

---

# ORCA Documentation

*Release Alpago*

**Antoine Hoarau**

**Jun 03, 2018**



---

## Getting Started

---

<b>1</b>	<b>Motivation</b>	<b>3</b>
1.1	Table of Contents . . . . .	3
<b>2</b>	<b>Authorship</b>	<b>43</b>
2.1	Maintainers . . . . .	43
2.2	Contributors . . . . .	43
2.3	Related Publications . . . . .	43
2.4	Partner Institutions . . . . .	43





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.



# CHAPTER 1

---

## Motivation

---

### 1.1 Table of Contents

#### 1.1.1 Installation and Configuration

This guide will take you through the steps to install ORCA on your machine. ORCA is cross platform so you should be able to install it on Linux, OSX, and Windows.

#### Dependencies

- A modern **c++11** compiler (gcc > 4.8 or clang > 3.8)
- **cmake** > 3.1
- **iDynTree** (optional, shipped)
- **qpOASES** 3 (optional, shipped)
- **Eigen** 3 (optional, shipped)
- **Gazebo** 8 (optional)

ORCA is self contained! That means that it ships with both **iDynTree** and **qpOASES** inside the project, allowing for fast installations and easy integration on other platforms. Therefore you can start by simply building ORCA from source and it will include the necessary dependencies so you can get up and running.

Always keep in mind that it's better to install the dependencies separately if you plan to use **iDynTree** or **qpOASES** in other projects. For now only **iDynTree** headers appear in public headers, but will be removed eventually to ease the distribution of this library.

If you want to install the dependencies separately please read the following section: *Installing the dependencies*. Otherwise, if you just want to get coding, then jump ahead to *Installing ORCA*.

---

**Note:** You can almost always avoid calling sudo, by calling cmake .. -DCMAKE\_INSTALL\_PREFIX=/some/dir and exporting the CMAKE\_PREFIX\_PATH variable: export CMAKE\_PREFIX\_PATH=\$CMAKE\_PREFIX\_PATH:/some/dir.

---

### Installing the dependencies

This installation requires you to build the dependencies separately, but will give you better control over versioning and getting the latest features and bug fixes.

#### Eigen

```
wget http://bitbucket.org/eigen/eigen/get/3.3.4.tar.bz2
tar xjvf 3.3.4.tar.bz2
cd eigen-eigen-dc6cfdf9bcec
mkdir build ; cd build
cmake --build .
sudo cmake --build . --target install
```

#### qpOASES

```
wget https://www.coin-or.org/download/source/qpOASES/qpOASES-3.2.1.zip
unzip qpOASES-3.2.1.zip
cd qpOASES-3.2.1
mkdir build ; cd build
cmake .. -DCMAKE_CXX_FLAGS="-fPIC" -DCMAKE_BUILD_TYPE=Release
cmake --build .
sudo cmake --build . --target install
```

#### iDynTree

```
git clone https://github.com/robotology/idyntree
cd idyntree
mkdir build ; cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
cmake --build .
sudo cmake --build . --target install
```

#### Gazebo

Examples are built with Gazebo 8. They can be adapted of course to be backwards compatible.

```
curl -ssl http://get.gazebosim.org | sh
```

## Installing ORCA

Whether or not you have installed the dependencies separately, you are now ready to clone, build and install ORCA. Hooray.

```
git clone https://github.com/syroco/orca
cd orca
mkdir build ; cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
cmake --build .
sudo cmake --build . --target install
```

## Testing your installation

Assuming you followed the directions to the letter and encountered no compiler errors along the way, then you are ready to get started with ORCA. Before moving on to the *Examples*, let's first test the installation.

To do so simply run the following command:

```
orca_install_test
```

## What's next?

Check out [Where to go from here?](#) for more info.

### 1.1.2 Where to go from here?

#### Check out the examples

A number of examples have been included in the source code to help you better understand how ORCA works and how you can use it. The examples are grouped based on the concepts they demonstrate. We also provide some examples for using 3rd party libraries together with ORCA.

#### Want to use ORCA in you project?

Check out the [Using ORCA in your projects](#) page for information on how to include the ORCA library into your next control project.

#### Check out the API Documentation

You can find the Doxygen generated API documentation at the following link: [API Documentation](#). This will help you navigate the ORCA API for your projects.

#### ROS or OROCOS user?

We have written ROS and OROCOS wrappers for the ORCA library and done most of the heavy lifting so you can get started using the controller right away. To learn more about these projects please check out their respective pages:

ORCA\_ROS: [https://github.com/syroco/orca\\_ros](https://github.com/syroco/orca_ros)



RTT\_ORCA: [https://github.com/syroco/rtt\\_orca](https://github.com/syroco/rtt_orca) (Compatible with ORCA < version 2.0.0)

### 1.1.3 API Reference

All of the API documentation is autogenerated using Doxygen. Click the link below to be redirected.

#### API Documentation

### 1.1.4 Building the documentation

The ORCA documentation is composed of two parts. The **user's manual** (what you are currently reading) and the **API Reference**. Since ORCA is written entirely in c++ the API documentation is generated with Doxygen. The manual, on the otherhand, is generated with python Sphinx... because frankly it is prettier.

Obviously, you can always visit the url: insert\_url\_here

to read the documentation online, but you can also generate it locally easily thanks to the magical powers of python.

#### How to build

First we need to install some dependencies for python and of course doxygen.

#### Python dependencies

```
pip3 install -U --user pip sphinx sphinx-autobuild recommonmark sphinx_rtd_theme
```

or if using Python 2.x

```
pip2 install -U --user pip sphinx sphinx-autobuild recommonmark sphinx_rtd_theme
```

#### Doxygen

You can always install Doxygen from source by following:

```
git clone https://github.com/doxygen/doxygen.git
cd doxygen
mkdir build
cd build
cmake -G "Unix Makefiles" ..
make
sudo make install
```

but we would recommend installing the binaries.

**Linux:**

```
sudo apt install doxygen
```

**OSX:**

```
brew install doxygen
```

**Windows:**

Download the executable file here: <http://www.stack.nl/~dimitri/doxygen/download.html> and follow the install wizard.

**Building the docs with Sphinx**

```
cd [orca_root]
cd docs/
make html
```

[orca\_root] is the path to wherever you cloned the repo i.e. /home/\$USER/orca/.

**How to browse**

Since Sphinx builds static websites you can simply find the file docs/build/html/index.html and open it in a browser.

If you prefer to be a fancy-pants then you can launch a local web server by navigating to docs/ and running:

```
make livehtml
```

This method has the advantage of automatically refreshing when you make changes to the .rst files. You can browse the site at: <http://127.0.0.1:8000>.

**1.1.5 Using ORCA in your projects**

If you want to use ORCA in your project you can either use pure CMake or catkin.

**CMake**

```
# You need at least version 3.1 to use the modern CMake targets.
cmake_minimum_required(VERSION 3.1.0)

# Your project's name
project(my_super_orca_project)

# Tell CMake to find ORCA
find_package(orca REQUIRED)
```

(continues on next page)

(continued from previous page)

```
# Add your executable(s) and/or library(ies) and their corresponding source files.
add_executable(${PROJECT_NAME} my_super_orca_project.cc)

# Point CMake to the ORCA targets.
target_link_libraries(${PROJECT_NAME} orca::orca)
```

### catkin

---

**Note:** As of now, catkin does not support modern cmake targets and so you have some superfluous cmake steps to do when working with catkin workspaces.

---

```
# You need at least version 2.8.3 to use the modern CMake targets.
cmake_minimum_required(VERSION 2.8.3)

# Your project's name
project(my_super_orca_catkin_project)

# Tell CMake to find ORCA
find_package(orca REQUIRED)

# Tell catkin to find ORCA
find_package(catkin REQUIRED COMPONENTS orca)

# Include the catkin headers
include_directories(${catkin_INCLUDE_DIRS})

# Add your executable(s) and/or library(ies) and their corresponding source files.
add_executable(${PROJECT_NAME} my_super_orca_catkin_project.cc)

# Point CMake to the catkin and ORCA targets.
target_link_libraries(${PROJECT_NAME} ${catkin_LIBRARIES} orca::orca)
```

## 1.1.6 Basic

### Simple controller

---

**Note:** The source code for this example can be found in [orca\_root]/examples/basic/01-simple\_controller.cc, or alternatively on github at: [https://github.com/syroco/orca/blob/dev/examples/basic/01-simple\\_controller.cc](https://github.com/syroco/orca/blob/dev/examples/basic/01-simple_controller.cc)

---

```
1 #include <orca/orca.h>
2 using namespace orca::all;
3
4 int main(int argc, char const *argv[])
5 {
6     // Get the urdf file from the command line
7     if(argc < 2)
8     {
```

(continues on next page)

(continued from previous page)

```

9      std::cerr << "Usage : " << argv[0] << " /path/to/robot-urdf.urdf (optionally -  

10     -l debug/info/warning/error)" << "\n";
11     return -1;
12 }
13 std::string urdf_url(argv[1]);
14
15 // Parse logger level as --log_level (or -l) debug/warning etc
16 orca::utils::Logger::parseArgv(argc, argv);
17
18 // Create the kinematic model that is shared by everybody. Here you can pass a_
19 //robot name
20 auto robot = std::make_shared<RobotDynTree>();
21
22 // If you don't pass a robot name, it is extracted from the urdf
23 robot->loadModelFromFile(urdf_url);
24
25 // All the transformations (end effector pose for example) will be expressed wrt_
26 //this base frame
27 robot->setBaseFrame("base_link");
28
29 // Sets the world gravity (Optional)
30 robot->setGravity(Eigen::Vector3d(0,0,-9.81));
31
32 // This is an helper function to store the whole state of the robot as eigen_
33 //vectors/matrices. This class is totally optional, it is just meant to keep_
34 //consistency for the sizes of all the vectors/matrices. You can use it to fill data_
35 //from either real robot and simulated robot.
36 RobotState eigState;
37
38 // resize all the vectors/matrices to match the robot configuration
39 eigState.resize(robot->getNrOfDegreesOfFreedom());
40
41 // Set the initial state to zero (arbitrary). @note: here we only set q, qdot_
42 //because this example asserts we have a fixed base robot
43 eigState.jointPos.setZero();
44 eigState.jointVel.setZero();
45
46 // Set the first state to the robot
47 robot->setRobotState(eigState.jointPos,eigState.jointVel);
48 // Now is the robot is considered 'initialized'
49
50
51 // Instanciate an ORCA Controller
52 orca::optim::Controller controller(
53     "controller"
54     ,robot
55     ,orca::optim::ResolutionStrategy::OneLevelWeighted
56     ,QPSSolver::qpOASES
57 );
58 // Other ResolutionStrategy options: MultiLevelWeighted, Generalized
59
60 // Cartesian Task
61 auto cart_task = std::make_shared<CartesianTask>("CartTask-EE");
62 // Add the task to the controller to initialize it.
63 controller.addTask(cart_task);
64 // Set the frame you want to control. Here we want to control the link_7.
65 cart_task->setControlFrame("link_7"); //

