AMR project Documentation

Atanasious 1861248
Bozza 1836119
Taliani 1841465

March 2, 2023

Abstract

The present paper is a detailed description of code produced to reach the goal of our project: Stable walking trajectory planning with multiple contacts using Gravito-Inertial Wrench Cone.

1 Introduction

Humanoid robots are a class of robots that are designed to imitate human-like movements and perform tasks in environments that are specifically designed for humans. These robots are inherently complex systems to deal with, particularly when it comes to motion planning. In order to move, legged robots have to continuously change their contact with the environment, this means that they must carefully manage their force exchange with the ground and surrounding objects in order to maintain balance and avoid falling.
When a link of the robot is in contact with the ground, a distribution of friction and reaction forces is generated on the contact surface. Thankfully, when the contact area is convex, it is possible to consider only the forces exerted to the vertex of the support polygon.
As in many cases to plan the robot motions we refer to the position of the center of mass (CoM) and constrain resultant accelerations and torques applied to it. In this particular case, in order to plan motions of the Humanoid robot, we first plan a sequence of contacts that the robot will undergo and then we solve an optimization problem to obtain feasible trajectories for the robot's CoM that brings to a dynamically stable walking for the whole body.
In the following section, we present in detail how such constraints are generated starting from the friction cones of each contact point and how they are included in the final optimization problem.

2 Gravito-Inertial Wrench Cone

Under the assumption that torque limits are sufficiently large, the stability of the robot's contacts depends only on the gravito-inertial wrench, a six-dimensional object that represents the combined effect of gravity and inertia on the robot's motion. This wrench should lie within a convex cone called the Gravito-Inertial Wrench Cone (GIWC), which is generated by the contact constraints. The GIWC provides a "universal stability criterion" for ensuring the robot's stability during movement, as it represents the set of all possible gravito-inertial wrenches that can be achieved while maintaining stability.

By constraining the robot's motion to remain within the GIWC, it is possible to plan stable and efficient trajectories for the robot's CoM that ensure it remains dynamically stable throughout the entire walking process. This approach to stability analysis is a critical concept in the analysis and planning of humanoid robot motion, as it provides a means of ensuring the robot remains stable and safe while performing complex tasks in a variety of environments.

2.1 Friction Cones and Surface Wrenches

As already mentioned in the introduction when a link is in contact with the environment we have a surface in which each point contributes to generate a friction and a reaction force.
We assume as it usually happens that each contact surface is a convex polygon. We want to prevent the robot to slip on the contact surface, hence we enforce static friction model to each contact point. We consider a contact force f to be *valid* when it is such that:

$$\begin{aligned} &\|f^t\| \leq f^n \mu_s \\ &f^n \geq 0 \end{aligned} \tag{1}$$

where μ is the friction coefficient, f^n and f^t denote respectively the normal and tangential components of the force with respect to a reference frame attached to the contact surface. In the following development, we will use the polyhedral approximation of the latter inequality, i.e.,

$$|f_x| \leq \mu f^n, \quad |f_y| \leq \mu f^n \tag{2}$$

where $\boldsymbol{f}^t = (f_x, f_y)$

Each cone has two possible representations the face and the span form.
The face form of a cone is given by the vectors which indicates a face of the cone by it's normal vector. The span form is instead represented by vectors which spanned generate the same space.
In the following analysis it is fudamental the ability to compute a form from the others since it will be important to exploit their advantages.

The reaction forces and torques given by the contact can be represented by a wrench $w = \begin{bmatrix} f \\ \tau \end{bmatrix}$ that is the result of contact forces at polygon's vertices.
A contact wrench is defined *valid* when it is generated by a set of *valid* contact forces.

To generate a *valid* contact wrench for each surface we have to consider constraints given by friction cones at each contact point and we sum their effects on the Center of Pressure (CoP) of the surface generating a convex cone which defines the region containing acceptable wrenches. This 6D cone can be generated for each contact surface, as described in [1], and we implemented the construction of such surface cone in two different ways.
First of all, for each vertex of the contact 2D polygon we compute the friction cone, as the cone that has the tangent of the angle between itself and the contact surface equal to μ, friction coefficient. Through the MATLAB *cylinder* function we generate a set of points that belongs to the pyramidal cone approximation. These are 3D cones that we can express in span form, given that from the cone points that we found, we can write the four vectors that span the approximated cone (a square pyramid).

