
CARDSFlow

Release 0.0.

Simon Trendel

Oct 13, 2019

USAGE AND INSTALLATION

1	Installation	3
2	Getting started	5
3	Working with CASPR	13
4	Context	19
5	Technical Interfaces	21
6	User Interfaces	23
7	API	27
	Index	153

Contents:

INSTALLATION

1.1 Prerequisites

The CARDSflow repo has been tested on Ubuntu 16.04 with ROS kinetic and Ubuntu 18.04 with ROS melodic. It may work on different platforms with different ROS versions. Make sure you install the following packages:

```
sudo apt install ros-$ROS_DISTRO-desktop-full libeigen3-dev libxml2-dev coinor-  
↳libipopt-dev \  
qtbase5-dev qtdeclarative5-dev qtmultimedia5-dev qml-module-qtquick2 \  
qml-module-qtquick-window2 qml-module-qtmultimedia qml-module-qtquick-dialogs \  
qml-module-qtquick-controls qml-module-qt-labs-folderlistmodel qml-module-qt-labs-  
↳settings \  
ros-$ROS_DISTRO-moveit-msgs doxygen swig mscgen ros-$ROS_DISTRO-grid-map \  
ros-$ROS_DISTRO-controller-interface ros-$ROS_DISTRO-controller-manager ros-$ROS_  
↳DISTRO-aruco-detect \  
ros-$ROS_DISTRO-effort-controllers libxml++2.6-dev ros-$ROS_DISTRO-robot-localization_  
↳libalglib-dev \  
ros-$ROS_DISTRO-tf ros-$ROS_DISTRO-interactive-markers ros-$ROS_DISTRO-tf-conversions_  
↳\  
ros-$ROS_DISTRO-robot-state-publisher
```

1.2 JSON

Install the newest json (version 3.6.1 at this time):

```
git clone https://github.com/nlohmann/json  
cd json && mkdir build && cd build  
cmake ..  
make -j8  
sudo make install
```

1.3 Get The Code

Clone the code to your catkin workspace:

```
cd path/to/your/catkin_ws/src  
git clone --recursive https://github.com/CARDSflow/CARDSflow
```

1.4 Build From Source

First we need to build iDynTree (for more details on this step, visit the [iDynTree](#) website):

```
cd path/to/CARDSflow/iDyntree
mkdir build && cd build
cmake ..
make -j9
sudo make install
```

Next we need to build and install qpOases:

```
cd qpOASES/
mkdir build && cd build
cmake ../
sudo make -j9 install
```

Now you can build CARDSflow simply by building your catkin workspace:

```
cd path/to/your/catkin_ws
catkin_make
source devel/setup.bash
```

1.5 RaspberryPi 3B+ installation

Download the image using the following command:

```
wget -nv http://bot.robey.org:8081/~robey/CARDSflow_raspberry_pi3B+.md5sum
wget -nv http://bot.robey.org:8081/~robey/CARDSflow_raspberry_pi3B+.img
```

Use at least a 16GB SDcard and flash the image using the following command. NOTE: make sure you are using the correct sd-card device (/dev/sdX), otherwise you might wipe your whole system!!!

```
sudo dd if=CARDSflow_raspberry_pi3B+.img of=/dev/sdX bs=1M status=progress
```

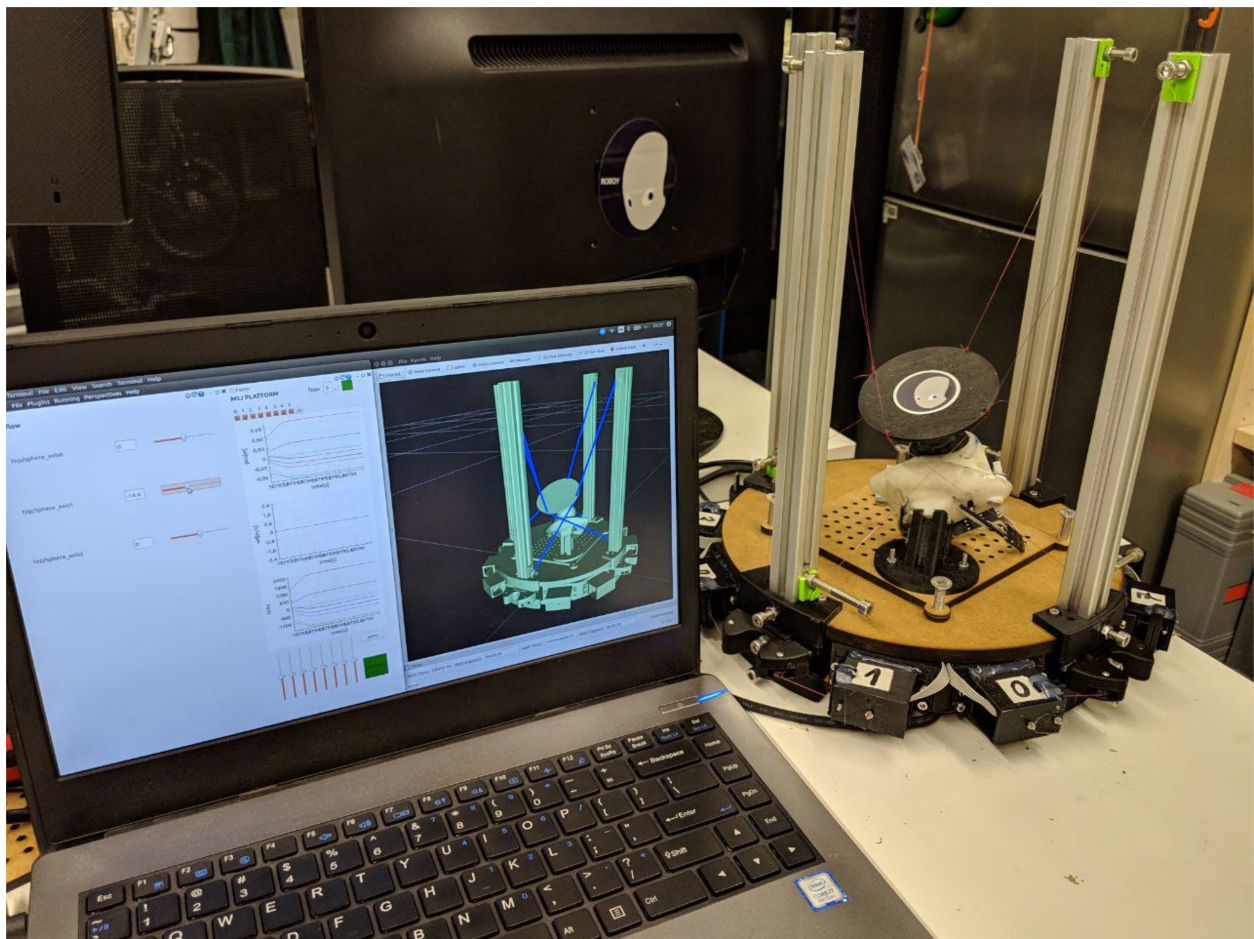
Insert the sd-card into the raspi and power it. The image provides Ubuntu 16.03 MATE with ROS kinetic installed. In the home folder you can find the robey_arcade_maschine workspace which contains a build version of CARDSflow.

GETTING STARTED

2.1 Booting up the robot

In the kindyn package a couple of example robots can be found. You can run them with the provided robot.launch file:

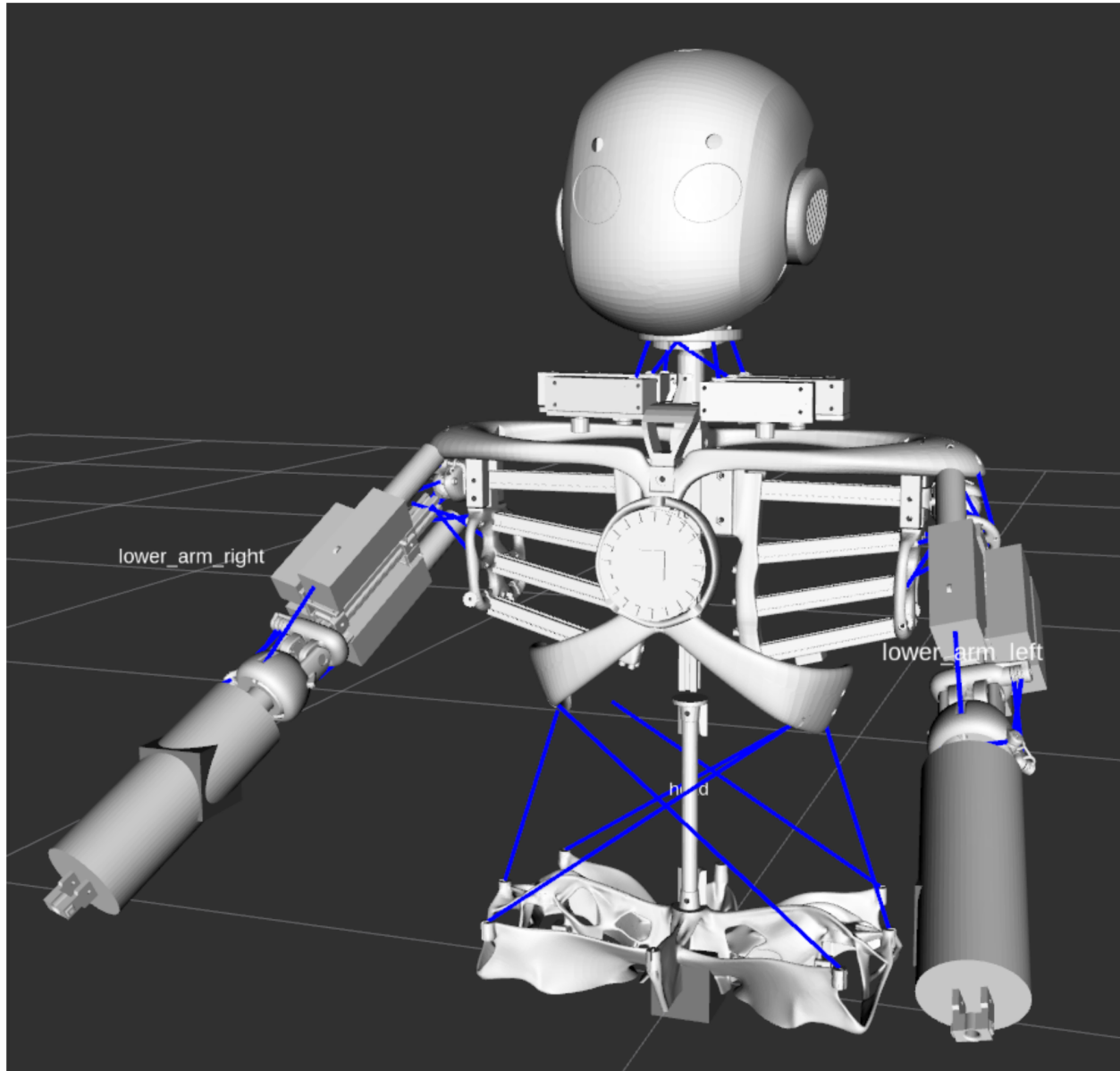
2.1.1 msj_platform



A robot that was build for training one of the 3-DoF shoulders of Roboy 2.0. It uses 8 cables to control the 3-DoFs. You can run it with the following command:

```
roslaunch kindyn robot.launch robot_name:=msj_platform start_controllers:='sphere_
↳axis0 sphere_axis1 sphere_axis2'
```

2.1.2 robo_upper_body



Roboy 2.0 upper body with the 3-DoF head and his two 5-DoF arms. Use the following command to launch it:

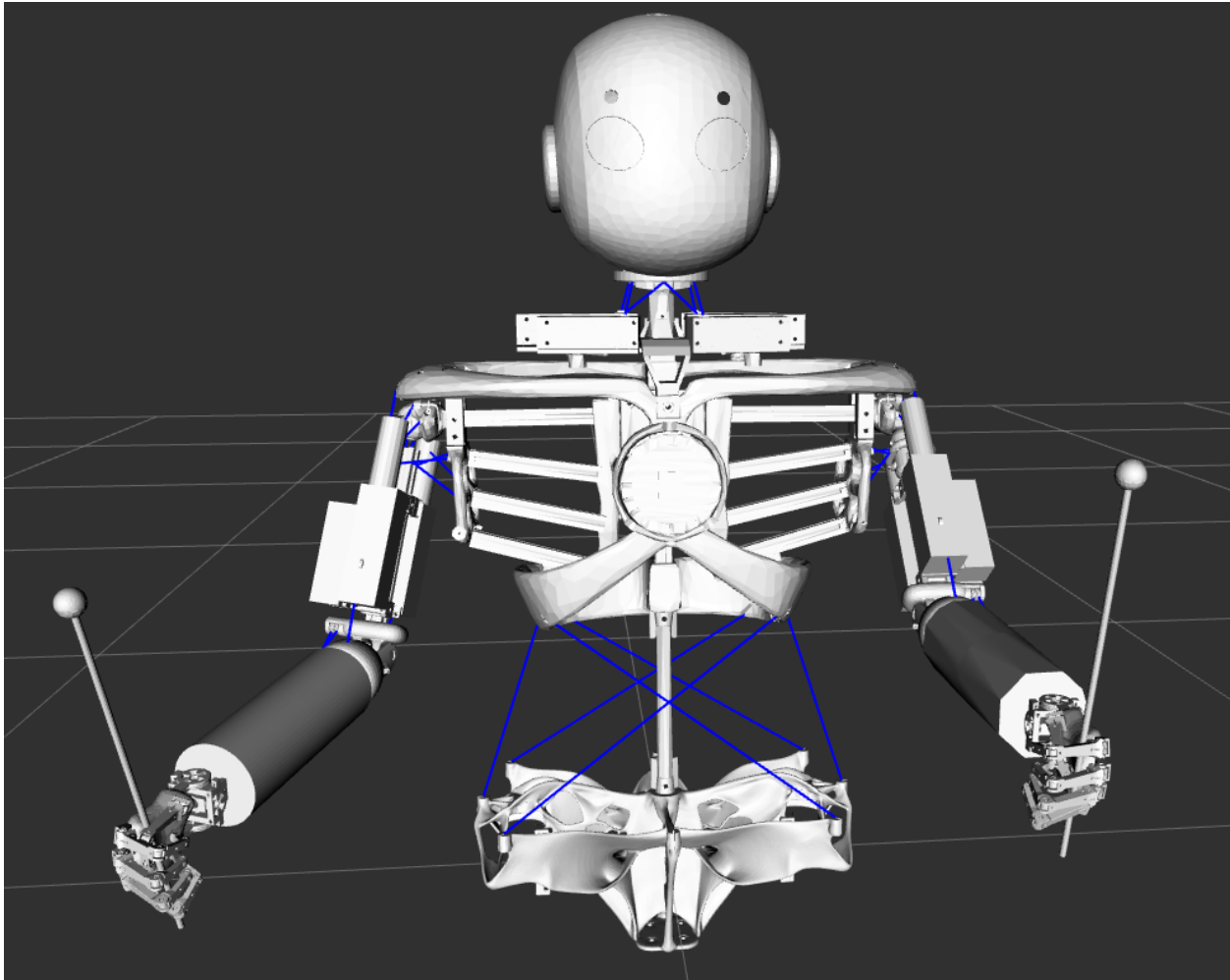
```
roslaunch kindyn robot.launch robot_name:=robo_upper_body start_controllers:='spine_
↳v0_axis0 spine_v0_axis1
spine_v0_axis2 spine_v1_axis0 spine_v1_axis1 spine_v1_axis2 spine_v2_axis0 spine_v2_
↳axis1 spine_v2_axis2
sphere_head_axis0 sphere_head_axis1 sphere_head_axis2 sphere_left_axis0 sphere_left_
↳axis1
```

