Motion Planning for Multi-Contact 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 motion generation pipeline motivated by vision for a TALOS humanoid robot. From a starting configuration and given a set of visual features and their desired values, the problem is to find a motion which makes the robot reach the desired values of the visual features. In order to achieve a feasible Gazebo simulation with the targeted robot, we had to use 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.**

I. Introduction

The aim of this work is to generate motions for a TALOS humanoid robot starting from an initial configuration and going to a final configuration such that a set of visual features reach desired values. On flat floor and without collision, the problem can be solved using Model Predictive Control (MPC) on a Linear Inverted Pendulum [1], [2], [3], [4] where footsteps, Center-of-Mass (CoM) and Center-of-Pressure (CoP) trajectories are solved together provided a desired velocity for the CoM. It is possible to use a visual servoing task to compute the desired velocity. However this desired velocity can not always be achieved due to the constraints of the foot steps and the balance which are of high priority. To have a coherent formulation, it is necessary to have the visual tasks expressed in the MPC problem such as in [5]. In any case the resulting CoM and CoP trajectories are then provided to a local whole-body controller to perform all the tasks including the visual one [6].

Following the outcome of the DRC, recent work have been proposed to include also multiple contacts together with vision [7]. The direct application is to be able to climb stairs with potential industrial application. In [7], the main idea is for a given set of contacts to generate a dynamically consistant motion using a whole body instantaneous controller and assuming a low level position control. Exteroceptive feedback is provided by SLAM and not by visual servoing.

The strategy used in these approaches is to decouple the entire problem in smaller sub-problems that could be handled more efficiently as shown in figure 2. The sub-parts are often structured as follows: contact planning to find feasible end effector positions in the environment, centroidal trajectory optimization to get the center of mass trajectory, and whole body controller to generate the final joints trajectories to apply on the robot.

To struggle against these complex challenges, a new solver architecture is raising up in humanoid robotic field, the so called DDP described in [8]. It has been used succesfully for the DHRC in [9], and proposed also for humanoids robots in [10]. In [11], the DDP is used to generate

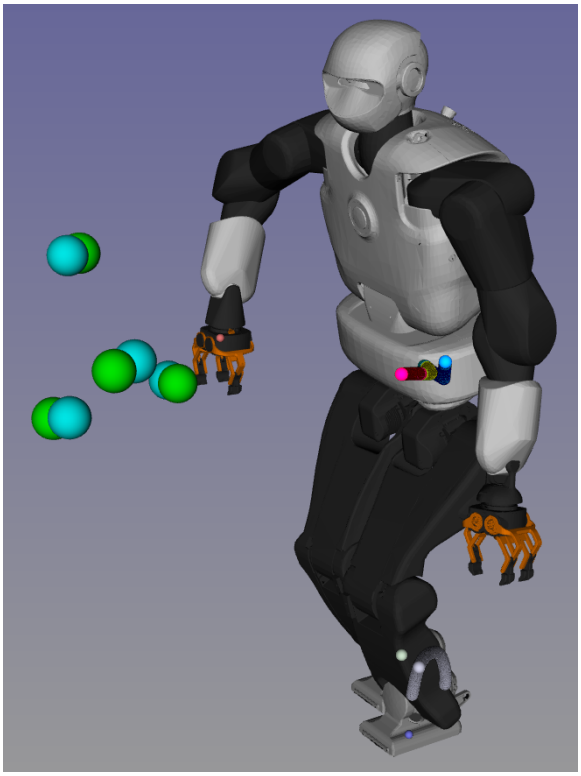


Fig. 1: Result position of the robot after simulation with three contacts and visual task. Talos' right hand (frame represented by a small red point) is equal with the target one. Center of mass trajectories are displayed : each color represents a phase corresponding to a contact change. Big green spheres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

[1] LAAS, CNRS, Toulouse, France.