```

(continues on next page)

(continued from previous page)

```

59      // Set the pose desired for the link_7
60      Eigen::Affine3d cart_pos_ref;
61
62
63      // Setting the translational components.
64      cart_pos_ref.translation() = Eigen::Vector3d(1., 0.75, 0.5); // x,y,z in meters
65
66
67
68      // Rotation is done with a Matrix3x3 and it can be initialized in a few ways. ↴
69      // Note that each of these methods produce equivalent Rotation matrices in this case.
70
71      // Example 1 : create a quaternion from Euler angles ZYZ convention
72      Eigen::Quaterniond quat;
73      quat = Eigen::AngleAxisd(0, Eigen::Vector3d::UnitZ())
74          * Eigen::AngleAxisd(0, Eigen::Vector3d::UnitY())
75          * Eigen::AngleAxisd(0, Eigen::Vector3d::UnitZ());
76      cart_pos_ref.linear() = quat.toRotationMatrix();
77
78      // Example 2 : create a quaternion from RPY convention
79      cart_pos_ref.linear() = quatFromRPY(0, 0, 0).toRotationMatrix();
80
81      // Example 3 : create a quaternion from Kuka Convention
82      cart_pos_ref.linear() = quatFromKukaConvention(0, 0, 0).toRotationMatrix();
83
84      // Example 4 : use an Identity quaternion
85      cart_pos_ref.linear() = Eigen::Quaterniond::Identity().toRotationMatrix();
86
87
88      // Set the desired cartesian velocity and acceleration to zero
89      Vector6d cart_vel_ref = Vector6d::Zero();
90      Vector6d cart_acc_ref = Vector6d::Zero();
91
92      // Now set the servoing PID
93      Vector6d P;
94      P << 1000, 1000, 1000, 10, 10, 10;
95      cart_task->servoController()->pid()->setProportionalGain(P);
96      Vector6d D;
97      D << 100, 100, 100, 1, 1, 1;
98      cart_task->servoController()->pid()->setDerivativeGain(D);
99
100
101      // The desired values are set on the servo controller. Because cart_task->
102      // setDesired expects a cartesian acceleration. Which is computed automatically by the ↴
103      // servo controller
104      cart_task->servoController()->setDesired(cart_pos_ref.matrix(), cart_vel_ref, cart_
105      // acc_ref);
106
107      // Get the number of actuated joints
108      const int ndof = robot->getNrOfDegreesOfFreedom();
109
110      // Joint torque limit is usually given by the robot manufacturer
111      auto jnt_trq_cstr = std::make_shared<JointTorqueLimitConstraint>("JointTorqueLimit
112      ");
113
114      // Add the constraint to the controller to initialize - it is not read from the ↴
115      // URDF for now.

```

(continues on next page)

(continued from previous page)

```

110 controller.addConstraint(jnt_trq_cstr);
111 Eigen::VectorXd jntTrqMax(ndof);
112 jntTrqMax.setConstant(200.0);
113 jnt_trq_cstr->setLimits(-jntTrqMax, jntTrqMax);
114
115 // Joint position limits are automatically extracted from the URDF model. Note_
116 // that you can set them if you want. by simply doing jnt_pos_cstr->
117 // setLimits(jntPosMin, jntPosMax).
118 auto jnt_pos_cstr = std::make_shared<JointPositionLimitConstraint>(
119 // "JointPositionLimit");
120
121 // Add the constraint to the controller to initialize
122 controller.addConstraint(jnt_pos_cstr);
123
124 // Joint velocity limits are usually given by the robot manufacturer
125 auto jnt_vel_cstr = std::make_shared<JointVelocityLimitConstraint>(
126 // "JointVelocityLimit");
127
128 // Add the constraint to the controller to initialize - it is not read from the_
129 // URDF for now.
130 controller.addConstraint(jnt_vel_cstr);
131 Eigen::VectorXd jntVelMax(ndof);
132 jntVelMax.setConstant(2.0);
133 jnt_vel_cstr->setLimits(-jntVelMax, jntVelMax);
134
135
136 double dt = 0.001;
137 double current_time = 0;
138
139 controller.activateTasksAndConstraints();
140
141 // If your robot's low level controller takes into account the gravity and_
142 // coriolis torques already (Like with KUKA LWR) then you can tell the controller to_
143 // remove these components from the torques computed by the solver. Setting them to_
144 // false keeps the components in the solution (this is the default behavior).
145 controller.removeGravityTorquesFromSolution(true);
146 controller.removeCoriolisTorquesFromSolution(true);
147
148 // Now you can run the control loop
149 for (; current_time < 2.0; current_time +=dt)
150 {
151     // Here you can get the data from your REAL robot (API is robot-specific)
152     // Something like :
153     // eigState.jointPos = myRealRobot.getJointPositions();
154     // eigState.jointVel = myRealRobot.getJointVelocities();
155
156     // Now update the internal kinematic model with data from the REAL robot
157     robot->setRobotState(eigState.jointPos, eigState.jointVel);
158
159     // Step the controller + solve the internal optimal problem
160     controller.update(current_time, dt);
161
162     // Do what you want with the solution
163     if(controller.solutionFound())
164     {
165         // The whole optimal solution [AccFb, Acc, Tfb, T, eWrenches]

```

(continues on next page)

(continued from previous page)

```
159     const Eigen::VectorXd& full_solution = controller.getSolution();
160     // The optimal joint torque command
161     const Eigen::VectorXd& trq_cmd = controller.getJointTorqueCommand();
162     // The optimal joint acceleration command
163     const Eigen::VectorXd& trq_acc = controller.getJointAccelerationCommand();
164
165     // Send torques to the REAL robot (API is robot-specific)
166     //real_tobot->set_joint_torques(trq_cmd);
167 }
168 else
169 {
170     // WARNING : Optimal solution is NOT found
171     // Switching to a fallback strategy
172     // Typical are :
173     // - Stop the robot (robot-specific method)
174     // - Compute KKT Solution and send to the robot (dangerous)
175     // - PID around the current position (dangerous)
176
177     // trq = controller.computeKTTorques();
178     // Send torques to the REAL robot (API is robot-specific)
179     // real_tobot->set_joint_torques(trq_cmd);
180 }
181
182
183 // Print the last computed solution (just for fun)
184 const Eigen::VectorXd& full_solution = controller.getSolution();
185 const Eigen::VectorXd& trq_cmd = controller.getJointTorqueCommand();
186 const Eigen::VectorXd& trq_acc = controller.getJointAccelerationCommand();
187 LOG_INFO << "Full solution : " << full_solution.transpose();
188 LOG_INFO << "Joint Acceleration command : " << trq_acc.transpose();
189 LOG_INFO << "Joint Torque command : " << trq_cmd.transpose();
190
191 // At some point you want to close the controller nicely
192 controller.deactivateTasksAndConstraints();
193
194
195 // Let all the tasks ramp down to zero
196 while (!controller.tasksAndConstraintsDeactivated())
197 {
198     current_time += dt;
199     controller.update(current_time,dt);
200 }
201
202 // All objets will be destroyed here
203 return 0;
204 }
```

---

## Simulating the controller performance

---

**Note:** The source code for this example can be found in [orca\_root]/examples/basic/02-simulating\_results.cc, or alternatively on github at: [https://github.com/syroco/orca/blob/dev/examples/basic/02-simulating\\_results.cc](https://github.com/syroco/orca/blob/dev/examples/basic/02-simulating_results.cc)

---

```

1 #include <orca/orca.h>
2 using namespace orca::all;
3
4
5
6 int main(int argc, char const *argv[])
7 {
8     if(argc < 2)
9     {
10         std::cerr << "Usage : " << argv[0] << " /path/to/robot-urdf.urdf (optionally -"
11             "l debug/info/warning/error)" << "\n";
12         return -1;
13     }
14     std::string urdf_url(argv[1]);
15
16     orca::utils::Logger::parseArgv(argc, argv);
17
18     auto robot = std::make_shared<RobotDynTree>();
19     robot->loadModelFromFile(urdf_url);
20     robot->setBaseFrame("base_link");
21     robot->setGravity(Eigen::Vector3d(0,0,-9.81));
22     RobotState eigState;
23     eigState.resize(robot->getNrOfDegreesOfFreedom());
24     eigState.jointPos.setZero();
25     eigState.jointVel.setZero();
26     robot->setRobotState(eigState.jointPos,eigState.jointVel);
27
28     orca::optim::Controller controller(
29         "controller"
30         ,robot
31         ,orca::optim::ResolutionStrategy::OneLevelWeighted
32         ,QPSSolver::qpOASES
33     );
34
35     auto cart_task = std::make_shared<CartesianTask>("CartTask-EE");
36     controller.addTask(cart_task);
37     cart_task->setControlFrame("link_7"); //
38     Eigen::Affine3d cart_pos_ref;
39     cart_pos_ref.translation() = Eigen::Vector3d(1.,0.75,0.5); // x,y,z in meters
40     cart_pos_ref.linear() = Eigen::Quaterniond::Identity().toRotationMatrix();
41     Vector6d cart_vel_ref = Vector6d::Zero();
42     Vector6d cart_acc_ref = Vector6d::Zero();
43
44     Vector6d P;
45     P << 1000, 1000, 1000, 10, 10, 10;
46     cart_task->servoController()->pid()->setProportionalGain(P);
47     Vector6d D;
48     D << 100, 100, 100, 1, 1, 1;
49     cart_task->servoController()->pid()->setDerivativeGain(D);
50
51     cart_task->servoController()->setDesired(cart_pos_ref.matrix(),cart_vel_ref,cart_
52         &acc_ref);
53
54     const int ndof = robot->getNrOfDegreesOfFreedom();
55
56     auto jnt_trq_cstr = std::make_shared<JointTorqueLimitConstraint>("JointTorqueLimit"
57         &);

```

(continues on next page)

(continued from previous page)

```

55 controller.addConstraint(jnt_trq_cstr);
56 Eigen::VectorXd jntTrqMax(ndof);
57 jntTrqMax.setConstant(200.0);
58 jnt_trq_cstr->setLimits(-jntTrqMax, jntTrqMax);

59
60 auto jnt_pos_cstr = std::make_shared<JointPositionLimitConstraint>(
61     "JointPositionLimit");
62 controller.addConstraint(jnt_pos_cstr);

63 auto jnt_vel_cstr = std::make_shared<JointVelocityLimitConstraint>(
64     "JointVelocityLimit");
65 controller.addConstraint(jnt_vel_cstr);
66 Eigen::VectorXd jntVelMax(ndof);
67 jntVelMax.setConstant(2.0);
68 jnt_vel_cstr->setLimits(-jntVelMax, jntVelMax);

69
70 controller.activateTasksAndConstraints();
71 // for each task, it calls task->activate(), that can call onActivationCallback()
72 // if it is set.
73 // To set it :
74 // task->setOnActivationCallback([&] ())
75 // {
76 //     // Do some initialisation here
77 // };
78 // Note : you need to set it BEFORE calling
79 // controller.activateTasksAndConstraints();

80
81
82
83
84 double dt = 0.001;
85 double current_time = 0.0;
86 Eigen::VectorXd trq_cmd(ndof);
87 Eigen::VectorXd acc_new(ndof);

88
89 controller.update(current_time, dt);

90
91 std::cout << "\n\n\n" << '\n';
92 std::cout << "===== " << '\n';
93 std::cout << "Initial State:\n" << cart_task->servoController()->
94     getCurrentCartesianPose() << '\n';
95 std::cout << "Desired State:\n" << cart_pos_ref.matrix() << '\n';
96 std::cout << "===== " << '\n';
97 std::cout << "\n\n\n" << '\n';
98 std::cout << "Beginning Simulation..." << '\n';

99 for (; current_time < 2.0; current_time +=dt)
100 {
101
102     robot->setRobotState(eigState.jointPos,eigState.jointVel);

103     // if(current_time % 0.1 == 0.0)
104     // {
105     // 
106     // }