(continues on next page)

(continued from previous page)

```
sphere_left_axis2 elbow_left_rot0 elbow_left_rot1 sphere_right_axis0 sphere_right_
↪axis1 sphere_right_axis2
elbow_right_rot0 elbow_right_rot1'
```

2.1.3 roboy_xylophone



Roboy 2.0 upper body with the 3-DoF head and his two 7-DoF arms holding xylophone playing sticks. Use the following command to launch it:

```
roslaunch kindyn robot.launch robot_name:=roboy_xylophone start_controllers:='sphere_
↪head_axis0 sphere_head_axis1
sphere_head_axis2 sphere_left_axis0 sphere_left_axis1 sphere_left_axis2 elbow_left_
↪rot0 elbow_left_rot1
sphere_right_axis0 sphere_right_axis1 sphere_right_axis2 elbow_right_rot0 elbow_right_
↪rot1 left_wrist_0
left_wrist_1 right_wrist_0 right_wrist_1 hip_joint left_stick_tip_joint right_stick_
↪tip_joint'
```

2.1.4 roboy_arcade_maschine

A retro arcade machine in the Roboy lab that has a 3-DoF Roboy head mounted on the top. Launch it with:

```
roslaunch kindyn robot.launch robot_name:=roboy_arcade_maschine start_controllers:=  
↪ 'sphere_axis0 sphere_axis1 sphere_axis2'
```

2.1.5 test_robot

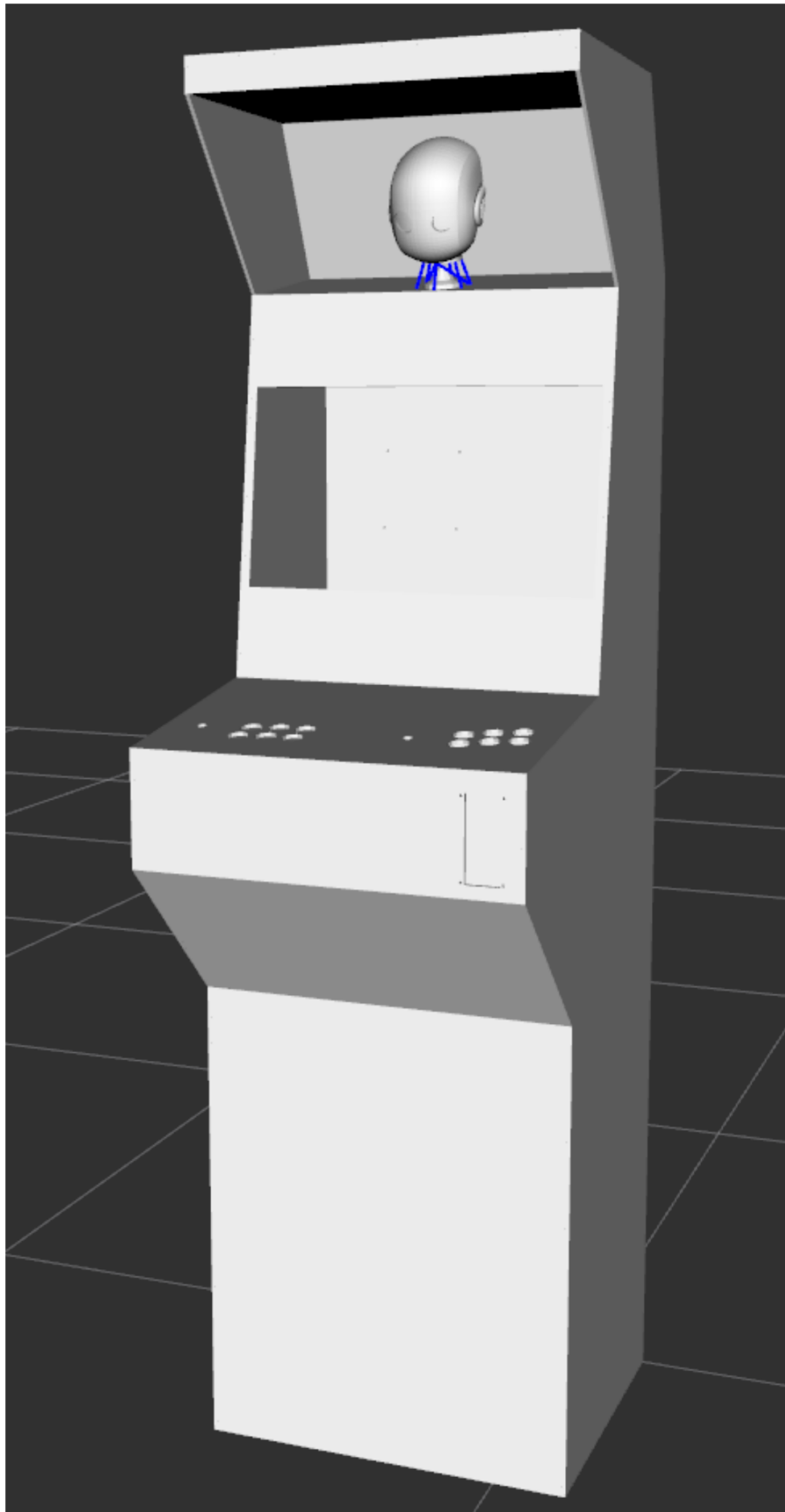
A robot with 6-DoF, 3 prismatic and 3 rotational, with 20 cables. You can launch it with the following command

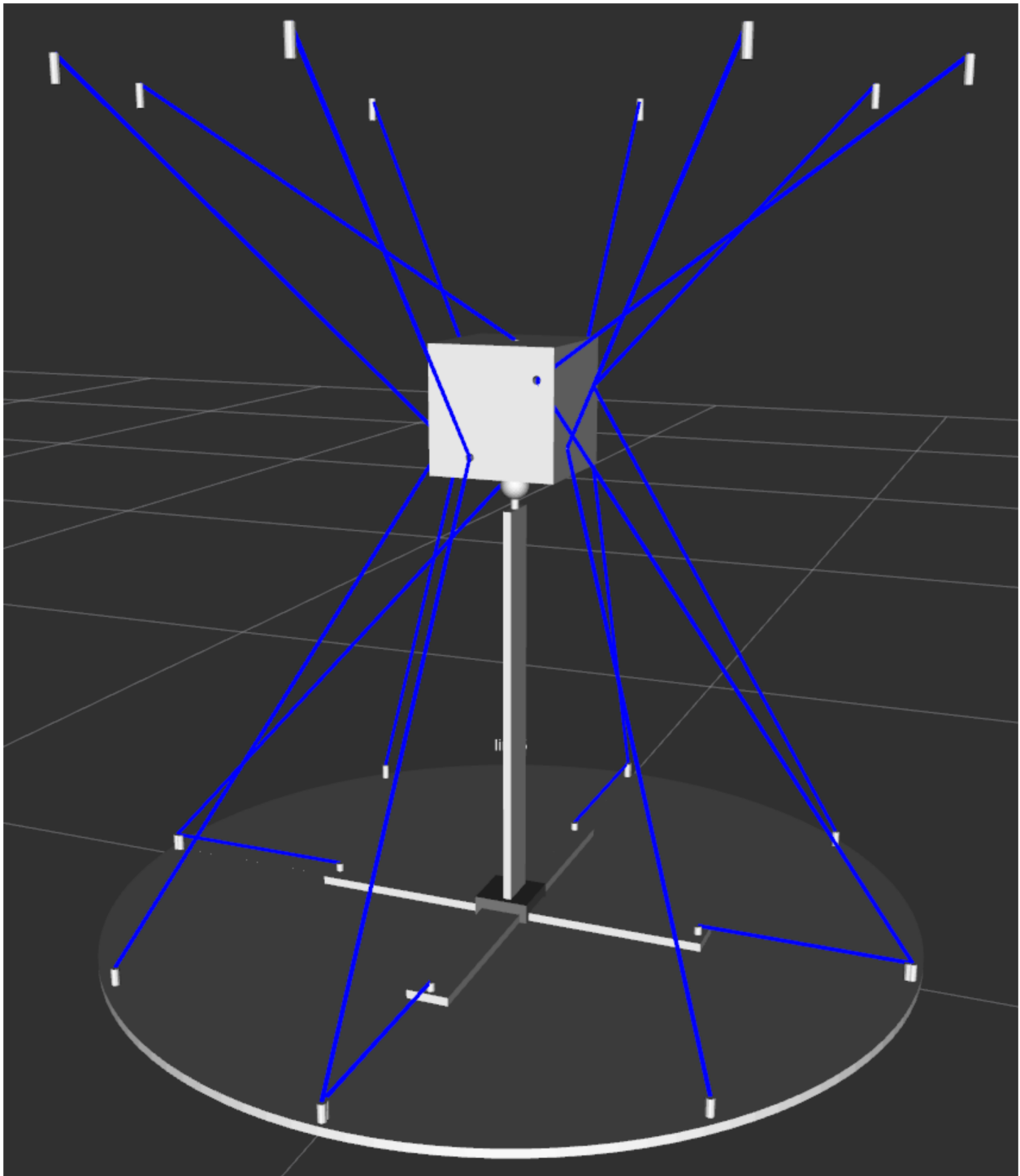
```
roslaunch kindyn robot.launch robot_name:=test_robot start_controllers:='joint0_  
↪ joint1 joint2 sphere_axis0 sphere_axis1 sphere_axis2'
```

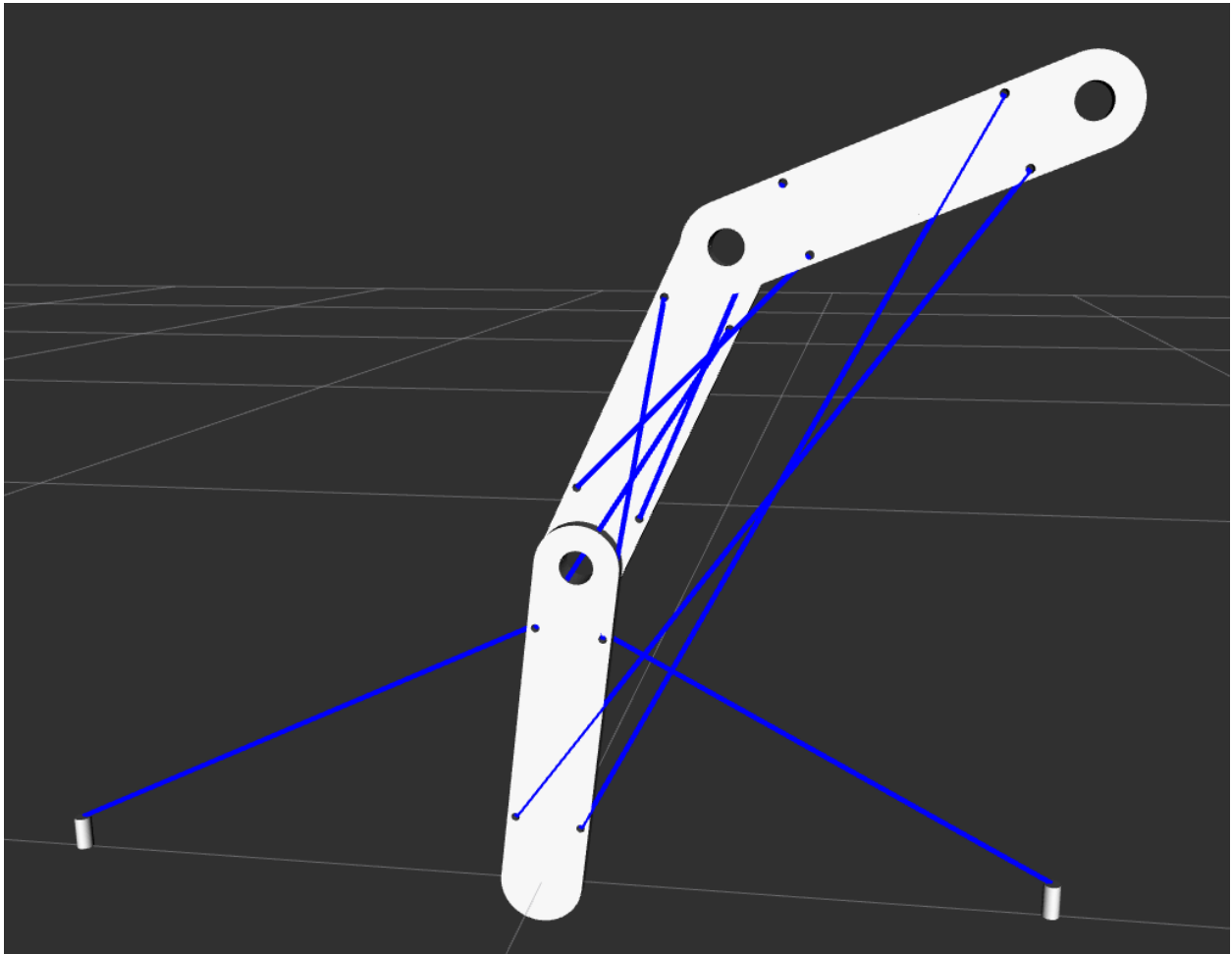
2.1.6 yatr

Yet another test robot, 3-DoF, 8 cables. You can launch it with the following command

```
roslaunch kindyn robot.launch robot_name:=yatr start_controllers:='joint0 joint1_  
↪ joint2'
```





WORKING WITH CASPR

CARDSflow supports visualization of robots in **CASPR** (*The Cable-robot Analysis and Simulation Platform for Research*), an open-source software platform developed in MATLAB for the analysis of arbitrary cable-robot models.

