{\mathbf{v}}_{free}\|_{\mathbf{M}} \\ &\text{s.t.} \quad \mathbf{J}_c\dot{\mathbf{v}} + \dot{\mathbf{J}}_c\mathbf{v} = \mathbf{0}, \end{aligned} \tag{10}$$

where $\mathbf{M}\dot{\mathbf{v}}_{free} = \boldsymbol{\tau}_b$ is the unconstrained robot dynamics, $\mathbf{M} \in \mathbb{R}^{n_j \times n_j}$ is the joint-space inertia matrix, $\boldsymbol{\tau}_b = \mathbf{S}\boldsymbol{\tau} - \mathbf{b} \in \mathbb{R}^{n_j}$ is the force-bias vector that accounts for the control $\boldsymbol{\tau}$, the Coriolis and gravitational effects $\mathbf{b}$, $\mathbf{S}$ is the selection matrix of the actuated joint coordinates and $\mathbf{J}_c = (\mathbf{J}_{c_1} \cdots \mathbf{J}_{c_f})$ is a stack of f contact Jacobians. The analytical solution of this QP as the form:

$$\begin{bmatrix} \mathbf{M} & \mathbf{J}_c^T \\ \mathbf{J}_c & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\nu}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S\tau - b} \\ \dot{\mathbf{J}}_c\mathbf{v} \end{bmatrix}, \tag{11}$$

where $\boldsymbol{\lambda}$ are the stack of contact forces, a.k.a. as the dual variables of (10). To take into account the dual variable in the resolution of the problem, dynamic equation is augmented as follows:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \\ \boldsymbol{\lambda}_k &= \mathbf{g}(\mathbf{x}_k, \mathbf{u}_k), \end{aligned} \tag{12}$$

where $\mathbf{g}(\cdot)$ is the dual solution of (11).

Since the main principles underlying DDP are exposed in this paragraph, visual tasks equations are briefly presented in the next paragraph in order to derive its integration and implementation.

D. Visual tasks

As the DDP algorithm needs residuals (or regulators) and derivatives of the tasks, this paragraph describes the formulation of the visual task and its derivatives.

Given the type of sensor / camera, the formulation of a visual task can differ. If the sensor provides depth information, the approach is called Point-Based Visual Servoing (PBVS). The formulation of that kind of task lies in $\mathrm{SE}(3)$ space. If the camera does not provide depth information (or if that data is not trustful due to errors, bias, noise), one will use the Image Based Visual Servoing (IBVS). This approach is detailed here.

Let us first consider the desired features $\mathbf{s}^d$ and the actual features $\mathbf{s}$. These last ones could refer to perceived information from camera or calculated by a simulator. The features can be points of interests, moments or more complex visual features. For sake of simplicity this study consider the simpler case of points.

The error of the task is then:

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^d \tag{13}$$

In our case, $\mathbf{s}^d$ is considered as fixed, not depending on the time. The error $\mathbf{e}$ is also considered as the residual of the cost l defined by:

$$l = \frac{1}{2}\|\mathbf{e}\|^2 \tag{14}$$

The model commonly used is a first order motion model:

$$\dot{\mathbf{e}} = \mathbf{L}_e\mathbf{v}_c \tag{15}$$

where $\mathbf{v}_c$ is the velocity of the camera in the camera frame, and $\mathbf{L}_e$ is the interaction matrix. This matrix can be considered as the features Jacobian. By differentiating the position of one feature in the 3D space, [25] has shown $\mathbf{L}_e$ can be written as follow:

$$\mathbf{L}_e = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{16}$$

Now, let us call $\mathbf{J}_c$ the Jacobian of the camera in the camera frame, and $\dot{\mathbf{q}}$ the joint-space velocity. Combining the well known expression $\mathbf{v}_c = \mathbf{J}_c\dot{\mathbf{q}}$ with (15), we find:

$$\dot{\mathbf{e}} = \mathbf{L}_e\mathbf{J}_c\dot{\mathbf{q}} \tag{17}$$

Contrary to the common visual servoing command law that enforces the exponential decrease by writing this relation: $\dot{\mathbf{e}} = -\lambda\mathbf{e}$, DDP needs the derivative of the task with respect to the state $\mathbf{x}$ and the control $\mathbf{u}$ as expressed in (4). As mentioned earlier, the state is composed by the robot configuration $\mathbf{q}$ and its joint-space velocity $\dot{\mathbf{q}}$, and its Jacobians are:

$$\frac{\partial \mathbf{e}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbf{0}_{2\times n_j} & \mathbf{L}_e\mathbf{J}_c \end{bmatrix} \tag{18}$$

$$\frac{\partial \mathbf{e}}{\partial \mathbf{u}} = \mathbf{0}_{2\times n_j - 6} \tag{19}$$

