Collision Avoidance Constraints in HuMoTo

Matteo Ciocca

***Abstract*— Integrate in HumoTo the collision avoidance constraints proposed in [1], [2].**

I. Control Problem in HuMoTo, WPG.04

Step Selections:

$$\hat{P} = V_0 + VP \tag{1}$$

V_0 current step, P future steps, (V_0, V) proper selection matrices. CoP positions along the horizon:

$$\begin{bmatrix} z_1 \\ \vdots \\ z_N \end{bmatrix} = V_0 + VP + \operatorname*{diag}_{k=1,\dots,N}(R_{\hat{p}_k})Z \tag{2}$$

Model of the system:

$$\begin{aligned} \hat{c}_{k+1} &= A\hat{c}_k + Bz_{k+1} \\ \dot{z}_k &= D\hat{c}_k + Ez_{k+1} \end{aligned} \tag{3}$$

Condensation:

$$\begin{aligned} \hat{C} &= U_x\hat{c}_0 + U_u\left(V_0 + VP + \operatorname*{diag}_{k=1,\dots,N}(R_{\hat{p}_k})Z\right) \\ \dot{Z} &= O_x\hat{c}_0 + O_u\left(V_0 + VP + \operatorname*{diag}_{k=1,\dots,N}(R_{\hat{p}_k})Z\right) \end{aligned} \tag{4}$$

The unknowns Z and P can be grouped together forming the control vector X:

$$\begin{aligned} \hat{C} &= \underbrace{\begin{bmatrix} U_u \operatorname{diag}_{k=1,\dots,N}(R_{\hat{p}_k}) & U_uV \end{bmatrix}}_{S} \underbrace{\begin{bmatrix} Z \\ P \end{bmatrix}}_{X} + \underbrace{U_x\hat{c}_0 + U_uV_0}_{s} \\ \dot{Z} &= \underbrace{\begin{bmatrix} O_u \operatorname{diag}_{k=1,\dots,N}(R_{\hat{p}_k}) & O_uV \end{bmatrix}}_{S_{\dot{z}}} \underbrace{\begin{bmatrix} Z \\ P \end{bmatrix}}_{X} + \underbrace{O_x\hat{c}_0 + O_uV_0}_{s_{\dot{z}}} \end{aligned} \tag{5}$$

Resulting in:

$$\begin{aligned} \hat{C} &= SX + s \\ \dot{Z} &= S_{\dot{z}}X + s_{\dot{z}} \end{aligned} \tag{6}$$

The velocity of the CoM can be extracted with a selection matrix (for the velocity) I_v:

$$\dot{C} = \operatorname*{diag}_N(I_v)\hat{C} = \operatorname*{diag}_N(I_v)(SX + s) = S_vX + s_v \tag{7}$$

For the position we use the selection matrix I_p:

$$C = \operatorname*{diag}_N(I_p)\hat{C} = \operatorname*{diag}_N(I_p)(SX + s) = S_pX + s_p \tag{8}$$

Vector $\hat{C}$ and C contains the followings:

$$\hat{C} = \begin{bmatrix} c^x_{k+1} \\ \dot{c}^x_{k+1} \\ \ddot{c}^x_{k+1} \\ c^y_{k+1} \\ \dot{c}^y_{k+1} \\ \ddot{c}^y_{k+1} \\ c^x_{k+2} \\ \dot{c}^x_{k+2} \\ \ddot{c}^x_{k+2} \\ \vdots \\ c^y_{k+N} \\ \dot{c}^y_{k+N} \\ \ddot{c}^y_{k+N} \end{bmatrix} \in \mathbb{R}^{3N}, \; C = \begin{bmatrix} c^x_{k+1} \\ c^y_{k+1} \\ c^x_{k+2} \\ c^y_{k+2} \\ \vdots \\ c^x_{k+N} \\ c^y_{k+N} \end{bmatrix} \in \mathbb{R}^{2N} \tag{9}$$