You need to install custom message support for matlab **MATLABROS**

3.1 Setting up CASPR

1. Follow the installation guide in **CASPR**
2. After having CASPR installed in MATLAB, note the IPs of your platforms running CARDSflow and CASPR
3. Set up the CARDSflow interface in CASPR by: (replace `<cardsflow_ip>`, `<caspr_ip>` with the corresponding IPs)

```
CARDSFlow_configuration.SetROSConfig('<cardsflow_ip>', '<caspr_ip>');
```

4. Build custom ROS messages in **roboy_communication** for MATLAB by following instructions [here](#)

3.2 Simple Visualization

3.2.1 Launching CARDSflow

1. Launch a simple visualizer in CARDSflow by:

```
roslaunch cardsflow_rviz simple_visualization.launch
```

2. Add the CARDSflow panel in RViz
3. At the Displays Sidebar set Globals Options > Fixed Frame to “world”
4. At the Displays Sidebar add the display types “rviz > Marker” and “rviz > TF”
5. File > Save Config

3.2.2 CASPR Simulations

1. Launch the GUI in CASPR by:

```
CASPR_GUI
```

2. Select `Example 2R planar XZ` from the `Model` pull down menu
3. Press the `To Rviz` button on the GUI and verify that the robot is visualized in RViz
4. Press the `Kinematics` button in `Simulators`
5. Run an inverse kinematics simulation by pressing the `Run` button
6. Press the `RViz` button to visualize the trajectory

CASPR GUI

File

CASPR GUI

Model Setup

Model
Example 2R planar XZ

Cable Set
basic_4_cables

Update Models Model Manager

Simulators

Kinematics

Dynamics

Control

Workspace

Model View

Example 2R planar XZ

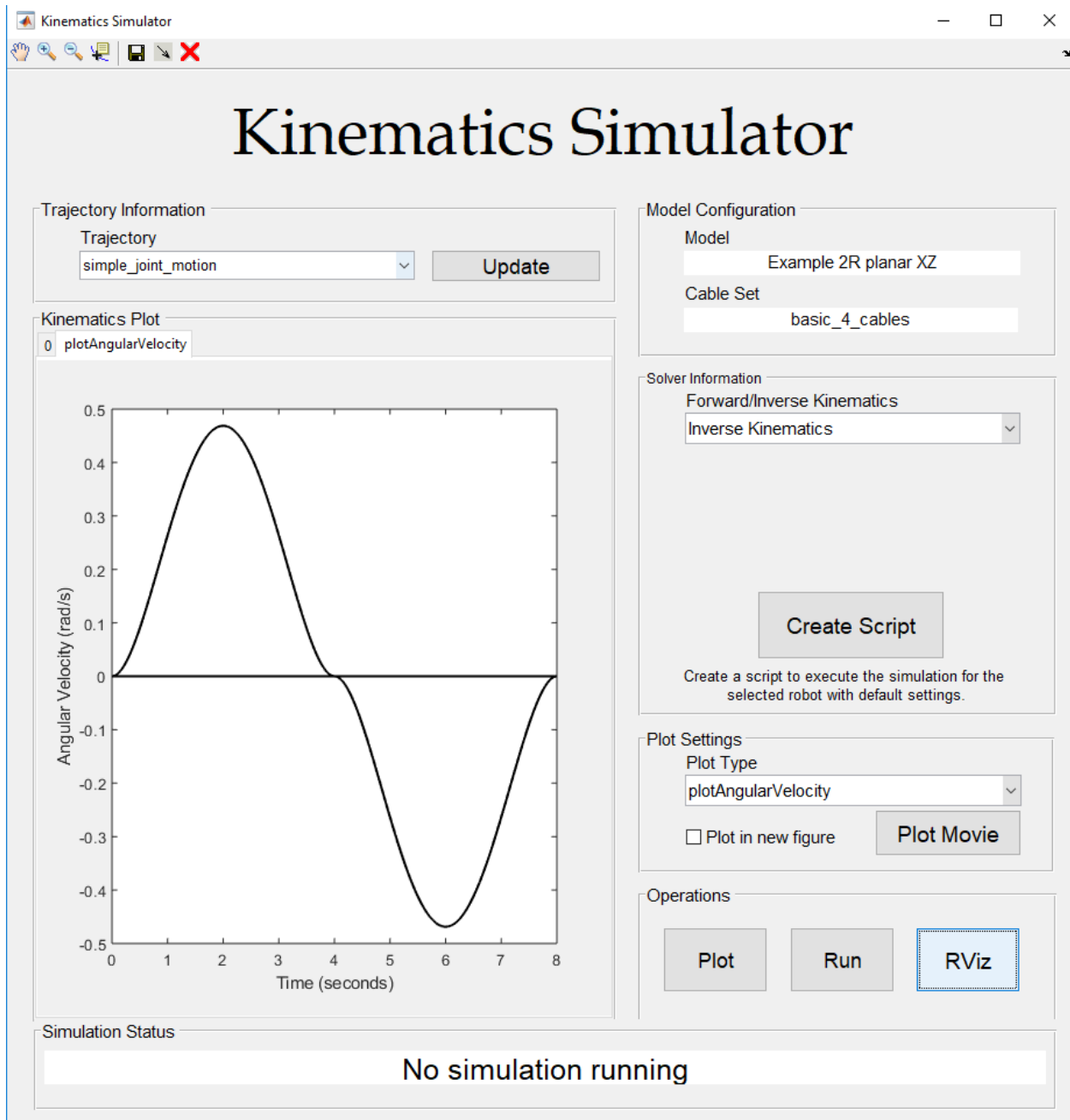
Model Position

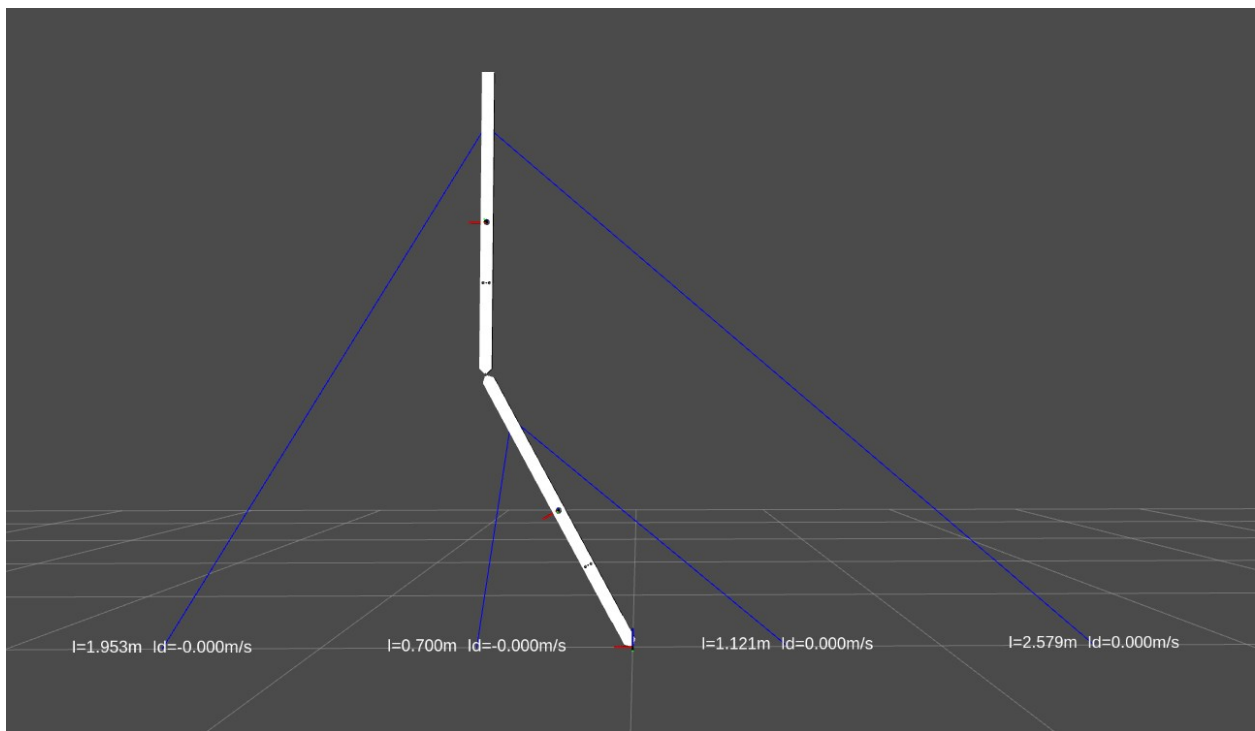
	q1	q2
1	0	0

Update Pose

To Console

To Rviz

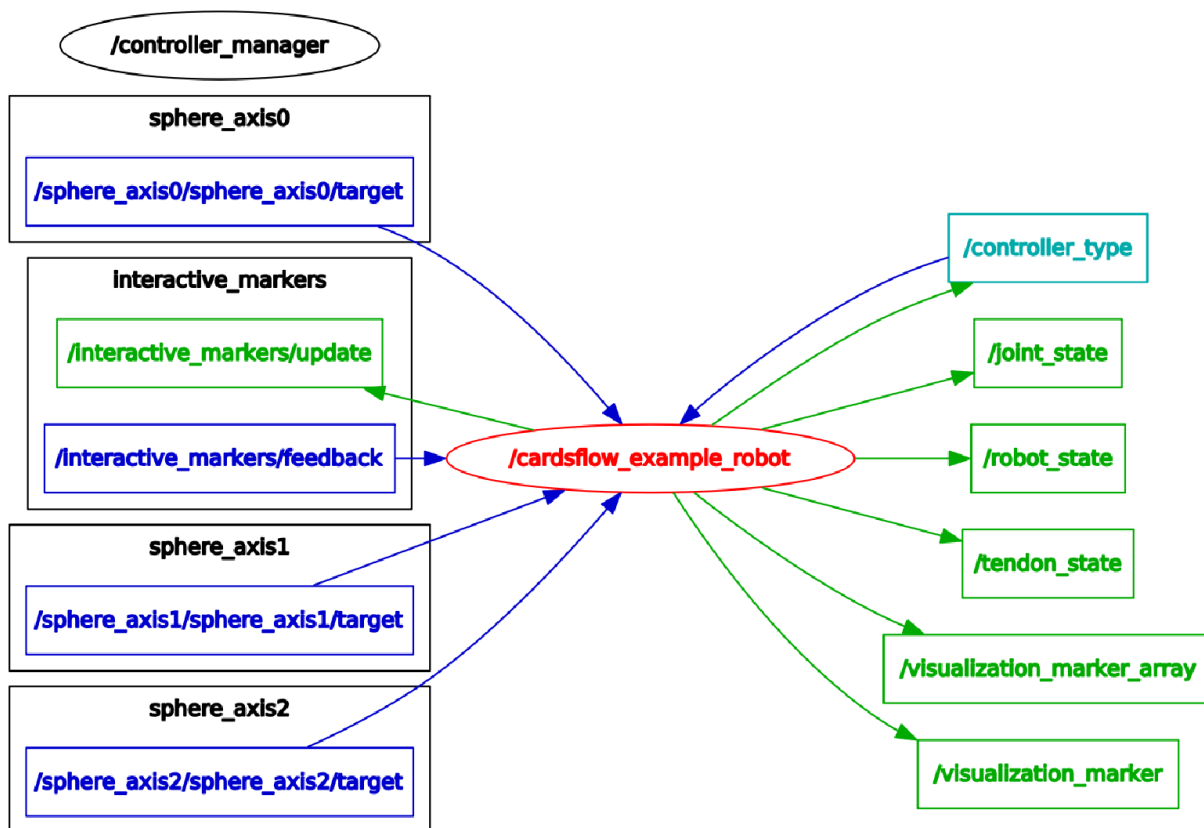




CONTEXT

TECHNICAL INTERFACES

The following rqt_graph shows an overview of the ROS message infrastructure for one of the example robots, the msj_platform.



Two ROS nodes are active, the `cardsflow_example_robot` and the `controller_manager`. The `cardsflow_example_robot` is our `msj_platform` doing all the kinematic calculations. The joint, robot and tendon states are published on the respective topics in green. On the left there are three joint target topics available in blue. The `controller_manager` spawned three joint controller and the respective controllers subscribe to these topics as their target setpoints. The `controller_type` topic is advertised by our `msj_platform` and the controllers publish their type on that topic once they are spawned by the `controller_manager`. The last topic is the `interactive_marker` topic which the `msj_platform` subscribes to. When a user drags the interactive markers in `rviz`, the inverse kinematics service is called and the robot will try to move to the requested position.