The surface cone is computed with respect to its CoP p, that is considered as the mean in the three space coordinate of the contact points. For a general set of contact forces applied to the vertices of the polygon, the resultant force for the surface is a direct sum of contact forces while the total torque is computed as $\tau = r_i \times f$ where r_i is $r_i = p_i - p$. This summation procedure is linear and can be encoded in A_{surf}:

$$A_{surf} = \begin{bmatrix} I_{3\times3} & \dots & I_{3\times3} & \dots \\ S(r_1) & \dots & S(r_i) & \dots \end{bmatrix} \tag{3}$$

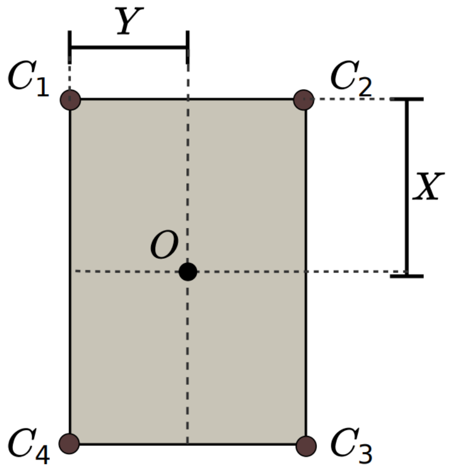
where $S(r_i)$ is the skew symmetric matrix of the vector r_i.
To get the surface cone we applied this linear mapping to each generating vector of the friction cones, multiplying A_{surf} by the stack of the span vectors:

$$V_{surf} = A_{surf} \begin{bmatrix} V^1_{point} \\ \dots \\ V^i_{point} \\ \dots \end{bmatrix} \tag{4}$$

A second approach is to consider the method described in [2], in which an algorithm for the computation of the Face form of a rectangular surface is presented. As a matter of fact the face form of each surface is found as:

$$
\boldsymbol{U} = \begin{bmatrix}
-1 & 0 & -\mu & 0 & 0 & 0 \\
+1 & 0 & -\mu & 0 & 0 & 0 \\
0 & -1 & -\mu & 0 & 0 & 0 \\
0 & +1 & -\mu & 0 & 0 & 0 \\
0 & 0 & -Y & -1 & 0 & 0 \\
0 & 0 & -Y & +1 & 0 & 0 \\
0 & 0 & -X & 0 & -1 & 0 \\
0 & 0 & -X & 0 & +1 & 0 \\
-Y & -X & -(X+Y)\mu & +\mu & +\mu & -1 \\
-Y & +X & -(X+Y)\mu & +\mu & -\mu & -1 \\
+Y & -X & -(X+Y)\mu & -\mu & +\mu & -1 \\
+Y & +X & -(X+Y)\mu & -\mu & -\mu & -1 \\
+Y & +X & -(X+Y)\mu & +\mu & +\mu & +1 \\
+Y & -X & -(X+Y)\mu & +\mu & -\mu & +1 \\
-Y & +X & -(X+Y)\mu & -\mu & +\mu & +1 \\
-Y & -X & -(X+Y)\mu & -\mu & -\mu & +1
\end{bmatrix}
$$

(a) Face form matrix



(b) Contact polygon

In this case the span form of the resulting cone is computed by deploying the *cdd* library double description conversion, as it is done in the analyzed paper simulations [1].