```

(continues on next page)

(continued from previous page)

```

108     std::cout << "Task position at t = " << current_time << "\t---\t" << cart_
109     ↪task->servoController()->getCurrentCartesianPose().block(0,3,3,1).transpose() << '\n'
110     ↪';
111
112     controller.update(current_time, dt);
113
114     if(controller.solutionFound())
115     {
116         trq_cmd = controller.getJointTorqueCommand();
117     }
118     else
119     {
120         std::cout << "[warning] Didn't find a solution, using last valid solution.
121         " << '\n';
122     }
123
124     acc_new = robot->getMassMatrix().ldlt().solve(trq_cmd - robot->
125     ↪getJointGravityAndCoriolisTorques());
126
127     eigState.jointPos += eigState.jointVel * dt + ((acc_new*dt*dt)/2);
128     eigState.jointVel += acc_new * dt;
129 }
130 std::cout << "Simulation finished." << '\n';
131 std::cout << "\n\n\n" << '\n';
132 std::cout << "===== " << '\n';
133 std::cout << "Final State:\n" << cart_task->servoController()->
134     ↪getCurrentCartesianPose() << '\n';
135     // std::cout << "Position error:\n" << cart_task->servoController()->
136     ↪getCurrentCartesianPose(). - cart_pos_ref.translation() << '\n';
137
138
139
140
141
142
143
144
145     // All objects will be destroyed here
146
147     return 0;
148 }
```

## 1.1.7 Intermediate

## An introduction to the ORCA callback system

**Note:** The source code for this example can be found in `[orca_root]/examples/intermediate/02-using_callbacks.cc`, or alternatively on github at: [https://github.com/syroco/orca/blob/dev/examples/intermediate/02-using\\_callbacks.cc](https://github.com/syroco/orca/blob/dev/examples/intermediate/02-using_callbacks.cc)

```
1 #include <orca/orca.h>
2 #include <chrono>
3 using namespace orca::all;
4
5 class TaskMonitor {
6 private:
7     bool is_activated_ = false;
```

(continues on next page)

(continued from previous page)

```

8     bool is_deactivated_ = false;
9
10
11    public:
12        TaskMonitor ()
13        {
14            std::cout << "TaskMonitor class constructed." << '\n';
15        }
16        bool isActivated(){return is_activated_;}
17        bool isDeactivated(){return is_deactivated_;}
18
19        void onActivation()
20        {
21            std::cout << "[TaskMonitor] Called 'onActivation' callback." << '\n';
22        }
23
24        void onActivated()
25        {
26            std::cout << "[TaskMonitor] Called 'onActivated' callback." << '\n';
27            is_activated_ = true;
28        }
29
30        void onUpdateEnd(double current_time, double dt)
31        {
32            std::cout << "[TaskMonitor] Called 'onUpdateBegin' callback." << '\n';
33            std::cout << "    >> current time: " << current_time << '\n';
34            std::cout << "    >> dt: " << dt << '\n';
35        }
36
37        void onUpdateBegin(double current_time, double dt)
38        {
39            std::cout << "[TaskMonitor] Called 'onUpdateEnd' callback." << '\n';
40            std::cout << "    >> current time: " << current_time << '\n';
41            std::cout << "    >> dt: " << dt << '\n';
42        }
43        void onDeactivation()
44        {
45            std::cout << "[TaskMonitor] Called 'onDeactivation' callback." << '\n';
46        }
47
48        void onDeactivated()
49        {
50            std::cout << "[TaskMonitor] Called 'onDeactivated' callback." << '\n';
51            is_deactivated_ = true;
52        }
53    };
54
55
56
57
58    int main(int argc, char const *argv[])
59    {
60        if(argc < 2)
61        {
62            std::cerr << "Usage : " << argv[0] << " /path/to/robot-urdf.urdf (optionally -"
63            << "l debug/info/warning/error)" << "\n";
64            return -1;

```

(continues on next page)

(continued from previous page)

```

64 }
65 std::string urdf_url(argv[1]);
66
67 orca::utils::Logger::parseArgv(argc, argv);
68
69 auto robot = std::make_shared<RobotDynTree>();
70 robot->loadModelFromFile(urdf_url);
71 robot->setBaseFrame("base_link");
72 robot->setGravity(Eigen::Vector3d(0,0,-9.81));
73 RobotState eigState;
74 eigState.resize(robot->getNrOfDegreesOfFreedom());
75 eigState.jointPos.setZero();
76 eigState.jointVel.setZero();
77 robot->setRobotState(eigState.jointPos,eigState.jointVel);
78
79 orca::optim::Controller controller(
80     "controller"
81     ,robot
82     ,orca::optim::ResolutionStrategy::OneLevelWeighted
83     ,QPSSolver::qpOASES
84 );
85
86 auto cart_task = std::make_shared<CartesianTask>("CartTask-EE");
87 controller.addTask(cart_task);
88 cart_task->setControlFrame("link_7"); //
89 Eigen::Affine3d cart_pos_ref;
90 cart_pos_ref.translation() = Eigen::Vector3d(1.,0.75,0.5); // x,y,z in meters
91 cart_pos_ref.linear() = Eigen::Quaterniond::Identity().toRotationMatrix();
92 Vector6d cart_vel_ref = Vector6d::Zero();
93 Vector6d cart_acc_ref = Vector6d::Zero();
94
95 Vector6d P;
96 P << 1000, 1000, 1000, 10, 10, 10;
97 cart_task->servoController()->pid()->setProportionalGain(P);
98 Vector6d D;
99 D << 100, 100, 100, 1, 1, 1;
100 cart_task->servoController()->pid()->setDerivativeGain(D);
101
102 cart_task->servoController()->setDesired(cart_pos_ref.matrix(),cart_vel_ref,cart_
103 ↵acc_ref);
104
105 const int ndof = robot->getNrOfDegreesOfFreedom();
106
107 auto jnt_trq_cstr = std::make_shared<JointTorqueLimitConstraint>("JointTorqueLimit
108 ↵");
109 controller.addConstraint(jnt_trq_cstr);
110 Eigen::VectorXd jntTrqMax(ndof);
111 jntTrqMax.setConstant(200.0);
112 jnt_trq_cstr->setLimits(-jntTrqMax,jntTrqMax);
113
114 auto jnt_pos_cstr = std::make_shared<JointPositionLimitConstraint>(
115 ↵"JointPositionLimit");
116 controller.addConstraint(jnt_pos_cstr);
117
118 auto jnt_vel_cstr = std::make_shared<JointVelocityLimitConstraint>(
119 ↵"JointVelocityLimit");
120 controller.addConstraint(jnt_vel_cstr);

```

(continues on next page)

(continued from previous page)

```

117     Eigen::VectorXd jntVelMax(ndof);
118     jntVelMax.setConstant(2.0);
119     jnt_vel_cstr->setLimits(-jntVelMax, jntVelMax);
120
121     double dt = 0.1;
122     double current_time = 0.0;
123     int delay_ms = 500;
124
125     // The good stuff...
126
127     auto task_monitor = std::make_shared<TaskMonitor>();
128
129     cart_task->onActivationCallback(std::bind(&TaskMonitor::onActivation, task_
130     ↵monitor));
130     cart_task->onActivatedCallback(std::bind(&TaskMonitor::onActivated, task_
131     ↵monitor));
131     cart_task->onComputeBeginCallback(std::bind(&TaskMonitor::onUpdateBegin, task_
132     ↵monitor, std::placeholders::_1, std::placeholders::_2));
132     cart_task->onComputeEndCallback(std::bind(&TaskMonitor::onUpdateEnd, task_monitor,
133     ↵ std::placeholders::_1, std::placeholders::_2));
133     cart_task->onDeactivationCallback(std::bind(&TaskMonitor::onDeactivation, task_
134     ↵monitor));
134     cart_task->onDeactivatedCallback(std::bind(&TaskMonitor::onDeactivated, task_
135     ↵monitor));
135
136     std::cout << "[main] Activating tasks and constraints." << '\n';
137     controller.activateTasksAndConstraints();
138     std::this_thread::sleep_for(std::chrono::milliseconds(delay_ms));
139
140     std::cout << "[main] Starting 'RUN' while loop." << '\n';
141     while(!task_monitor->isActivated()) // Run 10 times.
142     {
143         std::cout << "[main] 'RUN' while loop. Current time: " << current_time << '\n
144         ↵';
144         controller.update(current_time, dt);
145         current_time +=dt;
146         std::this_thread::sleep_for(std::chrono::milliseconds(delay_ms));
147     }
148     std::cout << "[main] Exiting 'RUN' while loop." << '\n';
149
150     std::cout << "-----\n";
151
152     std::cout << "[main] Deactivating tasks and constraints." << '\n';
153     controller.deactivateTasksAndConstraints();
154     std::this_thread::sleep_for(std::chrono::milliseconds(delay_ms));
155
156     std::cout << "[main] Starting 'DEACTIVATION' while loop." << '\n';
157
158     while(!task_monitor->isDeactivated())
159     {
160         std::cout << "[main] 'DEACTIVATION' while loop. Current time: " << current_
161         ↵time << '\n';
161         controller.update(current_time, dt);
162         current_time += dt;
163         std::this_thread::sleep_for(std::chrono::milliseconds(delay_ms));
164     }
165     std::cout << "[main] Exiting 'DEACTIVATION' while loop." << '\n';

```

(continues on next page)

(continued from previous page)

```

166
167
168     std::cout << "[main] Exiting main()." << '\n';
169     return 0;
170 }
```

## Using lambda functions in the callbacks

**Note:** The source code for this example can be found in [orca\_root]/examples/intermediate/02-using\_lambda\_callbacks.cc, or alternatively on github at: [https://github.com/syroco/orca/blob/dev/examples/intermediate/02-using\\_lambda\\_callbacks.cc](https://github.com/syroco/orca/blob/dev/examples/intermediate/02-using_lambda_callbacks.cc)