The Hessian of the visual task (i.e. $l_{\mathbf{xx}}$, $l_{\mathbf{xu}}$ and $l_{\mathbf{uu}}$) are equal to zero. Expressing visual task in the DDP formalism constitues the main theoretical contribution of this work.

IV. Simulations and Experiments

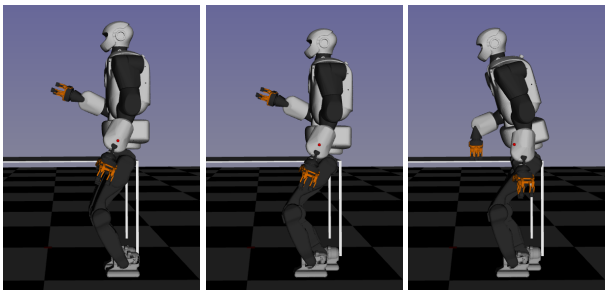
In this section we describe the situation of the robot and the tasks it has to manage, the software architecture used to generate appropriate motion and the results obtained in simulation.

A. Simulation setup

In our setup, TALOS begins in an initial double support standing configuration. It should make a step which is equivalent to walking motion and reach a contact surface (like a table) to create a third contact in order to bend sufficiently while maintaining balance to be able to see a target in its field of view. Then, keeping the three contacts, it should use visual tasks of the target to make it correspond to predefined desired features positions in the image plane. The main goal here is to be able to see an object and make it fit with reference while the posture needs a third (or more) contact. Concerning the visual features, the 3D reference used in simulation are supposed to be given by an external SLAM component.

Figure 4 shows the output of the contact planner: a sequence of three configurations in contact, with one contact change between each configuration.

In Crocoddyl, for each time step the dynamics and the cost of the problem are redefined so that tasks can be independently managed following a predetermined time line given from the previous stages, namely the contact planner and the centroidal trajectory generation method. In that way, our time line is divided as follows:



(a) (b) (c)

Fig. 4: Sequence of configurations in contact produced by the contact planner. In (a) and (b) only the two feet are in contact, in (c) both feet and the right hand are in contact.

- First phase set of tasks : $\{CoM\}$. A first phase to make the robot center of mass go down and on the right to be above the next foot of support. The CoM trajectory is followed through a task added in the cost function (through Lagrangian relaxation). The posture is regularized around the initial position (figure 4-a). Contacts are enforced on both feet in Eq.11.
- Second phase set of tasks : $\{CoM, LF_{SE(3)}, RH_{SE(3)}\}$. The second phase enforces only the right foot on the ground while a task is provided on the position of the left foot ($SE(3)$ task). This task is roughly constructed by interpolating the position of the foot between initial and final positions, with an offset of $10cm$ along the vertical axis. We see here that even if the reference for the foot position suffers from discontinuities, the DDP can provide feasible foot trajectories for the foot. For collision avoidance reasons, a $SE(3)$ task for the hand with relatively low weight is provided (staying at the same place). The last point of this phase is one time step before the contact creation between the flying feet and the ground. Here the set of tasks is $\{CoM, RF_{SE(3)}, RH_{SE(3)}, EE^{LF}_{se(3)}\}$. It is augmented by an impact model that enforces again the double contact of the feet and manages the different tasks weights to improve the contact. For example, regulation and $SE(3)$ task weights are increased, $EE_{se(3)}$ task for the flying foot is provided with a high cost on null velocity reference. From this point, the position is regulated around the second configuration given by the planner (figure 4-b).
- Third tasks set is $\{CoM, RH_{SE(3)}\}$. Third phase is made similarly as the first one. We only bring a new $SE(3)$ task for the right hand, referenced by an interpolation between the hand position at the beginning of this phase and the contact point position provided by the contact-planner. The final point is managed as creation contact point like previously, enforcing three contacts. At this point, tasks set is $\{CoM, RH_{SE(3)}, EE^{RH}_{se(3)}\}$.
- Fourth tasks set is $\{CoM\}$. Final phase is regulated around the next position from the contact planner (figure 4-c). CoM task is kept and the three contacts enforced. The final point is regulated around the last planner position and includes the visual task. For this point, the tasks set is $\{CoM, VT\}$. Even if VT seems to appear late in the time line, it does not make a noticeable difference in the resulting motion. The DDP propagates the image plane based non-linear visual error on previous time steps, hence the motion is smooth and the task is solved up to the concurrent tasks solutions. The visual task is made from targets that are 3D space points and projected on the image plane of the camera by a pin-hole model. We need at least four points to avoid multiple possible solutions to place the camera with respect to the points and the references. In figure 1 the green balls are the references, the blue ones are how they are positioned at the end of the motion.