2.2 Gravito-Inertial Wrench Cone

Once the wrench cone of each contact surface is computed we need to join the given constraints in order to characterize the admissible forces and torques for the CoM. The arising cone is computed for each stance (a stance is a particular configuration of contacts), with the following procedure.

$$
A_{stance} = \begin{bmatrix} -R_1 & 0_{3\times3} & \dots & -R_i & 0_{3\times3} & \dots \\ -R_1 \cdot S(CoP_1) & -R_1 & \dots & -R_i \cdot S(CoP_i) & -R_i & \dots \end{bmatrix} \tag{5}
$$

This relation allows us to bring V^i_{surf} expressed in the $i-th$ surface frame directly to the CoM expressed in the laboratory frame.

$$
V_{GIWC} = A_{stance} \begin{bmatrix} V^1_{surf} & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & V^i_{surf} & 0 \\ 0 & 0 & 0 & \ddots \end{bmatrix} \tag{6}
$$

In order to have a more suitable representation of the cone that allows to directly verify whether the Gravito-Inertial wrench of the robot belongs or not to it's feasibility region, we convert through the *cdd* library function, this span into a face form of the GIWC.

As described in [1], the computations that are operated each time that the stance (the contact surfaces) changes, are the computation of A_{stance} and V_{GIWC} with its face form U_{GIWC} to enforce in the optimal control problem the constraint on the current wrench:

$$
U_{GIWC} w \leq 0 \tag{7}
$$

2.3 Optimization Problem

At this point we can set the Optimization problem to find a feasible trajectory for the CoM, imposing the constraint (7).

The code provided is a MATLAB script, called ”traj_planning”, that utilizes the CasADi library to solve an optimization problem related to robot locomotion. The goal of the optimization problem is to plan a walking trajectory for a humanoid robot from an initial position, with both feet on the floor, to a final point where the robot must come to a stop.

To achieve this, the script begins by setting several parameters, such as:

- n_{stance}: desired number of stances or steps to take
- t_{stance}: desired time per step
- ss_{time}: time for single support
- ds_{time}: time for double support
- T: total simulation time
- dT: integration step
- m: mass of the robot
- g: gravity vector
- eps: tolerance for CoM height

These parameters are used throughout the script to define the problem and constraints.

Next, the script generates all of the possible contact surfaces for the robot's feet to interact with during walking. These surfaces are obtained through a function called "get_surf" and are stored in a variable called "surfaces".

After setting the parameters and generating the contact surfaces, the script begins to use the CasADi library to solve the optimization problem. The first step in doing so is to define the state space and input as symbolic matrices. The state space consists of a vector of six elements, representing the position and velocity of the robot in three-dimensional space. The input is a vector of three elements, representing the acceleration applied to the robot.

The next step is to define the Gravito-Inertial wrench cone, which is a constraint on the forces and torques that the robot CoM is able to exchange while maintaining stability. This is achieved through a function called "gravito_inertial_wrench", which takes as input the contact surface and the center of mass of the robot and returns the Face form of the Gravito-Inertial wrench cone.

The robot dynamics are then defined using a continuous function called "continuous_dynamics", which represents the equations of motion for the robot. In this case, the dynamics are represented by a simple integrator.

The dynamics are then discretized using an explicit Euler method and a function called "discrete_dynamics". This function takes the state and input at the current time step and returns the state at the next time step.

Once the dynamics are defined, the optimization problem can be formulated. The objective function of the optimization problem is to minimize the sum of the squares of the input vector U, which is defined as:

$$L = \sum_{i=1}^{N} \boldsymbol{u_i^T} \boldsymbol{u_i} \tag{8}$$

For each time step, the script computes the Gravito-Inertial wrench cone using the current contact surface and center of mass of the robot, knowing that the stance for each time interval is determined by calling the "find_stance" function. The constraint on the wrench cone is then enforced using the "opti.subject_to" function. The dynamics of the robot are also enforced using this function, as well as a constraint that ensures the robot maintains a positive velocity in the walking direction.

Constraints are also imposed on the height of the robot's center of mass, with upper and lower limits set to within 5 units of a specified goal height. The initial and final positions of the robot are also imposed using "opti.subject_to" functions.

Finally, the optimization problem is solved using the IPOPT solver, and the solution is obtained using the "opti.solve()" function. The solution provides a walking trajectory for the robot that satisfies all of the specified constraints.

References

[1] Stephane Caron, Quang Cuong Pham, and Yoshihiko Nakamura. Leveraging cone double description for multi-contact stability of humanoids with applications to statics and dynamics. 07 2015.

[2] Stéphane Caron, Quang-Cuong Pham, and Yoshihiko Nakamura. Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5107–5112, 2015.