## 1.1.8 Gazebo

### Simulating a single robot

**Note:** The source code for this example can be found in [orca\_root]/examples/gazebo/01-single\_robot.cc, or alternatively on github at: [https://github.com/syroco/orca/blob/dev/examples/gazebo/01-single\\_robot.cc](https://github.com/syroco/orca/blob/dev/examples/gazebo/01-single_robot.cc)

```

1 #include <orca/gazebo/GazeboServer.h>
2 #include <orca/gazebo/GazeboModel.h>
3
4 using namespace orca::gazebo;
5
6 int main(int argc, char** argv)
7 {
8     // Get the urdf file from the command line
9     if(argc < 2)
10    {
11        std::cerr << "Usage : " << argv[0] << " /path/to/robot-urdf.urdf" << "\n";
12        return -1;
13    }
14    std::string urdf_url(argv[1]);
15
16    // Instanciate the gazebo server with de dedfault empty world
17    // This is equivalent to GazeboServer gz("worlds/empty.world")
18    GazeboServer s;
19    // Insert a model onto the server and create the GazeboModel from the return value
20    // You can also set the initial pose, and override the name in the URDF
21    auto m = GazeboModel(s.insertModelFromURDFFile(urdf_url));
22
23    // This is how you can get the full state of the robot
24    std::cout << "Model '" << m.getName() << "' State :\n" << '\n';
25    std::cout << "- Gravity " << m.getGravity().transpose() << '\n';
26    std::cout << "- Base velocity\n" << m.getBaseVelocity().transpose() << '\n';
27    std::cout << "- Tworld->base\n" << m.getWorldToBaseTransform().
matrix() << '\n';
```

(continues on next page)

(continued from previous page)

```

28     std::cout << "-- Joint positions "           << m.getJointPositions().transpose() 
29     << '\n';
30     std::cout << "-- Joint velocities "         << m.getJointVelocities().transpose() 
31     << '\n';
32     std::cout << "-- Joint external torques "   << m.getJointExternalTorques(). 
33 transpose() << '\n';
34     std::cout << "-- Joint measured torques "   << m.getJointMeasuredTorques(). 
35 transpose() << '\n';
36
37     // You can optionally register a callback that will be called
38     // after every WorldUpdateEnd, so the internal gazebo model is updated
39     // and you can get the full state (q,qdot,Tworld->base, etc)
40     m.setCallback([&](uint32_t n_iter,double current_time,double dt)
41     {
42         std::cout << "[" << m.getName() << "]" << '\n'
43         << "-- iteration " << n_iter << '\n'
44         << "-- current time " << current_time << '\n'
45         << "-- dt " << dt << '\n';
46         // Example : get the minimal state
47         const Eigen::VectorXd q = m.getJointPositions();
48         const Eigen::VectorXd qdot = m.getJointVelocities();
49
50         std::cout << "ExtTrq " << m.getJointExternalTorques().transpose() << '\n';
51         std::cout << "MeaTrq " << m.getJointMeasuredTorques().transpose() << '\n';
52     });
53
54     // Run the main simulation loop.
55     // This is a blocking call that runs the simulation steps
56     // It can be stopped by CTRL+C
57     // You can optionally add a callback that happens after WorldUpdateEnd
58     s.run();
59     return 0;
60 }
```

## Simulating multiple robots

**Note:** The source code for this example can be found in [orca\_root]/examples/gazebo/02-multi\_robot.cc, or alternatively on github at: [https://github.com/syroco/orca/blob/dev/examples/gazebo/02-multi\\_robot.cc](https://github.com/syroco/orca/blob/dev/examples/gazebo/02-multi_robot.cc)

```

1 #include <orca/gazebo/GazeboServer.h>
2 #include <orca/gazebo/GazeboModel.h>
3
4 using namespace orca::gazebo;
5 using namespace Eigen;
6
7 int main(int argc, char** argv)
8 {
9     // Get the urdf file from the command line
10    if(argc < 2)
11    {
12        std::cerr << "Usage : " << argv[0] << " /path/to/robot-urdf.urdf" << "\n";
13        return -1;
14 }
```

(continues on next page)

(continued from previous page)

```

14 }
15 std::string urdf_url(argv[1]);
16
17 // Instanciate the gazebo server with de default empty world
18 // This is equivalent to GazeboServer gz("worlds/empty.world")
19 GazeboServer gz_server;
20
21 // Insert a model onto the server and create the GazeboModel from the return value
22 // You can also set the initial pose, and override the name in the URDF
23 auto gz_model_one = GazeboModel(gz_server.insertModelFromURDFFile(urdf_url
24     ,Vector3d(-2,0,0)
25     ,quatFromRPY(0,0,0)
26     ,"one"));
27
28 // Insert a second model with a different pose and a different name
29 auto gz_model_two = GazeboModel(gz_server.insertModelFromURDFFile(urdf_url
30     ,Vector3d(2,0,0)
31     ,quatFromRPY(0,0,0)
32     ,"two"));
33
34 // You can optionally register a callback for each GazeboModel so you can do
35 // individual updates on it
36 // The function is called after every WorldUpdateEnd, so the internal gazebo_
37 // model is updated
38 // and you can get the full state (q,qdot,Tworld->base, etc)
39 gz_model_two.setCallback([&](uint32_t n_iter,double current_time,double dt)
40 {
41     std::cout << "gz_model_two \" " << gz_model_two.getName() << "\" callback " <<
42     '\n'
43         << "- iteration " << n_iter << '\n'
44         << "- current time " << current_time << '\n'
45         << "- dt " << dt << '\n';
46     // Example : get the joint positions
47     // gz_model_two.getJointPositions()
48 });
49
50 // Run the main simulation loop.
51 // This is a blocking call that runs the simulation steps
52 // It can be stopped by CTRL+C
53 // You can optionally add a callback that happens after WorldUpdateEnd
54 gz_server.run([&](uint32_t n_iter,double current_time,double dt)
55 {
56     std::cout << "GazeboServer callback " << '\n'
57         << "- iteration " << n_iter << '\n'
58         << "- current time " << current_time << '\n'
59         << "- dt " << dt << '\n';
60 });
61 return 0;
62 }
```

## Set robot state

**Note:** The source code for this example can be found in [orca\_root]/examples/gazebo/03-set\_robot\_state.cc, or alternatively on github at: [https://github.com/syroco/orca/blob/dev/examples/03-set\\_robot\\_state.cc](https://github.com/syroco/orca/blob/dev/examples/03-set_robot_state.cc)

### gazebo/03-set\_robot\_state.cc

---

```
1 #include <orca/orca.h>
2 #include <orca/gazebo/GazeboServer.h>
3 #include <orca/gazebo/GazeboModel.h>
4
5 using namespace orca::all;
6 using namespace orca::gazebo;
7
8 int main(int argc, char** argv)
9 {
10     // Get the urdf file from the command line
11     if(argc < 2)
12     {
13         std::cerr << "Usage : " << argv[0] << " /path/to/robot-urdf.urdf" << "\n";
14         return -1;
15     }
16     std::string urdf_url(argv[1]);
17
18     // Instanciate the gazebo server with de dedfault empty world
19     GazeboServer gzserver(argc,argv);
20     // This is equivalent to GazeboServer gz("worlds/empty.world")
21     // Insert a model onto the server and create the GazeboModel from the return value
22     // You can also set the initial pose, and override the name in the URDF
23     auto gzrobot = GazeboModel(gzserver.insertModelFromURDFFile(urdf_url));
24
25     // Create an ORCA robot
26     auto robot = std::make_shared<RobotDynTree>();
27     robot->loadModelFromFile(urdf_url);
28     robot->print();
29
30     // Update the robot on at every iteration
31     gzrobot.setCallback([&](uint32_t n_iter, double current_time, double dt)
32     {
33         robot->setRobotState(gzrobot.getWorldToBaseTransform().matrix()
34             ,gzrobot.getJointPositions()
35             ,gzrobot.getBaseVelocity()
36             ,gzrobot.getJointVelocities()
37             ,gzrobot.getGravity()
38         );
39     });
40
41     // Run the main simulation loop.
42     // This is a blocking call that runs the simulation steps
43     // It can be stopped by CTRL+C
44     // You can optionally add a callback that happens after WorldUpdateEnd
45     gzserver.run();
46     return 0;
47 }
```

### Set robot state with gravity compensation

---

**Note:** The source code for this example can be found in [orca\_root]/examples/gazebo/04-set\_robot\_state\_gravity\_compensation.cc, or alternatively on github at: <https://github.com/>

syroco/orca/blob/dev/examples/gazebo/04-set\_robot\_state\_gravity\_compensation.cc

```

1 #include <orca/orca.h>
2 #include <orca/gazebo/GazeboServer.h>
3 #include <orca/gazebo/GazeboModel.h>
4
5 using namespace orca::all;
6 using namespace orca::gazebo;
7
8 int main(int argc, char** argv)
9 {
10     // Get the urdf file from the command line
11     if(argc < 2)
12     {
13         std::cerr << "Usage : " << argv[0] << " /path/to/robot-urdf.urdf" << "\n";
14         return -1;
15     }
16     std::string urdf_url(argv[1]);
17
18     // Instanciate the gazebo server with de dedfault empty world
19     GazeboServer gzserver(argc,argv);
20     // This is equivalent to GazeboServer gz("worlds/empty.world")
21     // Insert a model onto the server and create the GazeboModel from the return value
22     // You can also set the initial pose, and override the name in the URDF
23     auto gzrobot = GazeboModel(gzserver.insertModelFromURDFFile(urdf_url));
24
25     // Create an ORCA robot
26     auto robot_kinematics = std::make_shared<RobotDynTree>();
27     robot_kinematics->loadModelFromFile(urdf_url);
28     robot_kinematics->print();
29
30     // Set the gazebo model init pose
31     // auto joint_names = robot_kinematics->getJointNames();
32     // std::vector<double> init_joint_positions(robot_kinematics->
33     // →getNrOfDegreesOfFreedom(),0);
34
35     // gzrobot.setModelConfiguration(joint_names,init_joint_positions);
36     // or like this
37     // gzrobot.setModelConfiguration({"joint_2","joint_5"},{1.5,0.0});
38
39     // Update the robot on at every iteration
40     gzrobot.setCallback([&](uint32_t n_iter,double current_time,double dt)
41     {
42         robot_kinematics->setRobotState(gzrobot.getWorldToBaseTransform().matrix()
43                                         ,gzrobot.getJointPositions()
44                                         ,gzrobot.getBaseVelocity()
45                                         ,gzrobot.getJointVelocities()
46                                         ,gzrobot.getGravity()
47                                         );
48         gzrobot.setJointGravityTorques(robot_kinematics->getJointGravityTorques());
49     });
50
51     // Run the main simulation loop.
52     // This is a blocking call that runs the simulation steps
53     // It can be stopped by CTRL+C
54     // You can optionally add a callback that happens after WorldUpdateEnd
55     std::cout << "Simulation running... (GUI with 'gzclient')" << "\n";

```

(continues on next page)

(continued from previous page)

```
55     gzserver.run();
56     return 0;
57 }
```