The DDP algorithm is shown to converge on tasks expressing a walking pattern with null initialization of the problem (command and state over the time line). But in our case, the impact on the hand and the three contacts enforced did not allow to find a convergence without any good initialization (warm-start). The motion found is made iteratively by warm-starting the previous parts of the motion and letting null initialization for the next. For instance, in our case, the motion until the flying foot touching the ground was generated by solving the first phase alone with null initial guess, until time $t = T_{footTakeOff}$, and then solve the problem for first and second phases together, warm-starting from $t = 0$ to $t = T_{footTakeOff}$ with previous solution while initialization from $t = T_{footTakeOff}$ to $t = T_{footLanding}$ was null. Another heuristic was used to help the solver to converge: the posture regulation weight has been set higher during the complete sequence convergence research, then turned lower to avoid high velocity motion during phase transitions.

Unfortunately, collision avoidance is currently not implemented in the DDP algorithm. To generate a motion able to be tried on the robot, we check the bound limits violation and self-collision for each time step, as shown in figure 3 with the block "Trajectory validator". However, if we find out that the motion produced by the DDP violates one of these constraints, we cannot directly add the constraint to the formulation of the problem in order to produce a valid motion. We have an iterative heuristic to avoid this issue: knowing that the reference configurations given by the planner are valid and away from these bounds, we increase the weight of the postural task for the corresponding joints. In case of joint limit violation, we increase only the weight corresponding for that joint in the postural task (regularization task). If a self-collision appears, all the joints of the kinematics chain from that body to the torso are involved.

To that point, DDP algorithm generates the references for the next algorithm blocks: joint trajectories, feet trajectories and dynamic whole body CoM trajectory. To be consistent with the next section, we have to notice here that the CoM reference trajectory taking as input in the DDP algorithm is discretized at $100Hz$. The output is then naturally discretized

at $100Hz$ too. The next block of code needs $1kHz$ as input, so then the trajectories are interpolated with a cubic spline, except joint trajectories that are interpolated in linear manner after the output. Until this point, all the verifications are handled in the viewer gepetto-viewer.

B. Control architecture

The motors of the robot are position-controlled. Rather than just sending the reference joint trajectory to the motors, we employ a stabilizing control scheme in order to improve the stability all along the motion. Note that the motion generated by the DDP alone did not work with the Gazebo simulator. This stabilization was necessary to make the simulation successful.

The DDP output is first decomposed into separate kinematic tasks, which are then sent to the hierarchical inverse-kinematics solver, namely the Stack of Tasks [26]. The tasks are, in decreasing order of priority:

- Pose of each foot
- Center of Mass position
- Upper body posture
- Waist orientation

It is important to notice that the order of priority of the tasks is crucial, as each task is projected in the null space of the previous one.

The dynamic stabilization is based on the Zero Moment Point (ZMP). We are applying the ZMP control by CoM acceleration strategy [27] as described in [28]. First, the current CoM position and velocity are estimated from joint sensor readings. Then, a commanded ZMP reference is computed based on the deviation between the desired CoM and the estimated value. Further feedback is obtained from the force sensors in order to estimate the current ZMP. Finally, the CoM reference is corrected so to achieve the desired ZMP. The stabilizer can be integrated seamlessly in the hierarchical inverse kinematics architecture, by simply replacing the desired CoM reference with the adjusted one.