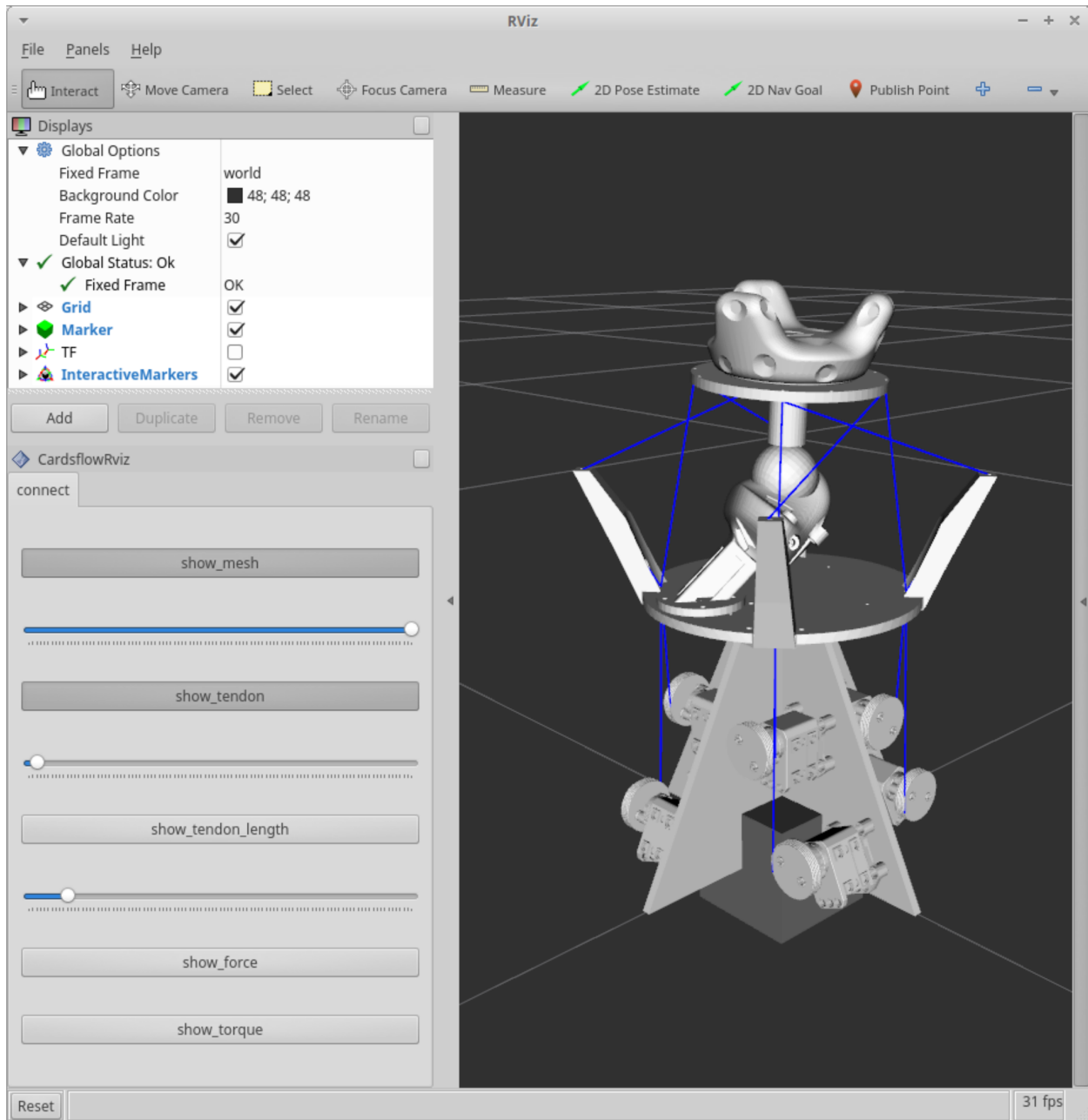
USER INTERFACES

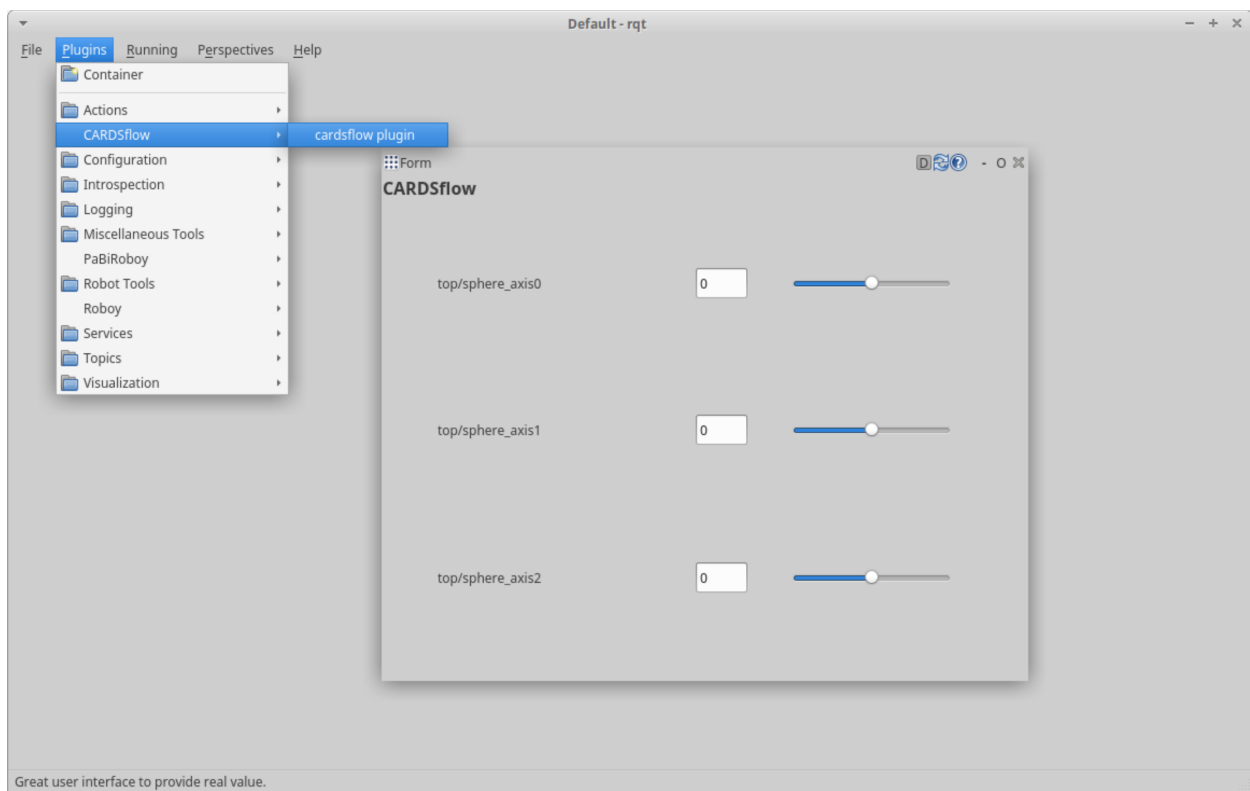
There are two user interfaces available to the user, a rviz plugin and a rqt plugin. The following figure shows the rviz plugin.

NOTE: In order for the mesh and tendons to be visible add the following plugins to rviz: 1) Marker 2) TF 3) InteractiveMarkers (you need to change the topic when you add the plugin) 4) you need to change the fixed frame in rviz from map to world

The buttons of the rviz plugin trigger visualization, while the slider change their appearance.

The rqt plugin can be seen in the figure below. The sliders can be used to change the setpoints for the endeffector joints.





struct Cable

Public Members

string **name**

vector<*ViaPointPtr*> **viaPoints**
name of the cable

class CableLengthController : public controller_interface::Controller<*hardware_interface::CardsflowCommandInterface*>

Public Functions

CableLengthController ()
Constructor.

bool **init** (*hardware_interface::CardsflowCommandInterface* *hw, ros::NodeHandle &n)
Initializes the controller.

Will be call by controller_manager when loading this controller

Return success

Parameters

- hw: pointer to the hardware interface
- n: the nodehandle

void **update** (**const** ros::Time &time, **const** ros::Duration &period)
Called regularly by controller manager.

The length change of the cables wrt to a PD controller on the joint target position is calculated.

Parameters

- time: current time
- period: period since last control

void **starting** (**const** ros::Time &time)
Called by controller manager when the controller is about to be started.

Parameters

- time: current time

void **stopping** (const ros::Time &time)

Called by controller manager when the controller is about to be stopped.

Parameters

- time: current time

void **JointPositionCommand** (const std_msgs::Float32ConstPtr &msg)

Joint position command callback for this joint.

Parameters

- msg: joint position target in radians

bool **setControllerParameters** (robo_control_msgs::SetControllerParameters::Request &req,
robo_control_msgs::SetControllerParameters::Response &res)

Controller Parameters service.

Return success

Parameters

- req: requested gains
- res: success

Private Members

double **Kp** = 100

double **Kd** = 0

double **p_error_last** = 0
PD gains.

ros::NodeHandle **nh**
last error

ros::Publisher **controller_state**
ROS nodehandle.

ros::ServiceServer **controller_parameter_srv**
publisher for controller state

boost::shared_ptr<ros::AsyncSpinner> **spinner**
service for controller parameters

hardware_interface::CardsflowHandle **joint**
ROS async spinner.

ros::Subscriber **joint_command**
cardsflow joint handle for access to joint/cable model state

string **joint_name**
joint command subscriber

int **joint_index**
name of the controlled joint

ros::Time **last_update**
index of the controlled joint in the robot model

```
class CableLengthVelocityController : public controller_interface::Controller<hardware_interface::CardsflowCommandInterface>
```

Public Functions

```
CableLengthVelocityController ()
```

Constructor.

```
bool init (hardware_interface::CardsflowCommandInterface *hw, ros::NodeHandle &n)
```

Initializes the controller.

Will be call by controller_manager when loading this controller

Return success

Parameters

- hw: pointer to the hardware interface
- n: the nodehandle

```
void update (const ros::Time &time, const ros::Duration &period)
```

Called regularly by controller manager.

The length change of the cables wrt to a PD controller on the joint target position is calculated.

Parameters

- time: current time
- period: period since last control

```
void starting (const ros::Time &time)
```

Called by controller manager when the controller is about to be started.

Parameters

- time: current time

```
void stopping (const ros::Time &time)
```

Called by controller manager when the controller is about to be stopped.

Parameters

- time: current time

```
void JointVelocityCommand (const std_msgs::Float32ConstPtr &msg)
```

Joint position command callback for this joint.

Parameters

- msg: joint position target in radians

```
bool setControllerParameters (roboym_control_msgs::SetControllerParameters::Request &req,  
                               roboym_control_msgs::SetControllerParameters::Response &res)
```

Controller Parameters service.

Return success

Parameters

- req: requested gains
- res: success

Private Members

double **Kp** = 0.1

double **Kd** = 0

double **p_error_last** = 0
PD gains.

ros::NodeHandle **nh**
last error

ros::Publisher **controller_state**
ROS nodehandle.

ros::ServiceServer **controller_parameter_srv**
publisher for controller state

boost::shared_ptr<ros::AsyncSpinner> **spinner**
service for controller parameters

hardware_interface::CardsflowHandle **joint**
ROS async spinner.

ros::Subscriber **joint_command**
cardsflow joint handle for access to joint/cable model state

string **joint_name**
joint command subscriber

int **joint_index**
name of the controlled joint

ros::Time **last_update**
index of the controlled joint in the robot model

class CardsflowCommandInterface : public HardwareResourceManager<*CardsflowHandle*, ClaimResources>

class CardsflowHandle : public *hardware_interface::CardsflowStateHandle*

Public Functions

CardsflowHandle ()

CardsflowHandle (const *CardsflowStateHandle* &js, double *joint_position_cmd, double *joint_velocity_cmd, double *joint_torque_cmd, VectorXd *motor_cmd)

Parameters

- js: joint state handle
- joint_position_cmd: joint position command
- joint_velocity_cmd: joint velocity command
- joint_torque_cmd: joint torque command
- motor_cmd: cable command

- `js`: This joint's state handle
- `cmd`: A pointer to the storage for this joint's output command

void **setMotorCommand** (*VectorXd command*)
Cable length command.

Parameters

- `command`:

double **getJointPositionCommand** () **const**
Returns the joint position command.

Return joint position command

double **getJointVelocityCommand** () **const**
Returns the joint velocity command.

Return joint velocity command

double **getJointTorqueCommand** () **const**
Returns the joint torque command.

Return joint torque command

void **setJointPositionCommand** (double *cmd*)
Sets the joint position command.

Parameters

- `cmd`: joint position command

void **setJointVelocityCommand** (double *cmd*)
Sets the joint velocity command.

Parameters

- `cmd`: joint velocity command

void **setJointTorqueCommand** (double *cmd*)
Sets the joint torque command.

Parameters

- `cmd`: joint torque command

Private Members

double **joint_position_cmd_*

double **joint_velocity_cmd_*

double **joint_torque_cmd_*

```
VectorXd *motor_cmd_  
    joint position/velocity/torque command
```

```
class CardsflowRviz : public Panel, public rviz_visualization
```

Public Functions

```
CardsflowRviz (QWidget *parent = 0)
```

```
~CardsflowRviz ()
```

```
void load (const rviz::Config &config)
```

Load all configuration data for this panel from the given Config object.

Parameters

- config: rviz config file

```
void save (rviz::Config config) const
```

Save all configuration data from this panel to the given Config object.

It is important here that you call *save()* on the parent class so the class id and panel name get saved.