## Using Gazebo to simulate an ORCA controller

---

**Note:** The source code for this example can be found in [orca\_root]/examples/gazebo/05-orca\_gazebo.cc, or alternatively on github at: [https://github.com/syroco/orca/blob/dev/examples/gazebo/05-orca\\_gazebo.cc](https://github.com/syroco/orca/blob/dev/examples/gazebo/05-orca_gazebo.cc)

---

```
1 #include <orca/orca.h>
2 #include <orca/gazebo/GazeboServer.h>
3 #include <orca/gazebo/GazeboModel.h>
4
5 using namespace orca::all;
6 using namespace orca::gazebo;
7
8
9
10 int main(int argc, char const *argv[])
11 {
12     if(argc < 2)
13     {
14         std::cerr << "Usage : " << argv[0] << " /path/to/robot-urdf.urdf (optionally -"
15         "l debug/info/warning/error)" << "\n";
16         return -1;
17     }
18     std::string urdf_url(argv[1]);
19
20     orca::utils::Logger::parseArgv(argc, argv);
21
22     auto robot = std::make_shared<RobotDynTree>();
23     robot->loadModelFromFile(urdf_url);
24     robot->setBaseFrame("base_link");
25     robot->setGravity(Eigen::Vector3d(0, 0, -9.81));
26     RobotState eigState;
27     eigState.resize(robot->getNrOfDegreesOfFreedom());
28     eigState.jointPos.setZero();
29     eigState.jointVel.setZero();
30     robot->setRobotState(eigState.jointPos, eigState.jointVel);
31
32     orca::optim::Controller controller(
33         "controller"
34         , robot
35         , orca::optim::ResolutionStrategy::OneLevelWeighted
36         , QPSolver::qpOASES
37     );
38
39     auto cart_task = std::make_shared<CartesianTask>("CartTask-EE");
40     controller.addTask(cart_task);
41     cart_task->setControlFrame("link_7"); //
42     Eigen::Affine3d cart_pos_ref;
43     cart_pos_ref.translation() = Eigen::Vector3d(0.5, -0.5, 0.8); // x,y,z in meters
```

(continues on next page)

(continued from previous page)

```

43     cart_pos_ref.linear() = Eigen::Quaterniond::Identity().toRotationMatrix();
44     Vector6d cart_vel_ref = Vector6d::Zero();
45     Vector6d cart_acc_ref = Vector6d::Zero();
46
47     Vector6d P;
48     P << 1000, 1000, 1000, 10, 10, 10;
49     cart_task->servoController()->pid()->setProportionalGain(P);
50     Vector6d D;
51     D << 100, 100, 100, 1, 1, 1;
52     cart_task->servoController()->pid()->setDerivativeGain(D);
53     cart_task->servoController()->setDesired(cart_pos_ref.matrix(), cart_vel_ref, cart_
54     ↵acc_ref);
55
56     const int ndof = robot->getNrOfDegreesOfFreedom();
57
58     auto jnt_trq_cstr = std::make_shared<JointTorqueLimitConstraint>("JointTorqueLimit
59     ↵");
60     controller.addConstraint(jnt_trq_cstr);
61     Eigen::VectorXd jntTrqMax(ndof);
62     jntTrqMax.setConstant(200.0);
63     jnt_trq_cstr->setLimits(-jntTrqMax, jntTrqMax);
64
64     auto jnt_pos_cstr = std::make_shared<JointPositionLimitConstraint>(
65     ↵"JointPositionLimit");
66     controller.addConstraint(jnt_pos_cstr);
67
67     auto jnt_vel_cstr = std::make_shared<JointVelocityLimitConstraint>(
68     ↵"JointVelocityLimit");
69     controller.addConstraint(jnt_vel_cstr);
70     Eigen::VectorXd jntVelMax(ndof);
71     jntVelMax.setConstant(2.0);
72     jnt_vel_cstr->setLimits(-jntVelMax, jntVelMax);
73
73     GazeboServer gzserver(argc, argv);
74     auto gzrobot = GazeboModel(gzserver.insertModelFromURDFFile(urdf_url));
75
76     /////////////////////////////////
77     /////////////////////////////////
78     /////////////////////////////////
79     ///////////////////////////////
80
81     bool cart_task_activated = false;
82
83     cart_task->onActivatedCallback([&cart_task_activated](){
84         std::cout << "CartesianTask activated. Removing gravity compensation and_
85     ↵beginning motion." << '\n';
86         cart_task_activated = true;
87     });
88
88     gzrobot.setCallback([](uint32_t n_iter, double current_time, double dt)
89     {
90         robot->setRobotState(gzrobot.getWorldToBaseTransform().matrix()
91             ,gzrobot.getJointPositions()
92             ,gzrobot.getBaseVelocity()
93             ,gzrobot.getJointVelocities()
94             ,gzrobot.getGravity())

```

(continues on next page)

(continued from previous page)

```

95         );
96     // All tasks need the robot to be initialized during the activation phase
97     if(n_iter == 1)
98         controller.activateTasksAndConstraints();
99
100    controller.update(current_time, dt);
101   if (cart_task_activated)
102   {
103       if(controller.solutionFound())
104       {
105           gzrobot.setJointTorqueCommand( controller.getJointTorqueCommand() );
106       }
107       else
108       {
109           gzrobot.setBrakes(true);
110       }
111   }
112   else
113   {
114       gzrobot.setJointGravityTorques(robot->getJointGravityTorques());
115   }
116 };
117
118 std::cout << "Simulation running... (GUI with \'gzclient\') " << "\n";
119 gzserver.run();
120 return 0;
121 }
```

## Minimum jerk Cartesian trajectory following

**Note:** The source code for this example can be found in [orca\_root]/examples/gazebo/06-trajectory\_following.cc, or alternatively on github at: [https://github.com/syroco/orca/blob/dev/examples/gazebo/06-trajectory\\_following.cc](https://github.com/syroco/orca/blob/dev/examples/gazebo/06-trajectory_following.cc)

```

1 #include <orca/orca.h>
2 #include <corca/gazebo/GazeboServer.h>
3 #include <corca/gazebo/GazeboModel.h>
4
5 using namespace orca::all;
6 using namespace orca::gazebo;
7
8 class MinJerkPositionTrajectory {
9 private:
10     Eigen::Vector3d alpha_, sp_, ep_;
11     double duration_ = 0.0;
12     double start_time_ = 0.0;
13     bool first_call_ = true;
14     bool traj_finished_ = false;
15
16 public:
17     MinJerkPositionTrajectory (double duration)
18     : duration_(duration)
19     {
```

(continues on next page)

(continued from previous page)

```

20 }
21
22 bool isTrajectoryFinished() {return traj_finished_;}
23
24 void resetTrajectory(const Eigen::Vector3d& start_position, const Eigen::Vector3d&
25   end_position)
26 {
27     sp_ = start_position;
28     ep_ = end_position;
29     alpha_ = ep_ - sp_;
30     first_call_ = true;
31     traj_finished_ = false;
32 }
33
34 void getDesired(double current_time, Eigen::Vector3d& p, Eigen::Vector3d& v,
35   Eigen::Vector3d& a)
36 {
37     if(first_call_)
38     {
39         start_time_ = current_time;
40         first_call_ = false;
41     }
42     double tau = (current_time - start_time_) / duration_;
43     if(tau >= 1.0)
44     {
45         p = ep_;
46         v = Eigen::Vector3d::Zero();
47         a = Eigen::Vector3d::Zero();
48
49         traj_finished_ = true;
50         return;
51     }
52     p =
53       sp_ + alpha_ * ( 10*pow(tau,3.0) - 15*pow(tau,4.
54       + 6*pow(tau,5.0) );
55     v = Eigen::Vector3d::Zero() + alpha_ * ( 30*pow(tau,2.0) - 60*pow(tau,3.0) +
56       -30*pow(tau,4.0) );
57     a = Eigen::Vector3d::Zero() + alpha_ * ( 60*pow(tau,1.0) - 180*pow(tau,2.0) +
58       -120*pow(tau,3.0) );
59   }
60 }
61
62 int main(int argc, char const *argv[])
63 {
64     if(argc < 2)
65     {
66         std::cerr << "Usage : " << argv[0] << " /path/to/robot-urdf.urdf (optionally -
67         -l debug/info/warning/error)" << "\n";
68         return -1;
69     }
70     std::string urdf_url(argv[1]);
71
72     orca::utils::Logger::parseArgv(argc, argv);
73
74     auto robot = std::make_shared<RobotDynTree>();
75     robot->loadModelFromFile(urdf_url);
76     robot->setBaseFrame("base_link");

```

(continues on next page)

(continued from previous page)

```

71     robot->setGravity(Eigen::Vector3d(0,0,-9.81));
72     RobotState eigState;
73     eigState.resize(robot->getNrOfDegreesOfFreedom());
74     eigState.jointPos.setZero();
75     eigState.jointVel.setZero();
76     robot->setRobotState(eigState.jointPos,eigState.jointVel);
77
78     orca::optim::Controller controller(
79         "controller"
80         ,robot
81         ,orca::optim::ResolutionStrategy::OneLevelWeighted
82         ,QPSSolver::qpOASES
83     );
84
85     auto cart_task = std::make_shared<CartesianTask>("CartTask-EE");
86     controller.addTask(cart_task);
87     cart_task->setControlFrame("link_7"); //
88     Eigen::Affine3d cart_pos_ref;
89     cart_pos_ref.translation() = Eigen::Vector3d(1.,0.75,0.5); // x,y,z in meters
90     cart_pos_ref.linear() = Eigen::Quaterniond::Identity().toRotationMatrix();
91     Vector6d cart_vel_ref = Vector6d::Zero();
92     Vector6d cart_acc_ref = Vector6d::Zero();
93
94     Vector6d P;
95     P << 1000, 1000, 1000, 10, 10, 10;
96     cart_task->servoController()->pid()->setProportionalGain(P);
97     Vector6d D;
98     D << 100, 100, 100, 1, 1, 1;
99     cart_task->servoController()->pid()->setDerivativeGain(D);
100
101
102     const int ndof = robot->getNrOfDegreesOfFreedom();
103
104     auto jnt_trq_cstr = std::make_shared<JointTorqueLimitConstraint>("JointTorqueLimit
105     ↵");
106     controller.addConstraint(jnt_trq_cstr);
107     Eigen::VectorXd jntTrqMax(ndof);
108     jntTrqMax.setConstant(200.0);
109     jnt_trq_cstr->setLimits(-jntTrqMax, jntTrqMax);
110
111     auto jnt_pos_cstr = std::make_shared<JointPositionLimitConstraint>(
112     ↵"JointPositionLimit");
113     controller.addConstraint(jnt_pos_cstr);
114
115     auto jnt_vel_cstr = std::make_shared<JointVelocityLimitConstraint>(
116     ↵"JointVelocityLimit");
117     controller.addConstraint(jnt_vel_cstr);
118     Eigen::VectorXd jntVelMax(ndof);
119     jntVelMax.setConstant(2.0);
120     jnt_vel_cstr->setLimits(-jntVelMax, jntVelMax);
121
122     double dt = 0.001;
123     double current_time = 0.0;
124
125     GazeboServer gzserver(argc,argv);
126     auto gzrobot = GazeboModel(gzserver.insertModelFromURDFFile(urdf_url));