V. Results

We will now describe the results of this work. The DDP algorithm takes several minutes (the order of magnitude is 20 minutes) for the sum of all phases, knowing that the motion lasts almost 9 seconds. The code is currently written in python and a work to implement a c++ version is ongoing, we expect an increase of performance from this future implementation. A first stage of the simulation was made in a viewer called Gepetto-viewer (figure 1). The algorithm is based on a weighted optimization process, so errors of some tasks could remain. For instance, visual task with the relatively low weight suffers from several centimeters of errors for all the four points as it can be seen in figure 1 with big cyan and green spheres in front of the robot. The trajectories of the center of mass and the reference are also displayed, only one trajectory is visible because points are too close to be distinguished. Even if these two trajectories are very close, they are not perfectly equivalent for two reasons. Firstly, the task of the CoM

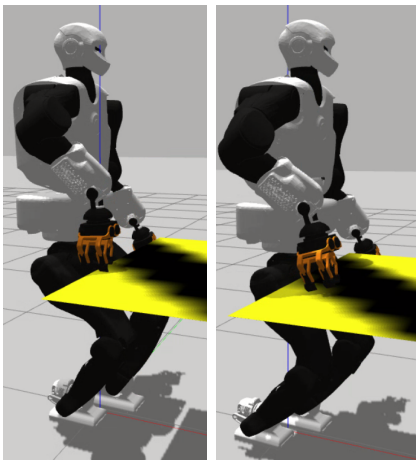


Fig. 5: Left: The robot is touching the table too early. Right: After a little bound and slide, the hand and the robot reach desired positions.

struggles against other tasks and regularizations during the optimization process. Secondly ,the DDP takes the complete dynamics of the system into account, contrary to the previous stages. So then, the DDP behaves as a dynamic filter without another calculation layer like in [1].
Concerning the simulation in a simulator, the motion was tested in Gazebo in the same way it would be tested on the real robot: Stack of Tasks controller, stabilizer and ROS architecture. The environment of simulation is a fixed plan positioned at 75cm from the ground. As shown in figure 5-Left, the robot is first touching the table before having a little leap forward of the right gripper until final stable position displayed in figure 5-Right. This could be due to the fact that the stabilizer is not a multicontact stabilizer or by using rigid contacts in Gazebo. With the input reference configuration given to the DDP, the results shown in figure 6 indicate that the forces on the right gripper are around $50N$ at the end of the motion with a peak of $250N$ on the z axis.

VI. Conclusion

We have generated a multicontact motion including visual features. The multicontact planner[2] provides a feasible CoM trajectory to be followed and reference contacts and postures of phases corresponding to contact changes, used as input for the DDP algorithm. Allowing to solve non-linear problems, it computes the complete dynamics of the robot and acts as a dynamic filter on the previous inputs. It also embeds the contact formulation directly in the dynamics.

We show how to express the visual task equations in the DDP formalism, specifically expressing their derivatives from state and control variables in the image plan. This allows us to integrate a visual task in the DDP to drive the motion to the target. The outputs of this algorithm, namely the joints and end effectors trajectories are then sent to the stabilizer to be played in a Gazebo simulation through the

[2]Open source code available at https://github.com/loco-3d/multicontact-locomotion-planning

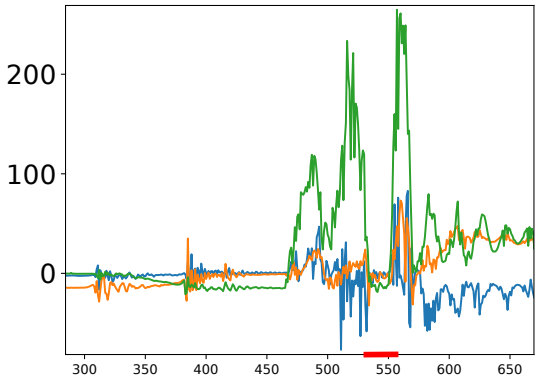


Fig. 6: Blue, orange and green curves are respectively x, y and z forces on contact hand, got from the simulation. Bounds are recognizable on z forces going to 0 after first contact with the table, this is highlighted with the red marker on simulated iteration number axis. Values are expressed in Newtons.

Stack of Tasks hierarchical controller. All the elements of the complete motion generation workflow have been integrated together in simulation to generate a multicontact motion integrating visual information. The simulation shows a slight unexpected sliding of the hand on the table, nonetheless data show that force peaks are not prohibitive to play such a motion on the robot. We consider playing this motion on the real robot with appropriate experimental setup very soon.