Parameters

- config: rviz config file

Public Slots

```
void show_mesh ()
```

Toggles mesh visualization.

```
void show_collision ()
```

Toggles collision visualization.

```
void show_target ()
```

Toggles target visualization.

```
void show_tendon ()
```

Toggles tendon visualization.

```
void show_tendon_length ()
```

Toggles tendon length visualization.

```
void show_force ()
```

Toggles force visualization.

```
void show_torque ()
```

Toggles torque visualization.

```
void visualizePose ()
```

Visualization of Pose.

```
void visualizeCollision ()
```

Visualization of Collision Shapes.

void **visualizePoseTarget** ()
Visualization of Pose Target.

void **visualizeTendon** ()
Visualization of *Tendon*.

void **visualizeTendonTarget** ()
Visualization of *Tendon* Target.

void **visualizeTorque** ()
Visualization of Torque.

void **visualizeTorqueTarget** ()
Visualization of Torque Target.

Signals

void **visualizePoseSignal** ()

void **visualizeCollisionSignal** ()

void **visualizeTargetSignal** ()

void **visualizePoseTargetSignal** ()

void **visualizeTendonSignal** ()

void **visualizeTendonTargetSignal** ()

void **visualizeTorqueSignal** ()

void **visualizeTorqueTargetSignal** ()

Private Functions

void **RobotState** (const geometry_msgs::PoseStampedConstPtr &msg)
Callback to robot state messages.

Parameters

- msg:

void **RobotStateTarget** (const geometry_msgs::PoseStampedConstPtr &msg)
Callback to robot state target messages.

Parameters

- msg:

void **TendonState** (const roboy_simulation_msgs::TendonConstPtr &msg)
Callback for *Tendon* state messages.

Parameters

- msg:

void **TendonStateTarget** (**const** roboy_simulation_msgs::TendonConstPtr &*msg*)
Callback for *Tendon* state target messages.

Parameters

- *msg*:

void **JointState** (**const** roboy_simulation_msgs::JointStateConstPtr &*msg*)
Callback for Joint state messages.

Parameters

- *msg*:

void **JointStateTarget** (**const** roboy_simulation_msgs::JointStateConstPtr &*msg*)
Callback for Joint state target messages.

Parameters

- *msg*:

Private Members

```
ros::NodeHandlePtr nh  
boost::shared_ptr<ros::AsyncSpinner> spinner  
ros::Subscriber robot_state  
ros::Subscriber tendon_state  
ros::Subscriber joint_state  
ros::Subscriber robot_state_target  
ros::Subscriber tendon_state_target  
ros::Subscriber joint_state_target  
tf::TransformListener tf_listener  
tf::TransformBroadcaster tf_broadcaster  
map<string, geometry_msgs::Pose> pose  
map<string, geometry_msgs::Pose> pose_target  
map<string, Tendon> tendon  
map<string, Tendon> tendon_target  
map<string, geometry_msgs::Vector3> joint_origin  
map<string, geometry_msgs::Vector3> joint_origin_target  
map<string, geometry_msgs::Vector3> joint_axis  
map<string, geometry_msgs::Vector3> joint_axis_target  
map<string, double> torque  
map<string, double> torque_target  
bool visualize_pose = true
```

```

bool visualize_collisions = true
bool visualize_targets = true
bool visualize_tendon = true
bool visualize_tendon_length = true
bool visualize_force = false
bool visualize_torque = false
bool visualize_tendon_target = true
bool visualize_tendon_length_target = true
bool visualize_force_target = false
bool visualize_torque_target = false
QPushButton *show_mesh_button
QPushButton *show_collision_button
QPushButton *show_target_button
QPushButton *show_tendon_button
QPushButton *show_force_button
QPushButton *show_torque_button
QPushButton *show_tendon_length_button
QSlider *mesh_transparency
QSlider *cable_thickness
QSlider *tendon_length_text_size
string model_name
class CardsflowStateHandle
    Subclassed by hardware_interface::CardsflowHandle

```

Public Functions

CardsflowStateHandle ()

CardsflowStateHandle (const *std::string* &name, int joint_index, const double *pos, const double *vel, const double *acc, const MatrixXd *L, const MatrixXd *M, const VectorXd *CG)

Parameters

- name: The name of the joint
- joint_index: index of the joint in the robot model
- pos: joint position
- vel: joint velocity
- acc: joint acceleration
- L: pointer to the L matrix
- M: pointer to the mass matrix

- CG: pointer to the Coriolis+Gravity vector
- name: The name of the joint
- joint_index: index of the joint in the robot model
- pos: A pointer to the storage for this joint's position
- vel: A pointer to the storage for this joint's velocity

`std::string getName () const`

Returns the joint name.

Return joint name

`int getJointIndex () const`

Returns the joint index in the robot model.

Return joint index

`double getPosition () const`

Returns the current joint position.

Return joint position

`double getVelocity () const`

Returns the current joint velocity.

Return joint velocity

`double getAcceleration () const`

Returns the current joint acceleration.

Return joint acceleration

`MatrixXd getL () const`

Returns the cable model L matrix.

Return L

`MatrixXd getM () const`

Returns the robot Mass matrix.

Return M

`VectorXd getCG () const`

Returns the robot Coriolis+Gravity vector.

Return CG

Private Members

```

std::string name_
int joint_index_
    joint name
const double *pos_
    joint index
const double *vel_
    joint position
const double *acc_
    joint velocity
const VectorXd *CG_
    joint acceleration
const MatrixXd *L_
    Coriolis+Gravity vector.
const MatrixXd *M_

```

```

class CardsflowStateInterface : public HardwareResourceManager<CardsflowStateHandle>
struct COLOR

```

Public Functions

```

COLOR (float r, float g, float b, float a)
void randColor ()

```

Public Members

```

float r
float g
float b
float a

```

```

struct EigenRobotAcceleration

```

Public Functions

```

void resize (int nrOfInternalDOFs)
void random ()

```

Public Members

```

Eigen::Matrix<double, 6, 1> baseAcc
Eigen::VectorXd jointAcc

```

struct EigenRobotState

A tutorial on how to use the KinDynComputations class with Eigen data structures.

CopyPolicy: Released under the terms of LGPL 2.0+ or later Struct containing the floating robot state using Eigen data structures.

Author Silvio Traversaro

Public Functions

void **resize** (int *nrOfInternalDOFs*)

void **random** ()

Public Members

Eigen::Matrix4d **world_H_base**

Eigen::VectorXd **jointPos**

Eigen::Matrix<double, 6, 1> **baseVel**

Eigen::VectorXd **jointVel**

Eigen::Vector3d **gravity**

class ForcePositionController : **public** controller_interface::Controller<*hardware_interface::CardsflowCommandInterface*>

Public Functions

ForcePositionController ()

Constructor.

bool **init** (*hardware_interface::CardsflowCommandInterface* *hw, ros::NodeHandle &n)

Initializes the controller.

Will be call by controller_manager when loading this controller

Return success

Parameters

- hw: pointer to the hardware interface
- n: the nodehandle

void **update** (const ros::Time &time, const ros::Duration &period)

Called regularly by controller manager.

The torque for a joint wrt to a PD controller on the joint target position is calculated.

Parameters

- time: current time
- period: period since last control

void **starting** (const ros::Time &time)

Called by controller manager when the controller is about to be started.

Parameters

- `time`: current time

void **stopping** (**const** `ros::Time &time`)

Called by controller manager when the controller is about to be stopped.

Parameters

- `time`: current time

void **JointPositionCommand** (**const** `std_msgs::Float32ConstPtr &msg`)

Joint position command callback for this joint.

Parameters

- `msg`: joint position target in radians

bool **setControllerParameters** (`robo_control_msgs::SetControllerParameters::Request &req`,
`robo_control_msgs::SetControllerParameters::Response &res`)

Controller Parameters service.

Return success

Parameters

- `req`: requested gains
- `res`: success

Private Members

double **q_target** = 0

double **p_error_prev** = 0
 joint position target

double **Kp** = 1

double **Kd** = 0

`ros::NodeHandle` **nh**
 PD gains.

`ros::Publisher` **controller_state**
 ROS nodehandle.

`ros::ServiceServer` **controller_parameter_srv**
 publisher for controller state

`boost::shared_ptr<ros::AsyncSpinner>` **spinner**
 service for controller parameters

`hardware_interface::CardsflowHandle` **joint**

`ros::Subscriber` **joint_command**
 cardsflow joint handle for access to joint/cable model state

string **joint_name**
 joint command subscriber

```
int joint_index  
    name of the controlled joint
```

```
template<typename _Scalar, int NX = Dynamic, int NY = Dynamic>  
struct Functor
```

Public Types

```
enum [anonymous]  
    Values:  
        InputsAtCompileTime = NX  
        ValuesAtCompileTime = NY  
typedef _Scalar Scalar  
typedef Eigen::Matrix<Scalar, InputsAtCompileTime, 1> InputType  
typedef Eigen::Matrix<Scalar, ValuesAtCompileTime, 1> ValueType  
typedef Eigen::Matrix<Scalar, ValuesAtCompileTime, InputsAtCompileTime> JacobianType
```

Public Functions

```
Functor ()  
Functor (int inputs, int values)  
int inputs () const  
int values () const
```

Public Members

```
const int m_inputs  
const int m_values  
struct iDynTreeRobotAcceleration
```

Public Functions

```
void resize (int nrOfInternalDOFs)
```

Public Members

```
iDynTree::Vector6 baseAcc  
iDynTree::VectorDynSize jointAcc  
struct iDynTreeRobotState  
    the base link of each endeffector
```


Public Functions

void **resize** (int *nrOfInternalDOFs*)

Public Members

iDynTree::Transform **world_H_base**

iDynTree::VectorDynSize **jointPos**

iDynTree::Twist **baseVel**

iDynTree::VectorDynSize **jointVel**

iDynTree::Vector3 **gravity**

struct iDynTreeRobotState

Struct containing the floating robot state using iDynTree data structures.

For the semantics of this structures, see KinDynComputation::setRobotState method.

Public Functions

void **resize** (int *nrOfInternalDOFs*)

Public Members

iDynTree::Transform **world_H_base**

iDynTree::VectorDynSize **jointPos**

iDynTree::Twist **baseVel**

iDynTree::VectorDynSize **jointVel**

iDynTree::Vector3 **gravity**

struct iDynTreeRobotState

the base link of each endeffector

Public Functions

void **resize** (int *nrOfInternalDOFs*)

Public Members

iDynTree::Transform **world_H_base**

iDynTree::VectorDynSize **jointPos**

iDynTree::Twist **baseVel**

iDynTree::VectorDynSize **jointVel**

iDynTree::Vector3 **gravity**

class MeasureExecutionTime

Public Functions

MeasureExecutionTime (string *logfile_name*)

~MeasureExecutionTime ()

void **start** ()

double **stop** (const char **name*)

Private Members

time_point<Clock> **start_point**

time_point<Clock> **stop_point**

ofstream **log_file**

class MotorConfig

Public Functions

MotorConfig ()

bool **readConfig** (const string &*filepath*)
Reads a yaml motor config file.

Return success

Parameters

- *filepath*: to config

bool **writeConfig** (const string &*filepath*)
Writes a yaml motor config file.

Return success

Parameters

- *filepath*:

bool **fileExists** (const string &*filepath*)
Checks if a file exists.

Return exists

Parameters

- *filepath*:

double **displacement2force** (double *displacement*, int *fpga*, int *motor*)
Transforms displacement to force using loaded coefficients.

Return force

Parameters

- displacement:
- fpga: for this fpga
- motor: motor id (as listed in read config)

double **force2displacement** (double *force*, int *fpga*, int *motor*)
Transforms force to displacement using loaded coefficients.

Return force

Parameters

- displacement:
- fpga: for this fpga
- motor: motor id (as listed in read config)

Public Members

vector<vector<vector<float>>> **coeffs_displacement2force**

vector<vector<vector<float>>> **coeffs_force2displacement**

class MsjPlatform: public *cardsflow::kindyn::Robot*, public *cardsflow::kindyn::Robot*

Public Functions

MsjPlatform (string *urdf*, string *cardsflow_xml*)
Constructor.

Parameters

- urdf: path to urdf
- cardsflow_xml: path to cardsflow xml

void **read** ()
Updates the robot model and if we do not use gazebo for simulation, we integrate using the forwardKinematics function with a small step length.

void **write** ()
Sends motor commands to the real robot.

bool **GymStepService** (roboy_simulation_msgs::GymStep::Request *&req*,
roboy_simulation_msgs::GymStep::Response *&res*)

bool **GymResetService** (roboy_simulation_msgs::GymReset::Request *&req*,
roboy_simulation_msgs::GymReset::Response *&res*)

MsjPlatform (string *urdf*, string *cardsflow_xml*)
Constructor.

Parameters

- urdf: path to urdf
- cardsflow_xml: path to cardsflow xml

void **read** ()
Updates the robot model and if we do not use gazebo for simulation, we integrate using the forwardKinematics function with a small step length.

void **write** ()
Sends motor commands to the real robot.

int **pnpoly** (vector<double> *limits_x*, vector<double> *limits_y*, double *testx*, double *testy*)

void **randomPose** ()

Public Members

bool **external_robot_state**

ros::NodeHandlePtr **nh**
indicates if we get the robot state externally

ros::Publisher **motor_command**
ROS nodehandle.

ros::ServiceServer **gym_step**
motor command publisher

ros::ServiceServer **gym_reset**

double **l_offset**
motor command publisher

float **Kp** = 0.001

ros::Publisher **sphere_axis0**

ros::Publisher **sphere_axis1**

ros::Publisher **sphere_axis2**

vector<double> **limits**[3]

boost::shared_ptr<std::thread> **pose_thread**

class Rickshaw_pedaling : public *cardsflow::kindyn::Robot*

Public Functions

Rickshaw_pedaling (string *urdf*, string *cardsflow_xml*)
Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

void **read** ()
Updates the robot model and integrates the robot model using the forwardKinematics function with a small step length.

void **write** ()
Sends motor commands to the real robot.