```

(continues on next page)

(continued from previous page)

```

125 ///////////////////////////////////////////////////////////////////
126 ///////////////////////////////////////////////////////////////////
127 ///////////////////////////////////////////////////////////////////
128 ///////////////////////////////////////////////////////////////////
129
130 MinJerkPositionTrajectory traj(5.0);
131 int traj_loops = 0;
132 bool exit_control_loop = true;
133 Eigen::Vector3d start_position, end_position;
134
135
136 cart_task->onActivationCallback([] () {
137     std::cout << "Activating CartesianTask..." << '\n';
138 });
139
140 bool cart_task_activated = false;
141
142 cart_task->onActivatedCallback([&] () {
143     start_position = cart_task->servoController()->getCurrentCartesianPose() .
144     ↪block(0,3,3,1);
145     end_position = cart_pos_ref.translation();
146     traj.resetTrajectory(start_position, end_position);
147     std::cout << "CartesianTask activated. Removing gravity compensation and" .
148     ↪beginning motion." << '\n';
149     cart_task_activated = true;
150 });
151
152 cart_task->onComputeBeginCallback([&] (double current_time, double dt) {
153     if (cart_task->getState() == TaskBase::State::Activated)
154     {
155         Eigen::Vector3d p, v, a;
156         traj.getDesired(current_time, p, v, a);
157         cart_pos_ref.translation() = p;
158         cart_vel_ref.head(3) = v;
159         cart_acc_ref.head(3) = a;
160         cart_task->servoController()->setDesired(cart_pos_ref.matrix(), cart_vel_
161         ↪ref, cart_acc_ref);
162     }
163 });
164
165 cart_task->onComputeEndCallback([&] (double current_time, double dt) {
166     if (cart_task->getState() == TaskBase::State::Activated)
167     {
168         if (traj.isTrajectoryFinished())
169         {
170             if (traj_loops < 5)
171             {
172                 // flip start and end positions.
173                 auto ep = end_position;
174                 end_position = start_position;
175                 start_position = ep;
176                 traj.resetTrajectory(start_position, end_position);
177                 std::cout << "Changing trajectory direction." << '\n';
178                 ++traj_loops;
179             }
180             else
181             {

```

(continues on next page)

(continued from previous page)

```

179         std::cout << "Trajectory looping finished. Deactivating task and_"
180         ↪starting gravity compensation." << '\n';
181         cart_task->deactivate();
182     }
183 }
184 );
185
186 cart_task->onDeactivationCallback([&cart_taskActivated] () {
187     std::cout << "Deactivating task." << '\n';
188     cart_taskActivated = false;
189 });
190
191 cart_task->onDeactivatedCallback([] () {
192     std::cout << "CartesianTask deactivated. Stopping controller" << '\n';
193 });
194
195
196
197 gzrobot.setCallback([&] (uint32_t n_iter, double current_time, double dt)
198 {
199     robot->setRobotState(gzrobot.getWorldToBaseTransform().matrix()
200                             , gzrobot.getJointPositions()
201                             , gzrobot.getBaseVelocity()
202                             , gzrobot.getJointVelocities()
203                             , gzrobot.getGravity()
204                             );
205 // All tasks need the robot to be initialized during the activation phase
206 if(n_iter == 1)
207     controller.activateTasksAndConstraints();
208
209 controller.update(current_time, dt);
210
211 if (cart_taskActivated)
212 {
213     if(controller.solutionFound())
214     {
215         gzrobot.setJointTorqueCommand( controller.getJointTorqueCommand() );
216     }
217     else
218     {
219         gzrobot.setBrakes(true);
220     }
221 }
222 else
223 {
224     gzrobot.setJointGravityTorques(robot->getJointGravityTorques());
225 }
226 );
227
228 std::cout << "Simulation running... (GUI with \'gzclient\')" << "\n";
229 gzserver.run();
230 return 0;
231 }
```

## 1.1.9 Plotting

### Using the internal plotting tools

**Note:** The source code for this example can be found in [orca\_root]/examples/plotting/01-plotting\_torques.cc, or alternatively on github at: [https://github.com/syroco/orca/blob/dev/examples/plotting/01-plotting\\_torques.cc](https://github.com/syroco/orca/blob/dev/examples/plotting/01-plotting_torques.cc)

```

1 #include <orca/orca.h>
2 #include <matplotlibcpp/matplotlibcpp.h>
3 using namespace orca::all;
4
5 namespace plt = matplotlibcpp;
6
7 int main(int argc, char const *argv[])
8 {
9     // Get the urdf file from the command line
10    if(argc < 2)
11    {
12        std::cerr << "Usage : " << argv[0] << " /path/to/robot-urdf.urdf (optionally -l debug/info/warning/error)" << "\n";
13        return -1;
14    }
15    std::string urdf_url(argv[1]);
16
17    // Parse logger level as --log_level (or -l) debug/warning etc
18    orca::utils::Logger::parseArgv(argc, argv);
19
20    // Create the kinematic model that is shared by everybody
21    auto robot = std::make_shared<RobotDynTree>(); // Here you can pass a robot name
22    robot->loadModelFromFile(urdf_url); // If you don't pass a robot name, it is extracted from the urdf
23    robot->setBaseFrame("base_link"); // All the transformations (end effector pose for example) will be expressed wrt this base frame
24    robot->setGravity(Eigen::Vector3d(0,0,-9.81)); // Sets the world gravity
25    // (Optional)
26
27    // This is an helper function to store the whole state of the robot as eigen_
28    // vectors/matrices
29    // This class is totally optional, it is just meant to keep consistency for the_
30    // sizes of all the vectors/matrices
31    // You can use it to fill data from either real robot and simulated robot
32    RobotState eigState;
33    eigState.resize(robot->getNrOfDegreesOfFreedom()); // resize all the vectors/
34    //matrices to match the robot configuration
35    // Set the initial state to zero (arbitrary)
36    // NOTE : here we only set q, qdot because this example asserts we have a fixed_
37    //base robot
38    eigState.jointPos.setZero();
    eigState.jointVel.setZero();
    // Set the first state to the robot
    robot->setRobotState(eigState.jointPos,eigState.jointVel); // Now is the robot is considered 'initialized'
    // Instanciate an ORCA Controller

```

(continues on next page)

(continued from previous page)

```

39     orca::optim::Controller controller(
40         "controller"
41         , robot
42         , orca::optim::ResolutionStrategy::OneLevelWeighted // MultiLevelWeighted,
43         ↪Generalized
44         , QPSolver::qpOASES
45     );
46
47     // Cartesian Task
48     auto cart_task = std::make_shared<CartesianTask>("CartTask-EE");
49     controller.addTask(cart_task); // Add the task to the controller to initialize it
50     // Set the frame you want to control
51     cart_task->setControlFrame("link_7"); // We want to control the link_7
52
53     // Set the pose desired for the link_7
54     Eigen::Affine3d cart_pos_ref;
55     // Translation
56     cart_pos_ref.translation() = Eigen::Vector3d(1., 0.75, 0.5); // x,y,z in meters
57     // Rotation is done with a Matrix3x3
58     Eigen::Quaterniond quat;
59     // Example 1 : create a quaternion from Euler angles ZYZ convention
60     quat = Eigen::AngleAxisd(0, Eigen::Vector3d::UnitZ())
61         * Eigen::AngleAxisd(0, Eigen::Vector3d::Unity())
62         * Eigen::AngleAxisd(0, Eigen::Vector3d::UnitZ());
63     // Example 2 : create a quaternion from RPY convention
64     cart_pos_ref.linear() = quatFromRPY(0, 0, 0).toRotationMatrix();
65     // Example 3 : create a quaternion from Kuka Convention
66     cart_pos_ref.linear() = quatFromKukaConvention(0, 0, 0).toRotationMatrix();
67
68     // Set the desired cartesian velocity to zero
69     Vector6d cart_vel_ref;
70     cart_vel_ref.setZero();
71
72     // Set the desired cartesian velocity to zero
73     Vector6d cart_acc_ref;
74     cart_acc_ref.setZero();
75
76     // Now set the servoing PID
77     Vector6d P;
78     P << 1000, 1000, 1000, 10, 10, 10;
79     cart_task->servoController()->pid()->setProportionalGain(P);
80     Vector6d D;
81     D << 100, 100, 100, 1, 1, 1;
82     cart_task->servoController()->pid()->setDerivativeGain(D);
83     // The desired values are set on the servo controller
84     // Because cart_task->setDesired expects a cartesian acceleration
85     // Which is computed automatically by the servo controller
86     cart_task->servoController()->setDesired(cart_pos_ref.matrix(), cart_vel_ref, cart_
87     ↪acc_ref);
88
89     // Get the number of actuated joints
90     const int ndof = robot->getNrOfDegreesOfFreedom();
91
92     // Joint torque limit is usually given by the robot manufacturer
93     auto jnt_trq_cstr = std::make_shared<JointTorqueLimitConstraint>("JointTorqueLimit
94     ↪");
95     controller.addConstraint(jnt_trq_cstr); // Add the constraint to the controller
96     ↪to initialize it

```

(continues on next page)

(continued from previous page)

```

93 Eigen::VectorXd jntTrqMax(ndof);
94 jntTrqMax.setConstant(200.0);
95 jnt_trq_cstr->setLimits(-jntTrqMax, jntTrqMax); // because not read in the URDF
96 for now

97 // Joint position limits are automatically extracted from the URDF model
98 // Note that you can set them if you want
99 // by simply doing jnt_pos_cstr->setLimits(jntPosMin, jntPosMax);
100 auto jnt_pos_cstr = std::make_shared<JointPositionLimitConstraint>(
101 "JointPositionLimit");
102 controller.addConstraint(jnt_pos_cstr); // Add the constraint to the controller
103 to initialize it

104 // Joint velocity limits are usually given by the robot manufacturer
105 auto jnt_vel_cstr = std::make_shared<JointVelocityLimitConstraint>(
106 "JointVelocityLimit");
107 controller.addConstraint(jnt_vel_cstr); // Add the constraint to the controller
108 to initialize it
109 Eigen::VectorXd jntVelMax(ndof);
110 jntVelMax.setConstant(2.0);
111 jnt_vel_cstr->setLimits(-jntVelMax, jntVelMax); // because not read in the URDF
112 for now

113 double dt = 0.001;
114 double total_time = 1.0;
115 double current_time = 0;

116 // Shortcut : activate all tasks
117 controller.activateTasksAndConstraints();

118 // Now you can run the control loop
119 std::vector<double> time_log;
120 int ncols = std::ceil(total_time/dt);
121 Eigen::MatrixXd torqueMat(ndof,ncols);
122 torqueMat.setZero();

123 for (int count = 0; current_time < total_time; current_time +=dt)
124 {
125     time_log.push_back(current_time);

126     // Here you can get the data from your REAL robot (API might vary)
127     // Some thing like :
128     //     eigState.jointPos = myRealRobot.getJointPositions();
129     //     eigState.jointVel = myRealRobot.getJointVelocities();

130     // Now update the internal kinematic model with data from REAL robot
131     robot->setRobotState(eigState.jointPos,eigState.jointVel);

132     // Step the controller
133     if(controller.update(current_time,dt))
134     {

135         // Get the controller output
136         const Eigen::VectorXd& full_solution = controller.getSolution();
137

138         torqueMat.col(count) = controller.getJointTorqueCommand();
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
625
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1630
1631
1631
1632
1632
1633
1633
1634
1634
1635
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1640
1641
1641
1642
1642
1643
1643
1644
1644
1645
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1650
1651
1651
1652
1652
1653
1653
1654
1654
1655
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1660
1661
1661
1662
1662
1663
1663
1664
1664
1665
1665
1666
1666
1667
1667
1668
1668
1669
1669
1
```