ACKNOWLEDGMENT

This work was partially supported by the project ROB4FAM.

REFERENCES

[1] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères, "A reactive walking pattern generator based on nonlinear model predictive control," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 10–17, 2017.

[2] M. Missura, "Analytic and learned footstep control for robust bipedal walking," Ph.D. dissertation, Bonn University, 2016.

[3] K. Imanishi and T. Sugihara, "Autonomous biped stepping control based on the LIPM potential," in *IEEE/RAS Int. Conf. on Humanoid Robotics (ICHR)*, 2018.

[4] A. Hildebrandt, D. Wahrmann, R. Wittmann, D. Rixen, and T. Buschmann, "Real-time pattern generation among obstacles for biped robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015.

[5] C. Dune, A. Herdt, O. Stasse, P.-B. Wieber, K. Yokoi, and E. Yoshida, "Cancelling the sway motion of dynamic walking in visual servoing," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 3175–3180.

[6] I. Ramirez-Alpizar, M. Naveau, C. Benazeth, O. Stasse, J.-P. Laumond, and E. Yoshida, "Motion generation for pulling a fire hose by a humanoid robot," in *IEEE/RAS Int. Conf. on Humanoid Robotics (ICHR)*, 2016.

[7] J. Carpentier, A. Del Prete, S. Tonneau, T. Flayols, F. Forget, A. Mifsud, K. Giraud, D. Atchuthan, P. Fernbach, R. Budhiraja *et al.*, "Multi-contact locomotion of legged robots in complex environments–the loco3d project," in *RSS Workshop on Challenges in Dynamic Legged Locomotion*, 2017, p. 3p.

[8] *The DARPA Robotics Challenge Finals*, vol. 34, 2017.

[9] A. Tanguy, P. Gergondet, A. Comport, and A. Kheddar, "Closed-loop rgb-d slam multi-contact control for humanoid robots," in *IEEE Int. Symposium on System Integrations (SII)*, 2016.

[10] M. Garcia, O. Stasse, and J.-B. Hayet, "Vision-driven walking pattern generation for humanoid reactive walking," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 216–221.

[11] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[12] A. Yamaguchi and C. Atkeson, "Differential dynamic programming with temporally decomposed dynamics," in *IEEE/RAS Int. Conf. on Humanoid Robotics (ICHR)*, 2015.

[13] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots," in *IEEE/RAS Int. Conf. on Humanoid Robotics (ICHR)*, 2013.

[14] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, Nov 2018, pp. 1–9.

[15] S. Tonneau, A. Del Prete, J. Pettr, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

[16] P. Fernbach, S. Tonneau, A. D. Prete, and M. Taïx, "A kinodynamic steering-method for legged multi-contact locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 3701–3707.

[17] P. Fernbach, S. Tonneau, and M. Taïx, "Croc: Convex resolution of centroidal dynamics trajectories to provide a feasibility criterion for the multi contact planning problem," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[18] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–7.

[19] J. Mirabel, S. Tonneau, P. Fernbach, A.-K. Seppälä, M. Campana, N. Mansard, and F. Lamiraux, "Hpp: A new software for constrained motion planning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 383–389.

[20] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *Proc. IEEE ICRA*, Minneapolis (MN), USA, May 2012, pp. 3859–3866.

[21] P. Fernbach, S. Tonneau, O. Stasse, J. Carpentier, and M. Taïx, "C-CROC: Continuous and Convex Resolution of Centroidal dynamic trajectories for legged robots in multi-contact scenarios," 2019, submitted. [Online]. Available: https://hal.laas.fr/hal-01894869

[22] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, S. Vijayakumar, and N. Mansard, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," 2019.

[23] 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 *IEEE International Symposium on System Integrations (SII)*, 2019.

[24] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 695–702.

[25] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[26] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *2009 International Conference on Advanced Robotics*. IEEE, 2009, pp. 1–6.

[27] S. Kajita, H. Hirukawa, K. Harada, and K. Yokoi, *Introduction to Humanoid Robotics*, ser. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2014, vol. 101.

[28] S. Caron, A. Kheddar, and O. Tempier, "Stair climbing stabilization of the HRP-4 humanoid robot using whole-body admittance control," in *IEEE International Conference on Robotics and Automation*, May 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01875387