Public Members

```

ros::NodeHandlePtr nh
ros::Publisher motor_command
    ROS nodehandle.
ros::ServiceClient motor_config
    motor command publisher
ros::ServiceClient sphere_left_axis0_params
ros::ServiceClient sphere_left_axis1_params
ros::ServiceClient sphere_left_axis2_params
map<string, ros::ServiceClient> motor_control_mode
vector<string> endeffectors = {"spine_right"}
map<string, vector<string>> endeffector_jointnames
bool external_robot_state
map<string, vector<short unsigned int> > Rickshaw_pedaling::motors= {           {"head", {0,0,0,0}}
    indicates if we get the robot state externally
map<string, vector<short unsigned int> > Rickshaw_pedaling::sim_motors= {           {"", {0,0,0,0}}
map<string, vector<double> > Rickshaw_pedaling::l_offset= {           {"head", {0,0,0,0}}
class Rikshaw : public cardsflow::kindyn::Robot, public cardsflow::kindyn::Robot

```

Public Functions

Rikshaw (string *urdf*, string *cardsflow_xml*)
 Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

void **read** ()
 Updates the robot model and integrates the robot model using the forwardKinematics function with a small step length.

void **write** ()
 Sends motor commands to the real robot.

Rikshaw (string *urdf*, string *cardsflow_xml*)
 Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

void **MotorStatus** (const roboymiddleware_msgs::MotorStatus::ConstPtr &*msg*)

void **read** ()
 Updates the robot model and integrates the robot model using the forwardKinematics function with a small step length.

void **write** ()
 Sends motor commands to the real robot.

Public Members

ros::NodeHandlePtr **nh**

ros::Publisher **motor_command**
 ROS nodehandle.

ros::ServiceClient **motor_config**
 motor command publisher

ros::ServiceClient **sphere_left_axis0_params**

ros::ServiceClient **sphere_left_axis1_params**

ros::ServiceClient **sphere_left_axis2_params**

map<string, ros::ServiceClient> **motor_control_mode**

vector<string> **endeffectors** = {"spine_right"}

map<string, vector<string>> **endeffector_jointnames**

bool **external_robot_state**

map< string, vector< short unsigned int > > **Rikshaw::motors**= { {"head", {9,1
 indicates if we get the robot state externally

map< string, vector< short unsigned int > > **Rikshaw::sim_motors**= { {"head",

map< string, vector< double > > **Rikshaw::l_offset**= { {"head", {0,0,0,0,0,0}}

ros::Publisher **joint_hip_left**
 motor command publisher

ros::Publisher **joint_hip_right**

ros::Publisher **joint_knee_left**

ros::Publisher **joint_knee_right**

ros::Publisher **joint_foot_left**

ros::Publisher **joint_foot_right**

ros::Publisher **left_shoulder_axis0**

ros::Publisher **left_shoulder_axis1**

ros::Publisher **left_shoulder_axis2**

ros::Publisher **elbow_left_rot0**

ros::Publisher **elbow_left_rot1**

ros::Publisher **left_wrist_0**

ros::Publisher **left_wrist_1**

ros::Publisher **right_shoulder_axis0**

```

ros::Publisher right_shoulder_axis1
ros::Publisher right_shoulder_axis2
ros::Publisher elbow_right_rot0
ros::Publisher elbow_right_rot1
ros::Publisher right_wrist_0
ros::Publisher right_wrist_1
ros::Subscriber motor_status
map<string, ros::ServiceClient> motor_config
bool status_received = false
    indicates if we get the robot state externally
std_msgs::Float32 roll
std_msgs::Float32 pitch
std_msgs::Float32 yaw
float err_x = 0
float error_y = 0
float pitch_max = 0.33
float pitch_min = -0.50
float yaw_min = -0.50
float yaw_max = 0
map<string, vector<double> > Rikshaw::encoder_offset= { {"head", {0,0,0,0,0,0,0
class Robot : public rviz_visualization
    Subclassed by Roboy2, RoboyIcecream

```

Public Functions

Robot ()
 Constructor.

~Robot ()
 Destructor.

void **init** (string *urdf_file_path*, string *viapoints_file_path*, vector<string> *joint_names*)
 initializes everything, call before use!

Parameters

- *urdf_file_path*: path to robot urdf
- *viapoints_file_path*: path to viapoints xml
- *joint_names*: a vector of joint_names to be considered from the model

void **update** ()
 Updates the model.

virtual void read ()

This is the read function and should implement reading the state of your robot.

virtual void write ()

This is the write function and should implement writing commands to your robot.

Public Members

iDynTree::KinDynComputations **kinDynComp**

iDynTree::KinDynComputations **kinDynCompTarget**

size_t **number_of_dofs** = 0

the full robot model

vector<string> **endeffectors**

number of degrees of freedom of the whole robot

map<string, size_t> **endeffector_index**

names of the endeffectors

vector<size_t> **endeffector_number_of_dofs**

vector<size_t> **endeffector_dof_offset**

size_t **number_of_joints** = 0

number of degrees of freedom of each endeffector

size_t **number_of_cables** = 0

number of joints of the whole robot

size_t **number_of_links** = 0

number of cables, ie muscles of the whole robot

Matrix4d **world_H_base**

number of links of the whole robot

vector<Matrix4d> **world_to_link_transform**

floating base 6-DoF pose

vector<Matrix4d> **link_to_world_transform**

vector<Matrix4d> **frame_transform**

*Eigen::*Matrix<double, 6, 1> **baseVel**

Vector3d **gravity**

the velocity of the floating base

MatrixXd **M**

gravity vector (default: (0,0,-9.81))

VectorXd **CG**

The Mass matrix of the robot.

VectorXd **q**

The Coriolis+Gravity term of the robot.

VectorXd **qd**

VectorXd **qdd**

VectorXd **q_min**

joint position, velocity, acceleration

VectorXd **q_max**
 VectorXd **q_target**
 joint limits
 VectorXd **qd_target**
 VectorXd **qdd_target**
 VectorXd **q_target_prev**
 joint position, velocity, acceleration targets
 VectorXd **qd_target_prev**
 VectorXd **qdd_target_prev**
 VectorXd **l_int**
 joint position, velocity, acceleration targets
 VectorXd **l**
 VectorXd **l_target**
 vector<VectorXd> **Ld**
 tendon length and length change
 VectorXd **torques**
 VectorXd **cable_forces**
 joint torques
 vector<VectorXd> **ld**
 the cable forces in Newton
 MatrixXd **L**
 tendon length changes for each controller
 MatrixXd **L_t**
 MatrixXd **S**
 L and $-L^T$.
 MatrixXd **P**
 MatrixXd **V**
 MatrixXd **W**
 vector<vector<pair<ViaPointPtr, ViaPointPtr>>> **segments**
 matrices of cable model
 bool **external_robot_state**
 cable segments
 bool **simulated** = false
 indicates if we get the robot state externally
 bool **initialized** = false
 indicates if the robots is simulated or hardware is used

Protected Types

typedef boost::array<double, 2> **state_type**
 odeint integration time

Protected Attributes

iDynTree::FreeFloatingGeneralizedTorques **bias**
iDynTree::MatrixDynSize **Mass**
 Coriolis+Gravity term.
 bool **torque_position_controller_active** = false
 Mass matrix.
 bool **force_position_controller_active** = false
 bool **cable_length_controller_active** = false
 VectorXd **qdd_torque_control**
 VectorXd **qdd_force_control**
 vector<Cable> **cables**
 vector<VectorXd> **joint_axis**
 all cables of the robot
 vector<string> **link_names**
 joint axis of each joint
 vector<string> **joint_names**
 map<string, int> **link_index**
 link and joint names of the robot
 map<string, int> **joint_index**
 vector<ros::Publisher> **joint_command_pub**
 link and joint indices of the robot
 vector<int> **controller_type**
 double **integration_time** = 0
 currently active controller type
 vector<*state_type*> **joint_state**
 second order dynamics integration
 vector<*state_type*> **motor_state**
 bool **first_time_solving** = true
 joint and cable states
 int **nWSR** = 1000
 VectorXd **f_min**
 qp working sets
 VectorXd **f_max**
 int **qp_print_level** = PL_NONE
 SQProblem **qp_solver**
 qpases print level
 real_t ***H**
 qpases quadratic problem solver
 real_t ***g**
 real_t ***A**

```

real_t *lb
real_t *ub
real_t *b
real_t *FOpt
ros::Time last_visualization
    quadratic problem variables
Eigen::IOFormat fmt
    timestamp for visualization at reasonable intervals
hardware_interface::JointStateInterface joint_state_interface
    formator for terminal printouts
hardware_interface::EffortJointInterface joint_command_interface
    ros control joint state interface
hardware_interface::CardsflowStateInterface cardsflow_state_interface
    ros control joint command interface
hardware_interface::CardsflowCommandInterface cardsflow_command_interface
    cardsflow state interface
bool first_update = true
    cardsflow command interface

```

Private Functions

void **MoveEndEffector** (const roboy_control_msgs::MoveEndEffectorGoalConstPtr &goal)
 Move endeffector action server.

Parameters

- goal:

bool **parseViapoints** (const string &viapoints_file_path, vector<Cable> &cables)
 parses the cardsflow xml file for viapoint definitions

Return success

Parameters

- viapoints_file_path: path to the cardsflow.xml file
- cables: will be filled with the parsed viapoints of the defined cables

bool **ForwardKinematicsService** (roboy_middleware_msgs::ForwardKinematics::Request &req,
 roboy_middleware_msgs::ForwardKinematics::Response
 &res)
 Forward kinematic service for endeffectors.

Return success

Parameters

- req: endeffector, requested joint angles
- res: 3d resulting position of endeffector

bool **InverseKinematicsService** (roby_middleware_msgs::InverseKinematics::Request &req,
roby_middleware_msgs::InverseKinematics::Response &res)
Inverse kinematic service for endeffectors.

Return success

Parameters

- req: endeffector and ik type
- res: joint configuration solution

bool **InverseKinematicsMultipleFramesService** (roby_middleware_msgs::InverseKinematicsMultipleFrames::Request &req,
roby_middleware_msgs::InverseKinematicsMultipleFrames::Response &res)

void **InteractiveMarkerFeedback** (const visualization_msgs::InteractiveMarkerFeedbackConstPtr &msg)

Callback for Interactive Marker Feedback of endeffectors.

When the Interactive Marker is released in rviz, the IK routine is called and the solution directly applied to the robot q_target angles

Parameters

- msg: Interactive Marker Feedback message

void **JointState** (const sensor_msgs::JointStateConstPtr &msg)

Callback for the joint state of the robot.

This can come from gazebo, the real robot or else where.

Parameters

- msg: message containing joint_name/angle information

void **JointTarget** (const sensor_msgs::JointStateConstPtr &msg)

Callback for the joint target of the robot.

Parameters

- msg: message containing joint_name/angle information

void **FloatingBase** (const geometry_msgs::PoseConstPtr &msg)

Callback for the floating base world pose.

This can come from gazebo, the real robot or else where.

Parameters

- msg: message containing the 6 DoF pose of the floating base

VectorXd **resolve_function** (MatrixXd &A_eq, VectorXd &b_eq, VectorXd &f_min, VectorXd &f_max)

Private Members

Matrix3d ***link_to_link_transform**

ros::NodeHandlePtr **nh**

```

boost::shared_ptr<ros::AsyncSpinner> spinner
    ROS node handle.

ros::Publisher robot_state_pub
    async ROS spinner

ros::Publisher tendon_state_pub

ros::Publisher joint_state_pub

ros::Publisher cardsflow_joint_states_pub

ros::Publisher robot_state_target_pub
    ROS robot pose and tendon publisher.

ros::Publisher tendon_state_target_pub

ros::Publisher joint_state_target_pub

ros::Subscriber controller_type_sub
    target publisher

ros::Subscriber joint_state_sub

ros::Subscriber joint_target_sub

ros::Subscriber floating_base_sub

ros::Subscriber interactive_marker_sub

ros::ServiceServer ik_srv
    ROS subscribers.

ros::ServiceServer ik_two_frames_srv

ros::ServiceServer fk_srv

map<string, boost::shared_ptr<actionlib::SimpleActionServer<roboy_control_msgs::MoveEndEffectorAction>>> moveEndEff

map<string, iDynTree::KinDynComputations> ik_models

map<string, iDynTree::InverseKinematics> ik
    the robot models for each endeffector

map<string, string> ik_base_link
    the ik for each endeffector

struct cardsflow::vrpuppet::Robot::iDynTreeRobotState robotstate

class Robot : public cardsflow::kindyn::Robot

```

Public Functions

Robot (string *urdf*, string *cardsflow_xml*)

void **read** ()

This is the read function and should implement reading the state of your robot.

void **write** ()

This is the write function and should implement writing commands to your robot.

Private Members

ros::NodeHandlePtr nh

bool external_robot_state

class Robot : public RobotHW, public rviz_visualization

Subclassed by *MsjPlatform*, *MsjPlatform*, *Rickshaw_pedaling*, *Rikshaw*, *Rikshaw*, *Robot*, *RoboyArcadeMachine*, *RoboyHead*, *RoboyUpperBody*, *RoboyXylophone*, *ShoulderTestbed*, *TheClaw*, *VRpuppet*, *Yatr*

Public Functions

Robot ()

Constructor.

~Robot ()

Destructor.

void **init** (string urdf_file_path, string viapoints_file_path, vector<string> joint_names)
initializes everything, call before use!

Parameters

- urdf_file_path: path to robot urdf
- viapoints_file_path: path to viapoints xml
- joint_names: a vector of joint_names to be considered from the model

void **update** ()

Updates the model.

virtual void **read** ()

This is the read function and should implement reading the state of your robot.

virtual void **write** ()

This is the write function and should implement writing commands to your robot.

void **forwardKinematics** (double dt)

Integrates the robot equation of motions using odeint.

Parameters

- dt: the integrations step length in seconds

Public Members

size_t **number_of_dofs** = 0

vector<string> **endeffectors**

number of degrees of freedom of the whole robot

map<string, size_t> **endeffector_index**

names of the endeffectors

vector<size_t> **endeffector_number_of_dofs**

vector<size_t> **endeffector_dof_offset**

size_t **number_of_joints** = 0
 number of degrees of freedom of each endeffector

size_t **number_of_cables** = 0
 number of joints of the whole robot

size_t **number_of_links** = 0
 number of cables, ie muscles of the whole robot

Matrix4d **world_H_base**
 number of links of the whole robot

vector<Matrix4d> **world_to_link_transform**
 floating base 6-DoF pose

vector<Matrix4d> **link_to_world_transform**

vector<Matrix4d> **frame_transform**

*Eigen::*Matrix<double, 6, 1> **baseVel**

Vector3d **gravity**
 the velocity of the floating base

MatrixXd **M**
 gravity vector (default: (0,0,-9.81))

VectorXd **CG**
 The Mass matrix of the robot.