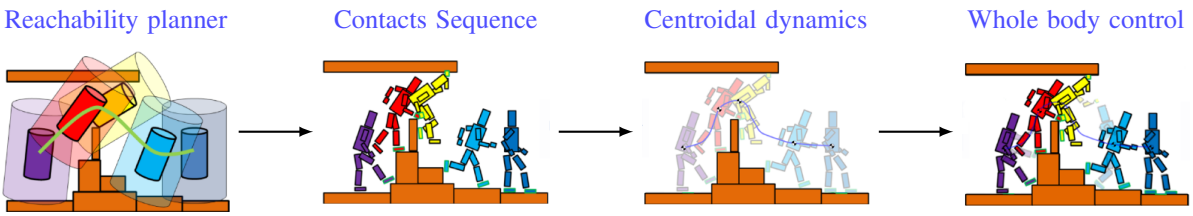


Fig. 2: Overall approach: The reachability planner takes a starting configuration ($\mathcal{C}_S$) and a goal configuration $\mathcal{C}_G$. The motion planner described in section II provides a contact sequence, and a centroidal dynamics trajectory. The DDP described in section III generates a whole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

whole body motions (corresponding to the third sub-part previously mentioned), taking into account a part of the most challenging issues such as the multi-contact and the angular momentum equations. It was validated on the HRP-2 humanoid robot. Compared to [12] the main difference lies in the fact that contact models are neither convex nor soft, they are directly provided by the multi-contact planner. We apply the same strategy in this work. We pushed even further as the multi-contact planner provides also a centroidal dynamics (CoM/CoP) trajectory.

In this work we report our first tests in integrating a fast multi-contact planner used to set a DDP problem which in turns provides reference trajectories to a local whole body instantaneous controller. It was tested in dynamical simulation (Gazebo/ODE) on the TALOS humanoid robot.

In section II we briefly explain how the multi-contact planner works. In Section III we give some reminders about DDP algorithm and visual servoing elements to understand how it can be integrated. Experiments and results are shown respectively in Section IV and V using the robot Pyrene with multi-contact and visual servoing tasks.

II. Multicontact planner

The complete architecture of the multi-contact planner presented in [13] is shown in figure 3. The following paragraphs describe briefly each method of the architecture and refer the interested reader to the papers introducing this methods.

A. Guide path

The first bloc of the figure 3 produces a rough guide trajectory for the root of the robot. The method RB-RRT was first proposed in [14] and then extended to a kinodynamic version in [15]. This method plan a trajectory for the center of a simplified model of the robot, using an 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

Once this guide trajectory is found the second bloc of the framework needs to find a sequence of feasible contacts. The contact generation method presented in [14] produces a sequence of whole body configurations in contact, such that there is only one contact change between each adjacent configuration. This method generates contact candidates using a set of configurations candidates built offline. It is able to consider each limb separatly allowing fast exploration. The reachable space of each limb is intersected with the environment while the origin of the robot follows the guide computed previously in order to find configuration candidates close to the contact. Then, whole body configurations in contact are found by inverse kinematics projection from this candidates.

For every adjacent generated configuration in contact, the algorithm has to check if there exists a motion that connects these two configurations. This is decided by solving a problem which is a convex reformulation of the multi-contact centroidal dynamic trajectory generation problem [16].

C. Centroidal trajectory

The centroidal trajectory is generated with the method proposed in [17]. 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 maximising a tailored cost function. This method can generate centroidal trajectory for multi-contact scenario in real-time thanks to a convex relaxation of the problem.

D. Validation

From the contact sequences it is possible to generate the end-effectors trajectories imposing zero velocity and zero acceleration at the start and at the end of the contacts. The CoM trajectory is provided by the solution of the centroidal dynamic problem presented in the previous paragraph. Considering a whole body controller (Kinematics or OSID), it is possible to solve a problem tracking the trajectories (CoM and end-effectors) in order to find the final articular trajectories. From this the collision library FCL [18] checks if the robot collides or not with the environment. The trajectory is validated if not any collision is detected.