(continues on next page)

(continued from previous page)

```

144     const Eigen::VectorXd& trq_acc = controller.getJointAccelerationCommand();
145
146     // Here you can send the commands to your REAL robot
147     // Something like :
148     // myRealRobot.setTorqueCommand(trq_cmd);
149 }
150 else
151 {
152     // Controller could not get the optimal torque
153     // Now you have to save your robot
154     // You can get the return code with controller.getReturnCode();
155 }
156
157 count++;
158
159 std::cout << "current_time " << current_time << '\n';
160 std::cout << "total_time " << total_time << '\n';
161 std::cout << "time log size " << time_log.size() << '\n';
162 std::cout << "torqueMat.cols " << torqueMat.cols() << '\n';
163 }
164
165 // Print the last computed solution (just for fun)
166 const Eigen::VectorXd& full_solution = controller.getSolution();
167 const Eigen::VectorXd& trq_cmd = controller.getJointTorqueCommand();
168 const Eigen::VectorXd& trq_acc = controller.getJointAccelerationCommand();
169 LOG_INFO << "Full solution : " << full_solution.transpose();
170 LOG_INFO << "Joint Acceleration command : " << trq_acc.transpose();
171 LOG_INFO << "Joint Torque command : " << trq_cmd.transpose();
172
173 // At some point you want to close the controller nicely
174 controller.deactivateTasksAndConstraints();
175 // Let all the tasks ramp down to zero
176 while (!controller.tasksAndConstraintsDeactivated())
177 {
178     current_time += dt;
179     controller.print();
180     controller.update(current_time, dt);
181 }
182
183 // Plot data
184 for (size_t i = 0; i < torqueMat.rows(); i++)
185 {
186     std::vector<double> trq(time_log.size());
187     Eigen::VectorXd::Map(trq.data(), time_log.size()) = torqueMat.row(i);
188     plt::plot(time_log, trq);
189 }
190 plt::show();
191 return 0;
192 }
```

### 1.1.10 Overview

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 optimization vector
- $H$  the hessian matrix ( $\text{size}(x) \times \text{size}(x)$ )
- $g$  the gradient vector ( $\text{size}(x) \times 1$ )
- $A$  the constraint matrix ( $\text{size}(x) \times \text{size}(x)$ )
- $lb$  and  $ub$  the lower and upper bounds of  $x$  ( $\text{size}(x) \times 1$ )
- $lb_A$  and  $ub_A$  the lower and upper bounds of  $A$  ( $\text{size}(x) \times 1$ )

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

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

- $x$  the optimization vector, or **part** of the optimization vector
- $E$  the linear matrix of the affine function ( $\text{size}(x) \times \text{size}(x)$ )
- $f$  the origin vector ( $\text{size}(x) \times 1$ )
- $w_{task}$  the weight of the tasks in the overall quadratic cost (scalar  $[0 : 1]$ )
- $W_{norm}$  the weight of the euclidean norm ( $\text{size}(x) \times \text{size}(x)$ )

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

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

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

$$lb_C \leq Cx \leq ub_C$$

- $C$  the constraint matrix ( $\text{size}(x) \times \text{size}(x)$ )
- $lb_C$  and  $ub_C$  the lower and upper bounds of  $A$  ( $\text{size}(x) \times 1$ )

### 1.1.11 Optimization Vector

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

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

### 1.1.12 Cartesian Acceleration

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

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

### 1.1.13 Dynamics Equation

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

$$[-M \quad S_\tau \quad J_{{}^e w}] X = C + G$$

```
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 optimization problem
```

## 1.1.14 License

### CeCILL-C FREE SOFTWARE LICENSE AGREEMENT

#### Notice

This Agreement is a Free Software license agreement that is the result of discussions between its authors in order to ensure compliance with the two main principles guiding its drafting:

- firstly, compliance with the principles governing the distribution of Free Software: access to source code, broad rights granted to users,
- secondly, the election of a governing law, French law, with which it is conformant, both as regards the law of torts and intellectual property law, and the protection that it offers to both authors and holders of the economic rights over software.

The authors of the CeCILL-C (for Ce[a] C[nrs] I[nria] L[ogiciel] L[ibre]) license are:

Commissariat à l'Energie Atomique - CEA, a public scientific, technical and industrial research establishment, having its principal place of business at 25 rue Leblanc, immeuble Le Ponant D, 75015 Paris, France.

Centre National de la Recherche Scientifique - CNRS, a public scientific and technological establishment, having its principal place of business at 3 rue Michel-Ange, 75794 Paris cedex 16, France.

Institut National de Recherche en Informatique et en Automatique - INRIA, a public scientific and technological establishment, having its principal place of business at Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay cedex, France.

#### Preamble

The purpose of this Free Software license agreement is to grant users the right to modify and re-use the software governed by this license.

The exercising of this right is conditional upon the obligation to make available to the community the modifications made to the source code of the software so as to contribute to its evolution.

In consideration of access to the source code and the rights to copy, modify and redistribute granted by the license, users are provided only with a limited warranty and the software's author, the holder of the economic rights, and the successive licensors only have limited liability.

In this respect, the risks associated with loading, using, modifying and/or developing or reproducing the software by the user are brought to the user's attention, given its Free Software status, which may make it complicated to use, with the result that its use is reserved for developers and experienced professionals having in-depth computer knowledge. Users are therefore encouraged to load and test the suitability of the software as regards their requirements in conditions enabling the security of their systems and/or data to be ensured and, more generally, to use and operate it in the same conditions of security. This Agreement may be freely reproduced and published, provided it is not altered, and that no provisions are either added or removed herefrom.

This Agreement may apply to any or all software for which the holder of the economic rights decides to submit the use thereof to its provisions.

#### Article 1 - DEFINITIONS

For the purpose of this Agreement, when the following expressions commence with a capital letter, they shall have the following meaning:

**Agreement:** means this license agreement, and its possible subsequent versions and annexes.

**Software:** means the software in its Object Code and/or Source Code form and, where applicable, its documentation, "as is" when the Licensee accepts the Agreement.

**Initial Software:** means the Software in its Source Code and possibly its Object Code form and, where applicable, its documentation, "as is" when it is first distributed under the terms and conditions of the Agreement.

Modified Software: means the Software modified by at least one Integrated Contribution.

Source Code: means all the Software's instructions and program lines to which access is required so as to modify the Software.

Object Code: means the binary files originating from the compilation of the Source Code.

Holder: means the holder(s) of the economic rights over the Initial Software.

Licensee: means the Software user(s) having accepted the Agreement.

Contributor: means a Licensee having made at least one Integrated Contribution.

Licensor: means the Holder, or any other individual or legal entity, who distributes the Software under the Agreement.

Integrated Contribution: means any or all modifications, corrections, translations, adaptations and/or new functions integrated into the Source Code by any or all Contributors.

Related Module: means a set of sources files including their documentation that, without modification to the Source Code, enables supplementary functions or services in addition to those offered by the Software.

Derivative Software: means any combination of the Software, modified or not, and of a Related Module.

Parties: mean both the Licensee and the Licensor.

These expressions may be used both in singular and plural form.

### **Article 2 - PURPOSE**

The purpose of the Agreement is the grant by the Licensor to the Licensee of a non-exclusive, transferable and worldwide license for the Software as set forth in Article 5 hereinafter for the whole term of the protection granted by the rights over said Software.

### **Article 3 - ACCEPTANCE**

3.1 The Licensee shall be deemed as having accepted the terms and conditions of this Agreement upon the occurrence of the first of the following events:

- (i) loading the Software by any or all means, notably, by downloading from a remote server, or by loading from a physical medium;
- (ii) the first time the Licensee exercises any of the rights granted hereunder.

3.2 One copy of the Agreement, containing a notice relating to the characteristics of the Software, to the limited warranty, and to the fact that its use is restricted to experienced users has been provided to the Licensee prior to its acceptance as set forth in Article 3.1 hereinabove, and the Licensee hereby acknowledges that it has read and understood it.

### **Article 4 - EFFECTIVE DATE AND TERM**

#### **4.1 EFFECTIVE DATE**

The Agreement shall become effective on the date when it is accepted by the Licensee as set forth in Article 3.1.

#### **4.2 TERM**

The Agreement shall remain in force for the entire legal term of protection of the economic rights over the Software.

### **Article 5 - SCOPE OF RIGHTS GRANTED**

The Licensor hereby grants to the Licensee, who accepts, the following rights over the Software for any or all use, and for the term of the Agreement, on the basis of the terms and conditions set forth hereinafter.

Besides, if the Licensor owns or comes to own one or more patents protecting all or part of the functions of the Software or of its components, the Licensor undertakes not to enforce the rights granted by these patents against successive Licensees using, exploiting or modifying the Software. If these patents are transferred, the Licensor undertakes to have the transferees subscribe to the obligations set forth in this paragraph.

### **5.1 RIGHT OF USE**

The Licensee is authorized to use the Software, without any limitation as to its fields of application, with it being hereinafter specified that this comprises:

1. permanent or temporary reproduction of all or part of the Software by any or all means and in any or all form.
2. loading, displaying, running, or storing the Software on any or all medium.
3. entitlement to observe, study or test its operation so as to determine the ideas and principles behind any or all constituent elements of said Software. This shall apply when the Licensee carries out any or all loading, displaying, running, transmission or storage operation as regards the Software, that it is entitled to carry out hereunder.

### **5.2 RIGHT OF MODIFICATION**

The right of modification includes the right to translate, adapt, arrange, or make any or all modifications to the Software, and the right to reproduce the resulting software. It includes, in particular, the right to create a Derivative Software.

The Licensee is authorized to make any or all modification to the Software provided that it includes an explicit notice that it is the author of said modification and indicates the date of the creation thereof.

### **5.3 RIGHT OF DISTRIBUTION**

In particular, the right of distribution includes the right to publish, transmit and communicate the Software to the general public on any or all medium, and by any or all means, and the right to market, either in consideration of a fee, or free of charge, one or more copies of the Software by any means.