II. Collision Avoidance constraints

The constraints:

$$||c - m^i|| \geq d \;\in \mathbb{R} \tag{10}$$

where the m^i is the position of the i obstalce. Along the horizon we ends up with:

$$\vec{n}_k \left[S_pX + s_p - M\right] \geq D \tag{11}$$

The condense position of the obstacle along the prediction horizon is formulated as:

$$M = \begin{bmatrix} m^x_1 \\ m^y_1 \\ m^x_2 \\ m^y_2 \\ \vdots \\ m^x_N \\ m^y_N \end{bmatrix} \in \mathbb{R}^{2N} \tag{12}$$

But the condense distance vector has different size:

$$D = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \in \mathbb{R}^N \tag{13}$$

That can be translated in a (A, b) structure as:

$$\underbrace{\vec{n}_kS_p}_{A} X \geq \underbrace{D + \vec{n}_k\left(M - s_p\right)}_{b} \tag{14}$$

Creating a set of constraints of type TaskAL:

$$AX \geq b \;\;(\text{ or } b \leq AX) \tag{15}$$

Condense vector of distance from obstacle D. The matrix n_k is:

$$n_k = \begin{bmatrix} n^x & n^y \end{bmatrix} \in \mathbb{R}^{N \times 2N} \tag{16}$$

where

$$n^T = \begin{bmatrix} \frac{(c_x - (m^i)^x)}{||c - m^i||} & \frac{(c_y - (m^i)^y)}{||c - m^i||} \end{bmatrix}^T \in \mathbb{R}^2 \tag{17}$$

and condensing the $\vec{n}_k$ matrix along the horizon, we obtain:

$$\vec{n}_k = \begin{bmatrix} n_k^x & n_k^y & 0 & \dots & \dots & 0 \\ 0 & 0 & n_{k+1}^x & n_{k+1}^y & 0 & \vdots \\ \vdots & 0 & 0 & \ddots & 0 & 0 \\ 0 & \dots & \dots & 0 & n_{k+N-1}^x & n_{k+N-1}^y \end{bmatrix} \in \mathbb{R}^{N \times 2N} \tag{18}$$

The matrix $\vec{n}_k$ uses the values of C CoM and M obstacle condense positions of the previous iteration. At the first instant in time it will have as the first row:

$$\vec{n}_k(1,:) = \begin{bmatrix} (n_{-1})^x & (n_{-1})^y & \mathbb{O}^T \end{bmatrix} \tag{19}$$

Where $\mathbb{O}$ is a zero matrix of proper dimensions.

III. Code: no Field of View and only static obstacles

Obstacles are initialized as vectors (containing their position and orientations) and added as `shared_pointers` in `model` before the simulation.

A. obstacle avoidance

Inside the simulation, after the update of the model of the robot, the obstacles "are updated": the constraints are updated by the presence of the obstacles. `updateObstacles` updates the collision avoidance constraints used in the hierarchical optimization problem. This update call `resetConstraints` in case the index simulation time is 0, or `updateConstraints` otherwise. Both, are used to fill (A, b) for the collision avoidance constraints.

Once the constraints are update, inside `task_collavoidance`, the matrice (A, b) of each obstacles are called and stacked together to be included in the optimization problem.

Inside `mpc_wpg` the (S_p, s_p) of the previous iteration are updated. The previous state of the obstacles are not updated because all visible and static.

B. model of the obstacle

For this scenario, the obstacle is modeled as a static circle in the space with the class `ObstacleCircle`.

REMARK: the orientation then does not make sense for this model.

References

[1] N. Bohorquez, A. Sherikov, D. Dimitrov, and P.-B. Wieber, "Safe navigation strategies for a biped robot walking in a crowd," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, nov 2016. [Online]. Available: https://doi.org/10.1109/humanoids.2016.7803304

[2] S. A. Homsi, A. Sherikov, D. Dimitrov, and P.-B. Wieber, "A hierarchical approach to minimum-time control of industrial robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2016. [Online]. Available: https://doi.org/10.1109/icra.2016.7487386