III. DDP and visual servoing

In this section we describe our visual servoing approach under multi-contact events based on DDP. For that, we

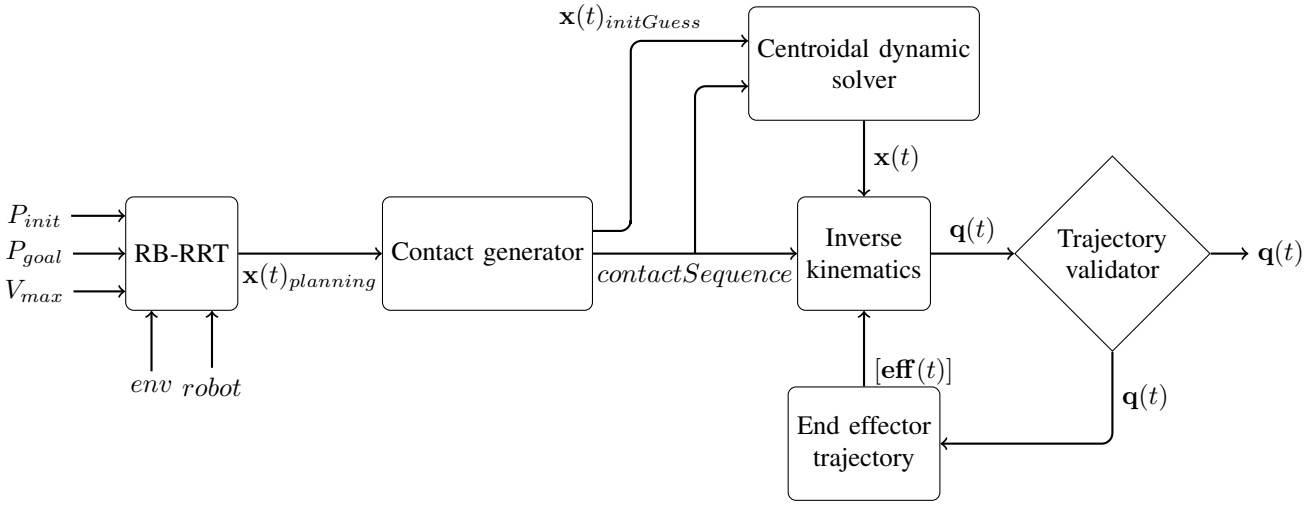


Fig. 3: Overview of the multi contact motion planner

first introduce our DDP algorithm tailored to mutiphase rigid dynamics [11]. And later, we explain the visual task formulation within our multi-contact DDP. This work is based on the DDP solver implemented in Crocoddyl [19], which computes efficiently the rigid body dynamics and its derivatives using Pinocchio [20].

A. Differential dynamic programming

DDP belongs to the family of Optimal Control (OC) and trajectory optimization [8]. It locally approximates the optimal flow (feedback gains), and as a consequence, the OC problem is splitted into simpler and smaller subproblems (sparse structure). The DDP promises to handle whole-body MPC on a humanoid thanks to its sparse structure [10]. However, the main drawback lies on the fact that it poorly handles constraints.

Let's consider a generic multi-contact OC problem as follows:

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

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 $\boldsymbol{\tau}$ are 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 tuple of states and controls along the defined horizon. Note that both – cost and dynamics – often are time varying functions.

DDP breaks the dynamic problem into simpler subproblem 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 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 (a.k.a. as 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 and constraints

A task is usually formulated as a regulator:

$$h_{i,task}(\mathbf{x_i}, \mathbf{u_i}) = s^*_{task} - s_{task}(\mathbf{x_i}) \tag{8}$$

where s^*_{task} is a desired value vector for a feature and $s_{task}(\mathbf{x_i})$ the value vector of this feature according to state $\mathbf{x_i}$. As one wants to minimize this value such that $lim_{t\to+\inf} h(x,u) = 0$, the task at each node is implemented as a penalty:

$$l_i(\mathbf{x_i}, \mathbf{u_i}) = \sum_{j \in Tasks} w_{i,j} h_{i,j}(\mathbf{x_i}, \mathbf{u_i}) \tag{9}$$

with $w_{i,j}$ the weight assigned at time i to task j. In our case $Tasks \subseteq \{CoM, RH_{SE(3)}, RF_{SE(3)}, LF_{SE(3)}, EE^{eeName}_{se(3)}, VT\}$ with CoM the task tracking the Center-of-Mass, $RH_{SE(3)}$ the task tracking the right hand pose, $RF_{SE(3)}$ the task tracking the right foot pose, $LF_{SE(3)}$ the task tracking the left foot pose, $EE^{eeName}_{se(3)}$ the task tracking an end effector velocity wich sould be null during the impact, $eeName$ is the name of the end effector (RH for right hand for instance), VT the visual task expressed in the image plan.