The Licensee is further authorized to distribute copies of the modified or unmodified Software to third parties according to the terms and conditions set forth hereinafter.

#### **5.3.1 DISTRIBUTION OF SOFTWARE WITHOUT MODIFICATION**

The Licensee is authorized to distribute true copies of the Software in Source Code or Object Code form, provided that said distribution complies with all the provisions of the Agreement and is accompanied by:

1. a copy of the Agreement,
2. a notice relating to the limitation of both the Lessor's warranty and liability as set forth in Articles 8 and 9,

and that, in the event that only the Object Code of the Software is redistributed, the Licensee allows effective access to the full Source Code of the Software at a minimum during the entire period of its distribution of the Software, it being understood that the additional cost of acquiring the Source Code shall not exceed the cost of transferring the data.

#### **5.3.2 DISTRIBUTION OF MODIFIED SOFTWARE**

When the Licensee makes an Integrated Contribution to the Software, the terms and conditions for the distribution of the resulting Modified Software become subject to all the provisions of this Agreement.

The Licensee is authorized to distribute the Modified Software, in source code or object code form, provided that said distribution complies with all the provisions of the Agreement and is accompanied by:

1. a copy of the Agreement,
2. a notice relating to the limitation of both the Lessor's warranty and liability as set forth in Articles 8 and 9,

and that, in the event that only the object code of the Modified Software is redistributed, the Licensee allows effective access to the full source code of the Modified Software at a minimum during the entire period of its distribution of the Modified Software, it being understood that the additional cost of acquiring the source code shall not exceed the cost of transferring the data.

#### **5.3.3 DISTRIBUTION OF DERIVATIVE SOFTWARE**

When the Licensee creates Derivative Software, this Derivative Software may be distributed under a license agreement other than this Agreement, subject to compliance with the requirement to include a notice concerning the rights over the Software as defined in Article 6.4. In the event the creation of the Derivative Software required modification of the Source Code, the Licensee undertakes that:

1. the resulting Modified Software will be governed by this Agreement,
2. the Integrated Contributions in the resulting Modified Software will be clearly identified and documented,
3. the Licensee will allow effective access to the source code of the Modified Software, at a minimum during the entire period of distribution of the Derivative Software, such that such modifications may be carried over in a subsequent version of the Software; it being understood that the additional cost of purchasing the source code of the Modified Software shall not exceed the cost of transferring the data.

### 5.3.4 COMPATIBILITY WITH THE CeCILL LICENSE

When a Modified Software contains an Integrated Contribution subject to the CeCILL license agreement, or when a Derivative Software contains a Related Module subject to the CeCILL license agreement, the provisions set forth in the third item of Article 6.4 are optional.

## Article 6 - INTELLECTUAL PROPERTY

### 6.1 OVER THE INITIAL SOFTWARE

The Holder owns the economic rights over the Initial Software. Any or all use of the Initial Software is subject to compliance with the terms and conditions under which the Holder has elected to distribute its work and no one shall be entitled to modify the terms and conditions for the distribution of said Initial Software.

The Holder undertakes that the Initial Software will remain ruled at least by this Agreement, for the duration set forth in Article 4.2.

### 6.2 OVER THE INTEGRATED CONTRIBUTIONS

The Licensee who develops an Integrated Contribution is the owner of the intellectual property rights over this Contribution as defined by applicable law.

### 6.3 OVER THE RELATED MODULES

The Licensee who develops a Related Module is the owner of the intellectual property rights over this Related Module as defined by applicable law and is free to choose the type of agreement that shall govern its distribution under the conditions defined in Article 5.3.3.

### 6.4 NOTICE OF RIGHTS

The Licensee expressly undertakes:

1. not to remove, or modify, in any manner, the intellectual property notices attached to the Software;
2. to reproduce said notices, in an identical manner, in the copies of the Software modified or not;
3. to ensure that use of the Software, its intellectual property notices and the fact that it is governed by the Agreement is indicated in a text that is easily accessible, specifically from the interface of any Derivative Software.

The Licensee undertakes not to directly or indirectly infringe the intellectual property rights of the Holder and/or Contributors on the Software and to take, where applicable, vis-à-vis its staff, any and all measures required to ensure respect of said intellectual property rights of the Holder and/or Contributors.

## Article 7 - RELATED SERVICES

7.1 Under no circumstances shall the Agreement oblige the Licensor to provide technical assistance or maintenance services for the Software.

However, the Licensor is entitled to offer this type of services. The terms and conditions of such technical assistance, and/or such maintenance, shall be set forth in a separate instrument. Only the Licensor offering said maintenance and/or technical assistance services shall incur liability therefor.

7.2 Similarly, any Licensor is entitled to offer to its licensees, under its sole responsibility, a warranty, that shall only be binding upon itself, for the redistribution of the Software and/or the Modified Software, under terms and conditions that it is free to decide. Said warranty, and the financial terms and conditions of its application, shall be subject of a separate instrument executed between the Licensor and the Licensee.

#### Article 8 - LIABILITY

8.1 Subject to the provisions of Article 8.2, the Licensee shall be entitled to claim compensation for any direct loss it may have suffered from the Software as a result of a fault on the part of the relevant Licensor, subject to providing evidence thereof.

8.2 The Licensor's liability is limited to the commitments made under this Agreement and shall not be incurred as a result of in particular: (i) loss due the Licensee's total or partial failure to fulfill its obligations, (ii) direct or consequential loss that is suffered by the Licensee due to the use or performance of the Software, and (iii) more generally, any consequential loss. In particular the Parties expressly agree that any or all pecuniary or business loss (i.e. loss of data, loss of profits, operating loss, loss of customers or orders, opportunity cost, any disturbance to business activities) or any or all legal proceedings instituted against the Licensee by a third party, shall constitute consequential loss and shall not provide entitlement to any or all compensation from the Licensor.

#### Article 9 - WARRANTY

9.1 The Licensee acknowledges that the scientific and technical state-of-the-art when the Software was distributed did not enable all possible uses to be tested and verified, nor for the presence of possible defects to be detected. In this respect, the Licensee's attention has been drawn to the risks associated with loading, using, modifying and/or developing and reproducing the Software which are reserved for experienced users.

The Licensee shall be responsible for verifying, by any or all means, the suitability of the product for its requirements, its good working order, and for ensuring that it shall not cause damage to either persons or properties.

9.2 The Licensor hereby represents, in good faith, that it is entitled to grant all the rights over the Software (including in particular the rights set forth in Article 5).

9.3 The Licensee acknowledges that the Software is supplied "as is" by the Licensor without any other express or tacit warranty, other than that provided for in Article 9.2 and, in particular, without any warranty as to its commercial value, its secured, safe, innovative or relevant nature.

Specifically, the Licensor does not warrant that the Software is free from any error, that it will operate without interruption, that it will be compatible with the Licensee's own equipment and software configuration, nor that it will meet the Licensee's requirements.

9.4 The Licensor does not either expressly or tacitly warrant that the Software does not infringe any third party intellectual property right relating to a patent, software or any other property right. Therefore, the Licensor disclaims any and all liability towards the Licensee arising out of any or all proceedings for infringement that may be instituted in respect of the use, modification and redistribution of the Software. Nevertheless, should such proceedings be instituted against the Licensee, the Licensor shall provide it with technical and legal assistance for its defense. Such technical and legal assistance shall be decided on a case-by-case basis between the relevant Licensor and the Licensee pursuant to a memorandum of understanding. The Licensor disclaims any and all liability as regards the Licensee's use of the name of the Software. No warranty is given as regards the existence of prior rights over the name of the Software or as regards the existence of a trademark.

#### Article 10 - TERMINATION

10.1 In the event of a breach by the Licensee of its obligations hereunder, the Licensor may automatically terminate this Agreement thirty (30) days after notice has been sent to the Licensee and has remained ineffective.

10.2 A Licensee whose Agreement is terminated shall no longer be authorized to use, modify or distribute the Software. However, any licenses that it may have granted prior to termination of the Agreement shall remain valid subject to their having been granted in compliance with the terms and conditions hereof.

#### Article 11 - MISCELLANEOUS

#### **11.1 EXCUSABLE EVENTS**

Neither Party shall be liable for any or all delay, or failure to perform the Agreement, that may be attributable to an event of force majeure, an act of God or an outside cause, such as defective functioning or interruptions of the electricity or telecommunications networks, network paralysis following a virus attack, intervention by government authorities, natural disasters, water damage, earthquakes, fire, explosions, strikes and labor unrest, war, etc.

11.2 Any failure by either Party, on one or more occasions, to invoke one or more of the provisions hereof, shall under no circumstances be interpreted as being a waiver by the interested Party of its right to invoke said provision(s) subsequently.

11.3 The Agreement cancels and replaces any or all previous agreements, whether written or oral, between the Parties and having the same purpose, and constitutes the entirety of the agreement between said Parties concerning said purpose. No supplement or modification to the terms and conditions hereof shall be effective as between the Parties unless it is made in writing and signed by their duly authorized representatives.

11.4 In the event that one or more of the provisions hereof were to conflict with a current or future applicable act or legislative text, said act or legislative text shall prevail, and the Parties shall make the necessary amendments so as to comply with said act or legislative text. All other provisions shall remain effective. Similarly, invalidity of a provision of the Agreement, for any reason whatsoever, shall not cause the Agreement as a whole to be invalid.

#### **11.5 LANGUAGE**

The Agreement is drafted in both French and English and both versions are deemed authentic.

#### **Article 12 - NEW VERSIONS OF THE AGREEMENT**

12.1 Any person is authorized to duplicate and distribute copies of this Agreement.

12.2 So as to ensure coherence, the wording of this Agreement is protected and may only be modified by the authors of the License, who reserve the right to periodically publish updates or new versions of the Agreement, each with a separate number. These subsequent versions may address new issues encountered by Free Software.

12.3 Any Software distributed under a given version of the Agreement may only be subsequently distributed under the same version of the Agreement or a subsequent version.

#### **Article 13 - GOVERNING LAW AND JURISDICTION**

13.1 The Agreement is governed by French law. The Parties agree to endeavor to seek an amicable solution to any disagreements or disputes that may arise during the performance of the Agreement.

13.2 Failing an amicable solution within two (2) months as from their occurrence, and unless emergency proceedings are necessary, the disagreements or disputes shall be referred to the Paris Courts having jurisdiction, by the more diligent Party.

Version 1.0 dated 2006-09-05.

# CHAPTER 2

---

## Authorship

---

Work on ORCA initially began in 2017 at the Institut des Systèmes Intelligents et de Robotique (ISIR). Since January 2018, active maintenance and development has been taken over by Fuzzy Logic Robotics S.A.S.

### 2.1 Maintainers

- Antoine Hoarau
- Ryan Lober
- Fuzzy Logic Robotics ([info@fuzzylogicrobotics.com](mailto:info@fuzzylogicrobotics.com))

### 2.2 Contributors

- Vincent Padois

### 2.3 Related Publications

### 2.4 Partner Institutions