VectorXd **q**
 The Coriolis+Gravity term of the robot.

VectorXd **qd**

VectorXd **qdd**

VectorXd **q_min**
 joint position, velocity, acceleration

VectorXd **q_max**

VectorXd **q_target**
 joint limits

VectorXd **qd_target**

VectorXd **qdd_target**

VectorXd **q_target_prev**
 joint position, velocity, acceleration targets

VectorXd **qd_target_prev**

VectorXd **qdd_target_prev**

VectorXd **l_int**
 joint position, velocity, acceleration targets

VectorXd **l**

VectorXd **l_target**

vector<VectorXd> **Ld**
 tendon length and length change

VectorXd **torques**

VectorXd **cable_forces**
joint torques

vector<VectorXd> **ld**
the cable forces in Newton

MatrixXd **L**
tendon length changes for each controller

MatrixXd **L_t**

MatrixXd **S**
 L and $-L^T$.

MatrixXd **P**

MatrixXd **V**

MatrixXd **W**

vector<vector<pair<*ViaPointPtr*, *ViaPointPtr*>>> **segments**
matrices of cable model

Protected Types

typedef boost::array<double, 2> **state_type**
odeint integration time

Protected Attributes

iDynTree::FreeFloatingGeneralizedTorques **bias**
cable segments

iDynTree::MatrixDynSize **Mass**
Coriolis+Gravity term.

bool **torque_position_controller_active** = false
Mass matrix.

bool **force_position_controller_active** = false

bool **cable_length_controller_active** = false

VectorXd **qdd_torque_control**

VectorXd **qdd_force_control**

vector<*Cable*> **cables**

vector<VectorXd> **joint_axis**
all cables of the robot

vector<string> **link_names**
joint axis of each joint

vector<string> **joint_names**

map<string, int> **link_index**
link and joint names of the robot

map<string, int> **joint_index**

```

vector<ros::Publisher> joint_command_pub
    link and joint indices of the robot

vector<int> controller_type
    currently active controller type

double integration_time = 0
    second order dynamics integration

vector<state_type> joint_state
    second order dynamics integration

vector<state_type> motor_state

bool first_time_solving = true
    joint and cable states

int nWSR = 1000

VectorXd f_min
    qp working sets

VectorXd f_max

int qp_print_level = PL_NONE

SQProblem qp_solver
    qpoptimizer print level

real_t *H
    qpoptimizer quadratic problem solver

real_t *g

real_t *A

real_t *lb

real_t *ub

real_t *b

real_t *FOpt

ros::Time last_visualization
    quadratic problem variables

Eigen::IOFormat fmt
    timestamp for visualization at reasonable intervals

hardware_interface::JointStateInterface joint_state_interface
    formator for terminal printouts

hardware_interface::EffortJointInterface joint_command_interface
    ros control joint state interface

hardware_interface::CardsflowStateInterface cardsflow_state_interface
    ros control joint command interface

hardware_interface::CardsflowCommandInterface cardsflow_command_interface
    cardsflow state interface

bool first_update = true
    cardsflow command interface

bool external_robot_target = false

```

Private Functions

void **MoveEndEffector** (**const** roby_control_msgs::MoveEndEffectorGoalConstPtr &goal)
Move endeffector action server.

Parameters

- goal:

bool **parseViapoints** (**const** string &viapoints_file_path, vector<Cable> &cables)
parses the cardsflow xml file for viapoint definitions

Return success

Parameters

- viapoints_file_path: path to the cardsflow.xml file
- cables: will be filled with the parsed viapoints of the defined cables

bool **ForwardKinematicsService** (roby_middleware_msgs::ForwardKinematics::Request &req,
roby_middleware_msgs::ForwardKinematics::Response
&res)
Forward kinematic service for endeffectors.

Return success

Parameters

- req: endeffector, requested joint angles
- res: 3d resulting position of endeffector

bool **InverseKinematicsService** (roby_middleware_msgs::InverseKinematics::Request &req,
roby_middleware_msgs::InverseKinematics::Response &res)
Inverse kinematic service for endeffectors.

Return success

Parameters

- req: endeffector and ik type
- res: joint configuration solution

bool **InverseKinematicsMultipleFramesService** (roby_middleware_msgs::InverseKinematicsMultipleFrames::Request &req,
roby_middleware_msgs::InverseKinematicsMultipleFrames::Response &res)

void **InteractiveMarkerFeedback** (**const** visualization_msgs::InteractiveMarkerFeedbackConstPtr &msg)
Callback for Interactive Marker Feedback of endeffectors.

When the Interactive Marker is released in rviz, the IK routine is called and the solution directly applied to the robot q_target angles

Parameters

- msg: Interactive Marker Feedback message

void **JointState** (**const** sensor_msgs::JointStateConstPtr &*msg*)

Callback for the joint state of the robot.

This can come from gazebo, the real robot or else where.

Parameters

- *msg*: message containing joint_name/angle information

void **FloatingBase** (**const** geometry_msgs::PoseConstPtr &*msg*)

Callback for the floating base world pose.

This can come from gazebo, the real robot or else where.

Parameters

- *msg*: message containing the 6 DoF pose of the floating base

VectorXd **resolve_function** (MatrixXd &*A_eq*, VectorXd &*b_eq*, VectorXd &*f_min*, VectorXd &*f_max*)

void **update_V** ()

Updates the V matrix of the cable model.

void **update_S** ()

Updates the S matrix of the cable model.

void **update_P** ()

Updates the P matrix of the cable model.

void **controllerType** (**const** roby_simulation_msgs::ControllerTypeConstPtr &*msg*)

Callback for controller type change.

The controller type defines how the forwardKinematics function integrates the robot states

Parameters

- *msg*: message containing the joint_name/type pair

Private Members

Matrix3d ***link_to_link_transform**

ros::NodeHandlePtr **nh**

boost::shared_ptr<ros::AsyncSpinner> **spinner**

ROS node handle.

ros::Publisher **robot_state_pub**

async ROS spinner

ros::Publisher **tendon_state_pub**

ros::Publisher **joint_state_pub**

ros::Publisher **robot_state_target_pub**

ROS robot pose and tendon publisher.

ros::Publisher **tendon_state_target_pub**

ros::Publisher **joint_state_target_pub**

ros::Subscriber **controller_type_sub**

target publisher

```

ros::Subscriber joint_state_sub
ros::Subscriber floating_base_sub
ros::Subscriber interactive_marker_sub
ros::ServiceServer ik_srv
    ROS subscribers.
ros::ServiceServer ik_two_frames_srv
ros::ServiceServer fk_srv
map<string, boost::shared_ptr<actionlib::SimpleActionServer<roboy_control_msgs::MoveEndEffectorAction>>> moveEndEff
iDynTree::KinDynComputations kinDynComp
iDynTree::KinDynComputations kinDynCompTarget
map<string, iDynTree::KinDynComputations> ik_models
    the full robot model
map<string, iDynTree::InverseKinematics> ik
    the robot models for each endeffector
map<string, string> ik_base_link
    the ik for each endeffector

struct cardsflow::kindyn::Robot::iDynTreeRobotState robotstate
struct RobotConfigurationEstimator : public Functor<double>

```

Public Functions

RobotConfigurationEstimator (int *number_of_samples*, int *number_of_cables*, int *number_of_links*)

int **operator ()** (const VectorXd &x, VectorXd &fvec) **const**
 This is the function that is called in each iteration.

Return

Parameters

- *x*: the pose vector (3 rotational 3 translational parameters)
- *fvec*: the error function (the difference between the sensor positions)

Public Members

int **number_of_samples**

int **number_of_cables**

int **number_of_links**

Public Static Attributes

cardsflow::kindyn::Robot ***robot**

class **Roboy2** : **public** *cardsflow::vrpuppet::Robot*

Public Functions

Roboy2 (string *urdf*, string *cardsflow_xml*)
 Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

bool **initPose** (std_srvs::Empty::Request &*req*, std_srvs::Empty::Response &*res*)

void **MotorStatus** (const roboy_middleware_msgs::MotorStatus::ConstPtr &*msg*)

void **read** ()
 Updates the robot model.

void **write** ()
 Sends motor commands to the real robot.

Public Members

ros::NodeHandlePtr **nh**

ros::Publisher **motor_command**
 ROS nodehandle.

ros::Subscriber **motor_status_sub**
 motor command publisher

ros::ServiceServer **init_pose**

ros::AsyncSpinner ***spinner**

map<string, ros::ServiceClient> **motor_control_mode**

map<string, ros::ServiceClient> **motor_config**

vector<string> **body_parts** = {"head"}

map<string, vector<string>> **endeffector_jointnames**

map<string, bool> **motor_status_received**

map<string, int> **init_mode**

map<string, int> **init_setpoint**

map<string, vector<int>> **real_motor_ids**

map<string, vector<int>> **sim_motor_ids**

map<string, vector<int>> **motor_type**

map<string, vector<double>> **l_offset**

map<string, vector<double>> **position**

map<string, vector<double>> **velocity**

map<string, vector<double>> **displacement**

map<string, int> **bodyPartIDs**

```
map<string, bool> use_motor_config
```

```
class RoboyArcadeMaschine : public cardsflow::kindyn::Robot
```

Public Functions

```
RoboyArcadeMaschine (string urdf, string cardsflow_xml)
```

Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

```
void read ()
```

Updates the robot model and integrates the robot model using the forwardKinematics function with a small step length.

```
void write ()
```

Sends motor commands to the real robot.

Public Members

```
ros::NodeHandlePtr nh
```

```
ros::Publisher motor_command
```

ROS nodehandle.

```
bool external_robot_state
```

motor command publisher

```
class RoboyHead : public cardsflow::kindyn::Robot
```

Public Functions

```
RoboyHead (string urdf, string cardsflow_xml)
```

Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

```
void FaceCoordinates (const geometry_msgs::Point::ConstPtr &msg)
```

```
void MotorStatus (const robyo_middleware_msgs::MotorStatus::ConstPtr &msg)
```

```
void read ()
```

Updates the robot model and integrates the robot model using the forwardKinematics function with a small step length.

```
void write ()
```

Sends motor commands to the real robot.

Public Members

```

ros::NodeHandlePtr nh
ros::Publisher motor_command
    ROS nodehandle.
ros::Publisher sphere_head_axis0
ros::Publisher sphere_head_axis1
ros::Publisher sphere_head_axis2
ros::Subscriber motor_status
    motor command publisher
ros::Subscriber face_coordinates
ros::ServiceClient sphere_left_axis0_params
ros::ServiceClient sphere_left_axis1_params
ros::ServiceClient sphere_left_axis2_params
map<string, ros::ServiceClient> motor_control_mode
map<string, ros::ServiceClient> motor_config
vector<string> endeffectors = {"shoulder_right"}
map<string, vector<string>> endeffector_jointnames
bool external_robot_state
bool status_received = false
    indicates if we get the robot state externally
std_msgs::Float32 roll
std_msgs::Float32 pitch
std_msgs::Float32 yaw
float err_x = 0
float error_y = 0
float pitch_max = 0.33
float pitch_min = -0.50
float yaw_min = -0.50
float yaw_max = 0
map<string, vector<int> > RoboyHead::motors= {                {"shoulder_right", {9,10,11,12
map<string, vector<int> > RoboyHead::sim_motors= {            {"shoulder_right", {0,1,2,
map<string, vector<float> > RoboyHead::l_change= {            {"shoulder_right", {0,0,0,
map<string, vector<double> > RoboyHead::l_offset= {            {"head", {0,0,0,0,0,0}},
map<string, vector<double> > RoboyHead::encoder_offset= {      {"head", {0,0,0,0,0,0
class RoboyIcecream: public cardsflow::vrpuppet::Robot

```

Public Functions

RoboyIcecream (string *urdf*, string *cardsflow_xml*)
Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

bool **initPose** (std_srvs::Empty::Request &*req*, std_srvs::Empty::Response &*res*)

void **MotorStatus** (const roboy_middleware_msgs::MotorStatus::ConstPtr &*msg*)

void **read** ()
Updates the robot model.

void **write** ()
Sends motor commands to the real robot.

Public Members

ros::NodeHandlePtr **nh**

ros::Publisher **motor_command**
ROS nodehandle.

ros::Subscriber **motor_status_sub**
motor command publisher

ros::ServiceServer **init_pose**

ros::AsyncSpinner ***spinner**

map<string, ros::ServiceClient> **motor_control_mode**

map<string, ros::ServiceClient> **motor_config**

vector<string> **body_parts** = {"head", , }

map<string, vector<string>> **endeffector_jointnames**

map<string, bool> **motor_status_received**

map<string, int> **init_mode**

map<string, int> **init_setpoint**

map<string, vector<int>> **real_motor_ids**

map<string, vector<int>> **sim_motor_ids**

map<string, vector<int>> **motor_type**

map<string, vector<double>> **l_offset**

map<string, vector<double>> **position**

map<string, vector<double>> **velocity**

map<string, vector<double>> **displacement**

map<string, int> **bodyPartIDs**


```
map<string, bool> use_motor_config
class RoboyUpperBody : public cardsflow::kindyn::Robot
```

Public Functions

RoboyUpperBody (string *urdf*, string *cardsflow_xml*)
Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

void **read** ()
Updates the robot model and integrates the robot model using the forwardKinematics function with a small step length.

void **write** ()
Sends motor commands to the real robot.