C. Handling dynamical constraints

Althought this basic formulation of DDP does not handle constraints it is possible to integrate them in the cost function using Lagrangian relaxation. Thus [11] modified the problem formulation to enforce contacts. The dynamic of robot is expressed as follows :

$$\mathbf{M}\dot{\nu} = \boldsymbol{S\tau} - \boldsymbol{b} + \mathbf{J}_c\boldsymbol{\lambda} \tag{10}$$

with $\mathbf{M}$ the inertial matrix, $\dot{\nu}$ the derivative of the state velocity, $\boldsymbol{S}$ the selection matrix corresponding to the actuated degrees of freedom (dof) , $\boldsymbol{\tau}$ the vector of torques of actuated joints and $\mathbf{b}$ the bias term consitting in coriolis and gravitationnal effects. $\mathbf{J}_c\boldsymbol{\lambda}$ is the term expressing the external forces at joint level. $\mathbf{J}_c$ is the stacked Jacobian corresponding to application points, $\boldsymbol{\lambda}$ is homogeneous is the positive value of the force applied to the application point, on the force application direction. In the formulation this term is viewed as the dual variables. To constraint the dynamic of the contact, [11] express null acceleration at the contact point by :

$$\begin{aligned} (\dot{\mathbf{J}_c} v_c) &= \mathbf{0} \\ &\Leftrightarrow \\ \mathbf{J}_c\dot{\mathbf{v}}_c &= -\dot{\mathbf{J}_c}\mathbf{v} \end{aligned} \tag{11}$$

With 11 and 10, using Gauss principle, KKT conditions are given by :

$$\begin{bmatrix} \mathbf{M} & \mathbf{J}_c^T \\ \mathbf{J}_c & \mathbf{0} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{\nu}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S\tau} - \boldsymbol{b} \\ \dot{\mathbf{J}_c}\mathbf{v} \end{bmatrix} \tag{12}$$

To take into account the dual variable in the resolution of the problem, dynamic equation is augmented as follows :

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{f}(\mathbf{x_i}, \mathbf{u_i}) \\ \boldsymbol{\lambda}_i &= \mathbf{g}(\mathbf{x_i}, \mathbf{u_i}) \end{aligned} \tag{13}$$

where g is the dual solution of 12. The action-value function Q and the cost l are now depending on $\boldsymbol{\lambda}$. Making the assumption that second derivative of the dynamic f are zero as ILQR algorithm does, but taking into account the derivative of the cost l by $\boldsymbol{\lambda}$, the new equations of the second order approximation are :

$$Q_x = l_x + f_x^T V'_x + g_x^T l_\lambda \tag{14}$$
$$Q_u = l_u + f_u^T V'_x + g_u^T l_\lambda \tag{15}$$
$$Q_{xx} = l_{xx} + f_x^T V'_{xx} f_x + g_x^T l_{\lambda\lambda} g_x \tag{16}$$
$$Q_{uu} = l_{uu} + f_u^T V'_{xx} f_u + g_u^T l_{\lambda\lambda} g_u \tag{17}$$
$$Q_{ux} = l_{ux} + f_u^T V'_{xx} f_x + g_u^T l_{\lambda\lambda} g_x \tag{18}$$

This method takes into account the contact constraints in the dynamic level and prevent the solver to allocate ressources to manage these constraints during solving run. Since the main principles underlying DDP are exposed in this paragraph, visual servoing is briefly presented in the next paragraph in order to derive its integration and implementation.

