ORCA Documentation

Release alpha

Antoine Hoarau

May 30, 2018

Installation and Configuration

ORCA is a c++ whole-body reactive controller meant to compute the desired actuation torque of a robot given some tasks to perform and some constraints.

The problem is written as a **quadratic problem** :

$$\begin{aligned} \min_x \frac{1}{2} x^t H x + x^t g \\ \text{subject to} \\ lb \leq x \leq ub \\ lb_A \leq Ax \leq ub_A \end{aligned}$$

- `x` the optimisation vector
- `H` the hessian matrix $(size(x) \times size(x))$
- `g` the gradient vector $(size(x) \times 1)$
- `A` the constraint matrix $(size(x) \times size(x))$
- `lb` and `ub` the lower and upper bounds of `x` $(size(x) \times 1)$
- `lbA` and `ubA` the lower and upper bounds of `A` $(size(x) \times 1)$

Tasks are written as **weighted euclidian distance function** :

$$w_{task} \|\mathbf{E}x + \mathbf{f}\|^2_{W_{norm}}$$

- `x` the optimisation vector, or **part** of the optimisation vector
- `E` the linear matrix of the affine function $(size(x) \times size(x))$
- `f` the origin vector $(size(x) \times 1)$
- `w task` the weight of the tasks in the overall quadratic cost (scalar $[0:1]$)
- `W norm` the weight of the euclidean norm $(size(x) \times size(x))$

Given n_t tasks, the **overall cost function** is such that:

$$\frac{1}{2} x^t H x + x^t g = \frac{1}{2} \sum_{i=1}^{n_t} w_{task,i} \|\mathbf{E}_i x + \mathbf{f}_i\|^2_{W_{norm,i}}$$

Constraints are written as **double bounded linear function** :

$$lb_C \leq Cx \leq ub_C$$

- `C` the constraint matrix $(size(x) \times size(x))$
- `lbC` and `ubC` the lower and upper bounds of `A` $(size(x) \times 1)$

The remainder of the documentation describes "classical" tasks and cosntraints which one may want to define

CHAPTER 1

Optimisation Vector

The optimisation vector in the quadratic problem is written as follows :

$$X = \begin{pmatrix} \dot{\nu}^{fb} \\ \dot{\nu}^{j} \\ \tau^{fb} \\ \tau^{j} \\ {}^{e}w_0 \\ \vdots \\ {}^{e}w_n \end{pmatrix}$$

- $\dot{\nu}^{fb}$: Floating base joint acceleration (6×1)
- $\dot{\nu}^{j}$: Joint space acceleration $(n_{dof} \times 1)$
- τ^{fb} : Floating base joint torque (6×1)
- τ^{j} : Joint space joint torque $(n_{dof} \times 1)$
- ${}^{e}w_n$: External wrench (6×1)
- τ^{fb} : Floating base joint torque (6×1)
- τ^{j} : Joint space joint torque $(n_{dof} \times 1)$
- ${}^{e}w_n$: External wrench (6×1)

In ORCA those are called *Control variables* and should be used to define every task and constraint. In addition to those necessary variables, you can specify also a combination :

- $\dot{\nu}$: Generalised joint acceleration, concatenation of $\dot{\nu}^{fb}$ and $\dot{\nu}^{j}$ $(6 + n_{dof} \times 1)$
- τ : Generalised joint torque, concatenation of τ^{fb} and τ^{j} $(6 + n_{dof} \times 1)$
- X : The whole optimisation vector $(6 + n_{dof} + 6 + n_{dof} + n_{wrenches} 6 \times 1)$
- ${}^{e}w$: External wrenches $(n_{wrenche} 6 \times 1)$
- X : The whole optimisation vector $(6 + n_{dof} + 6 + n_{dof} + n_{wrenches} 6 \times 1)$
- ${}^{e}w$: External wrenches $(n_{wrenche} 6 \times 1)$

CHAPTER 2

Cartesian Acceleration

$$w_{task}.\|\mathbf{E}x + \mathbf{f}\|_{W_{norm}}$$

$$\underset{n\times 1}{\mathrm{Y}} = \underset{n\times p}{X} \times \underset{p\times 1}{\theta} + \underset{n\times 1}{\varepsilon}$$

CHAPTER 3

Dynamics Equation

- **Control variable** : X (whole optimisation vector)
- **Type** : Equality constraint
- **Size** : $ndof \times size(X)$

$$\begin{bmatrix} -M & S_\tau & J_{e_w} \end{bmatrix} X = C + G$$

```
orca::constraint::DynamicsEquationConstraint dyn_eq;
dyn_eq.loadRobotModel( urdf );
dyn_eq.setGravity( Eigen::Vector3d(0,0,-9.81) );
dyn_eq.update(); // <-- Now initialized

dyn_eq.activate(); // <-- Now activated
dyn_eq.insertInProblem(); // <-- Now part of the optimisation problem
```