D. Visual servoing

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

Giving 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 $s*$ and the actual features s. These last 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 :

$$e = s - s* \tag{19}$$

In our case, $s*$ is considered as fixed, not depending on the time. The error e is also considered as the residual of the cost l defined by :

$$l = \frac{1}{2} \|e\|^2 \tag{20}$$

The model commonly used is a first order motion model:

$$\dot{e} = L_e v_c \tag{21}$$

where v_c is the velocity of the camera in the camera frame, and Le is the interaction matrix. This matrix can be considered as the features Jacobian By derivating the position of one feature in the 3D space, [21] has shown Le can be written as follow :

$$L_e = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{22}$$

Now, let's call J_c the Jacobian of the camera in the camera frame, and $\dot{q}$ the velocity of the degrees of freedom (DoF) Combining the well known expression $v_c = J_c \dot{q}$ with (21), we find :

$$\dot{e} = L_e J_c \dot{q} \tag{23}$$

Contrary to the common visual servoing command law that enforces the exponential decrease by writting 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:

$$\frac{\partial e}{\partial \begin{bmatrix} q \\ \dot{q} \end{bmatrix}} = \begin{bmatrix} 0_{2 \times n_q}, L_e J_c \end{bmatrix} \tag{24}$$

$$\frac{\partial e}{\partial u} = 0_{2 \times n_q - 6} \tag{25}$$

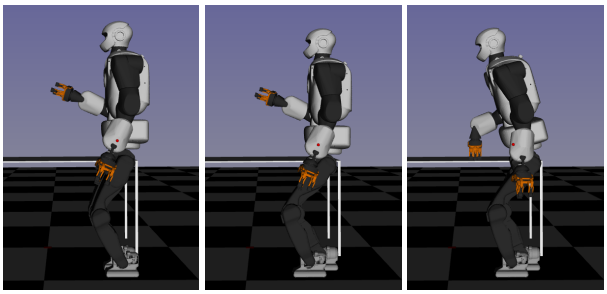
The Hessian of the visual task (i.e. $l_{\mathbf{xx}}$, $l_{\mathbf{xu}}$ and $l_{\mathbf{uu}}$) are zeros.
All the elements are gathered to implement the visual task in the DDP algorithm. Let's now explain the experimental conditions and tasks we have created for this work.

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 generated appropriate motion and the results obtained in simulation and then on the real robot.

A. Simulation setup

In our setup, Talos begins in an initial double support standing configuration. It should 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 visual servo the target with predefined desired features positions in the image plane. The main goal here is to be able to see an object while the posture needs a third (or more) contact.



(a) (b) (c)

Fig. 4: Sequence of configuration 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.

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

In Crocoddyl, for each time step the dynamic 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 follow:

- First phase set of tasks : $\{CoM\}$. A first phase to make the robot center of mass goes 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.12.
- 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 foot between initial and final position, with an offset of 10cm along the vertical axis. We see here that even the reference for the foot position suffers from discontinuities, 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 as 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 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 inter-

polation 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 set tasks is $\{CoM, VT\}$. Even if it seems to appear lately 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 multiples 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 initialisation 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 phase together, warmstarting from $t = 0$ to $t = T_{footTakeOff}$ with previous solution while initilization 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 checked the bounds limits violation and self-collision for each time step. For that purpose we used the tools provided by the Humanoid Path Planner framework [22]. However, if we found out that the motion produced by the DDP violate one of this constraints, we cannot directly add the constraint to the formulation of the problem in order to produce a valid motion. We found an iterative heuristic to avoid this issue: knowing that the reference configurations given by the planner are valid and away from these bounds, we increased the weight of the postural task for the corresponding joints. In case of joint limit violation, we increased only the weight corresponding for that joint in postural task (regularization task). If an autocollision appeared, all the joints of the kinematics chain from that body to the torso have been involved.

To that point, DDP algorithm generates the references for the next algorithm blocs: 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 bloc of code needs 1kHz as input, so then the trajectories were interpolated with the cubic mode of scipy interpolation, except joint trajectories that were interpolated in linear manner after the output. Until this point, all the verification were 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 successfull.

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 [23]. 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 [24] as described in [25]. First, the current CoM position and velocity are estimated from joint sensors 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 took several 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 futur implementation. A first stage of simulation was made in a viewer called Gepetto-viewer. The algorithm is based on a weighted optimization process so errors of

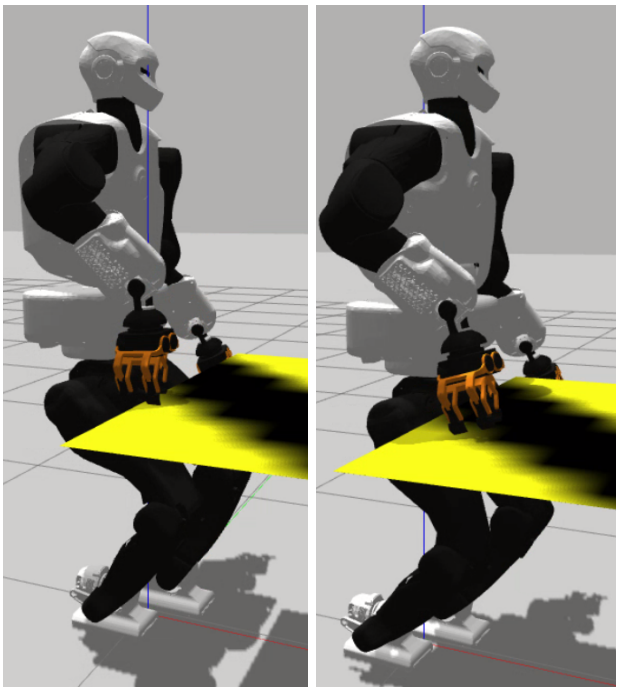


Fig. 5: Left: The robot is touching the table too early. Right: After a little bound and slide, the hand and the robot reach desired positions

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 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 struggles against other task and regularization during the optimization process. Secondly the DDP takes the complete dynamic 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 motion of the hand on the table is not expected and may be due to a lack of a $SE(3)$ hand task provided directly in stack of task controller. With the input reference configuration given to the DDP, the results shown in 6 indicate that the forces on the right gripper are around 50N at the end of the motion with 250 in peack on z axis.

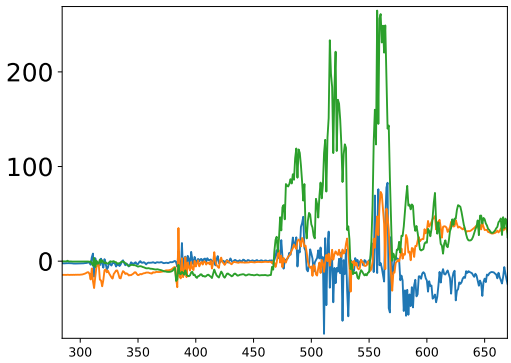


Fig. 6: Blue, orange and green curves are respectively x, y and z forces on contact hand, got from simulation. Bounds are recognizable on z forces going to 0 after first contact with the table. Values are expressed in Newtons.

VI. Conclusion

We have generated a multicontact motion motivated by vision. The multicontact planner provides a feasible CoM trajectory to be followed and reference 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.

By expressing its derivatives on state and control in the image plan, a visual task is integrated in the DDP to drive the motion to the target. The outputs of this algorithm, namely the joints and end effector trajectories are then sent to the stabilizer to be played in a Gazebo simulation through the Stack of Tasks hierarchical controller. 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.

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] 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.

[6] D. J. Agravante, A. Cherubini, A. Bussy, P. Gergondet, and A. Kheddar, "Collaborative human-humanoid carrying using vision and haptic sensing," in *IEEE/RAS Int. Conf. on Robotics and Automation (ICRA)*, 2014.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] I. Mordatch, J. M. Wang, E. Todorov, and V. Koltun, "Animating human lower limbs using contact-invariant optimization," *ACM Trans. Graph.*, vol. 32, no. 6, 2013.

[13] 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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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