Public Members

```
ros::NodeHandlePtr nh
ros::Publisher motor_command
    ROS nodehandle.
ros::ServiceClient motor_config
    motor command publisher
ros::ServiceClient sphere_left_axis0_params
ros::ServiceClient sphere_left_axis1_params
ros::ServiceClient sphere_left_axis2_params
map<string, ros::ServiceClient> motor_control_mode
vector<string> endeffectors = {"spine_right"}
map<string, vector<string>> endeffector_jointnames
bool external_robot_state
map<string, vector<int> > RoboyUpperBody::motors= {          {"head", {9, 10, 11, 12, 13, 14}}
    indicates if we get the robot state externally
map<string, vector<int> > RoboyUpperBody::sim_motors= {          {"head", {36, 37, 35, 34, 33}}
map<string, vector<double> > RoboyUpperBody::l_offset= {          {"head", {0, 0, 0, 0, 0, 0}}
class RoboyXylophone : public cardsflow::kindyn::Robot
```

Public Functions

RoboyXylophone (string *urdf*, string *cardsflow_xml*)
Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

void **read** ()
Updates the robot model and integrates the robot model using the forwardKinematics function with a small step length.

void **write** ()
Sends motor commands to the real robot.

Public Members

ros::NodeHandlePtr **nh**

ros::Publisher **motor_command**
ROS nodehandle.

bool **external_robot_state**
motor command publisher

vector<double> **l_offset**
indicates if we get the robot state externally

class rviz_visualization
Subclassed by *cardsflow::kindyn::Robot*, *cardsflow::vrpuppet::Robot*, *CardsflowRviz*

Public Functions

rviz_visualization ()

~rviz_visualization ()

void **initializeInteractiveMarkerServer** ()

Marker **makeBox** (InteractiveMarker &*msg*)

InteractiveMarkerControl &**makeBoxControl** (InteractiveMarker &*msg*)

void **make6DofMarker** (bool *fixed*, unsigned int *interaction_mode*, **const** tf::Vector3 &*position*, bool *show_6dof*, double *scale* = 1, **const** char **frame* = "world", **const** char **name* = "interactive_marker", **const** char **description* = "for interaction and shit")

Vector3d **convertGeometryToEigen** (**const** geometry_msgs::Vector3 &*vector_in*)

geometry_msgs::Vector3 **convertEigenToGeometry** (**const** Vector3d &*vector_in*)

void **PoseMsgToTF** (**const** geometry_msgs::Pose &*msg*, tf::Transform &*bt*)

```
void publishMesh (const char *package, const char *relative_path, const char *modelname,
                 Vector3d &pos, Quaterniond &orientation, double scale, const char *frame,
                 const char *ns, int message_id, double duration = 0, COLOR color = COLOR(1,
                 , , ))
```

Publishes a mesh visualization marker.

Parameters

- *package*: ros package in which this mesh is located
- *relative_path*: the relative path inside that ros packae
- *the*: mesh file name
- *pos*: at this position
- *orientation*: with this orientation
- *modelname*: name of the mesh.dae
- *frame*: in this frame
- *ns*: namespace
- *message_id*: unique id
- *duration*: in seconds
- *color*: of the mesh

```
void publishMesh (const char *package, const char *relative_path, const char *modelname, ge-
                 ometry_msgs::Pose &pose, double scale, const char *frame, const char *ns, int
                 message_id, double duration = 0, COLOR color = COLOR(1, , , ))
```

Publishes a mesh visualization marker.

Parameters

- *package*: ros package in which this mesh is located
- *relative_path*: the relative path inside that ros packae
- *the*: mesh file name
- *pose*: of mesh
- *modelname*: name of the mesh.dae
- *frame*: in this frame
- *ns*: namespace
- *message_id*: unique id
- *duration*: in seconds
- *color*: of the mesh

```
void publishSphere (Vector3d &pos, const char *frame, const char *ns, int message_id, COLOR
                 color, float radius = 0.01, double duration = 0)
```

Publishes a sphere visualization marker.

Parameters

- *pos*: at this positon
- *frame*: in this frame

- `ns`: namespace
- `message_id`: a unique id
- `rgda`: rgb color (0-1) plus transparency
- `duration`: for this duration in seconds (0=forever)

void **publishSphere** (geometry_msgs::Pose &*pose*, **const** char **frame*, **const** char **ns*, int *message_id*, *COLOR* color, float *radius* = 0.01, double *duration* = 0)
Publishes a sphere visualization marker.

Parameters

- `pose`: at this position
- `frame`: in this frame
- `ns`: namespace
- `message_id`: a unique id
- `rgda`: rgb color (0-1) plus transparency
- `duration`: for this duration in seconds (0=forever)

void **publishCube** (Vector3d &*pos*, Vector4d &*quat*, **const** char **frame*, **const** char **ns*, int *message_id*, *COLOR* color, float *radius* = 0.01, double *duration* = 0)
Publishes a cube visualization marker.

Parameters

- `pos`: at this position
- `quat`: with this orientation
- `frame`: in this frame
- `ns`: namespace
- `message_id`: a unique id
- `rgda`: rgb color (0-1) plus transparency
- `duration`: for this duration in seconds (0=forever)

void **publishCube** (geometry_msgs::Pose &*pose*, **const** char **frame*, **const** char **ns*, int *message_id*, *COLOR* color, float *radius* = 0.01, double *duration* = 0)
Publishes a cube visualization marker.

Parameters

- `pose`: with this pose
- `frame`: in this frame
- `ns`: namespace
- `message_id`: a unique id
- `rgda`: rgb color (0-1) plus transparency
- `duration`: for this duration in seconds (0=forever)

void **publishCube** (Vector3d &*pos*, Quaternionf &*quat*, **const** char **frame*, **const** char **ns*, int *message_id*, *COLOR* *color*, float *dx* = 0.01, float *dy* = 0.01, float *dz* = 0.01, double *duration* = 0)

Publishes a cube visualization marker.

Parameters

- *pos*: position
- *quat*: quaternion
- *frame*: in this frame
- *ns*: namespace
- *message_id*: a unique id
- *rgda*: rgb color (0-1) plus transparency
- *dx*: cube x dim
- *dy*: cube y dim
- *dz*: cube z dim
- *duration*: for this duration in seconds (0=forever)

void **publishCylinder** (Vector3d &*pos*, **const** char **frame*, **const** char **ns*, int *message_id*, *COLOR* *color*, float *radius* = 0.01, double *duration* = 0)

Publishes a cylinder visualization marker.

Parameters

- *pos*: at this position
- *frame*: in this frame
- *ns*: namespace
- *message_id*: a unique id
- *rgda*: rgb color (0-1) plus transparency
- *duration*: for this duration in seconds (0=forever)

void **publishRay** (Vector3d &*pos*, Vector3d &*dir*, **const** char **frame*, **const** char **ns*, int *message_id*, *COLOR* *color*, double *duration* = 0)

Publishes a ray visualization marker.

Parameters

- *pos*: at this position
- *dir*: direction
- *frame*: in this frame
- *message_id*: a unique id
- *ns*: namespace
- *color*: rgb color (0-1) plus transparency
- *duration*: for this duration in seconds (0=forever)

void **publishText** (Vector3d &pos, **const** char *text, **const** char *frame, **const** char *ns, int message_id, *COLOR* color, double duration, float height)
Publishes a text message marker.

Parameters

- pos: at this position
- text: with this text
- frame: in this frame
- ns: namespace
- message_id: a unique id
- color: rgb color (0-1) plus transparency
- duration: for this duration in seconds (0=forever)
- height: height of the text

void **clearMarker** (int id)
Clears a marker with this id.

void **clearAll** ()
Clears all markers in rviz.

bool **getTransform** (string from, string to, geometry_msgs::Pose &transform)
Gets a tf transform.

Return success

Parameters

- from: source frame
- to: target frame
- transform: will be filled with the transform

bool **getTransform** (string from, string to, Matrix4d &transform)
Gets a tf transform.

Return success

Parameters

- from: source frame
- to: target frame
- transform: will be filled with the transform

bool **getLighthouseTransform** (bool lighthouse, **const** char *to, Matrix4d &transform)
Queries the tf listener for the specified transform.

Return true if available

Parameters

- lighthouse:

- `to`: another frame
- `transform`: the transform if available

bool **getLighthouseTransform**(const char *from, bool *lighthouse*, Matrix4d &transform)

Queries the tf listener for the specified transform.

Return true if available

Parameters

- `lighthouse`:
- `from`: another frame
- `transform`: the transform if available

bool **getTransform**(const char *from, const char *to, tf::Transform &transform)

Queries the tf listener for the specified transform.

Return true if available

Parameters

- `to`: this frame
- `from`: another frame
- `transform`: the transform if available

bool **publishTransform**(string from, string to, geometry_msgs::Pose &transform)

Publishes a tf transform.

Return success

Parameters

- `from`: source frame
- `to`: target frame
- `transform`:

Public Members

ros::Publisher **visualization_pub**

ros::Publisher **visualization_array_pub**

bool **publish_as_marker_array** = false

int **number_of_markers_to_publish_at_once** = 100

Public Static Functions

void **processFeedback**(const *visualization_msgs::InteractiveMarkerFeedbackConstPtr* &feedback)

Private Members

```
ros::NodeHandlePtr nh  
visualization_msgs::MarkerArray marker_array  
tf::TransformListener listener  
tf::TransformBroadcaster broadcaster
```

Private Static Attributes

```
boost::shared_ptr<interactive_markers::InteractiveMarkerServer> interactive_marker_server  
interactive_markers::MenuHandler menu_handler  
bool first = true  
class ShoulderTestbed : public cardsflow::kindyn::Robot
```

Public Functions

```
ShoulderTestbed (string urdf, string cardsflow_xml)  
    Constructor.
```

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

```
void read ()  
    Updates the robot model and if we do not use gazebo for simulation, we integrate using the forwardKinematics function with a small step length.
```

```
void write ()  
    Sends motor commands to the real robot.
```

Public Members

```
ros::Time last_update  
bool external_robot_state  
ros::NodeHandlePtr nh  
    indicates if we get the robot state externally  
ros::Publisher motor_command  
    ROS nodehandle.  
double l_offset[NUMBER_OF_MOTORS]  
    motor command publisher  
double winch_radius[8] = {0.007, , , , , , , }  
float Kp = 0.001  
struct Tendon
```


Public Members

float **force**

float **l**

float **ld**

vector<geometry_msgs::Vector3> **viaPoints**

class TheClaw : public *cardsflow::kindyn::Robot*

Public Functions

TheClaw (string *urdf*, string *cardsflow_xml*)

Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

void **read** ()

Updates the robot model and if we do not use gazebo for simulation, we integrate using the forwardKinematics function with a small step length.

int **meterPerSecondToServoSpeed** (double *meterPerSecond*)

Converts tendon length chages from the cable model to pwm commands of the real robot.

Return pwm

Parameters

- *meterPerSecond*: tendon length change

void **write** ()

Sends motor commands to the real robot.

Public Members

bool **external_robot_state**

ros::NodeHandlePtr **nh**

indicates if we get the robot state externally

ros::Publisher **motor_command**

ROS nodehandle.

class TorquePositionController : public controller_interface::Controller<*hardware_interface::EffortJointInterface*>

Public Functions

TorquePositionController ()

Constructor.

bool **init** (*hardware_interface::EffortJointInterface *hw*, *ros::NodeHandle &n*)

Initializes the controller.

Will be call by controller_manager when loading this controller

Return success

Parameters

- *hw*: pointer to the hardware interface
- *n*: the nodehandle

void **update** (**const** *ros::Time &time*, **const** *ros::Duration &period*)

Called regularly by controller manager.

The torque for a joint wrt to a PD controller on the joint target position is calculated.

Parameters

- *time*: current time
- *period*: period since last control

void **starting** (**const** *ros::Time &time*)

Called by controller manager when the controller is about to be started.

Parameters

- *time*: current time

void **stopping** (**const** *ros::Time &time*)

Called by controller manager when the controller is about to be stopped.

Parameters

- *time*: current time

void **JointCommand** (**const** *std_msgs::Float32ConstPtr &msg*)

Joint position command callback for this joint.

Parameters

- *msg*: joint position target in radians

bool **setControllerParameters** (*robby_control_msgs::SetControllerParameters::Request &req*,
robby_control_msgs::SetControllerParameters::Response &res)

Controller Parameters service.

Return success

Parameters

- *req*: requested gains
- *res*: success

Private Members

```
double q_target = 0
double Kp = 3000
    joint position target
double Kd = 5
ros::NodeHandle nh
    PD gains.
ros::Publisher controller_state
    ROS nodehandle.
ros::ServiceServer controller_parameter_srv
    publisher for controller state
boost::shared_ptr<ros::AsyncSpinner> spinner
    service for controller parameters
hardware_interface::JointHandle joint
ros::Subscriber joint_command
string joint_name
    joint command subscriber
```

```
class UDPSocket
```

Public Functions

```
UDPSocket (const char *server_IP, int server_port, const char *client_IP, int client_port, bool exclusive = true)
```

Creates a socket on the given server_IP and server_port and sets up the “connection” with the client.

Because it is UDP, there is no handshake, the socket just sends and listens to packages from the client_IP and client_port

Parameters

- server_IP: the server socket IP
- server_port: the server socket port
- client_IP: the client to send and receive UDP packets to/from
- client_port: the client port
- exclusive: receive exclusively packages from client

```
UDPSocket (const char *client_IP, int client_port, bool exclusive = true)
```

Tries to guess the users IP and sets up the socket on Port 8000 and “connects” to client_IP on client_port.

Parameters

- client_IP: the client to send and receive UDP packets to/from
- client_port: the client port
- exclusive: receive exclusively packages from client

```
UDPSocket (const char *client_IP, int client_port, int server_port, bool exclusive = true)
```

UDPSocket (int *port*, int *broadcastIP* = 0xffffffff, bool *broadcaster* = false)
Creates a broadcast socket on port.

Parameters

- *port*: on this port
- *broadcastIP*: use this broadcast IP
- *broadcaster*: if true this binds the port to the broadcast port

UDPSocket (int *port*, bool *broadcaster*)
Creates a broadcast socket on port.

Parameters

- *port*: on this port
- *broadcaster*: if true this binds the port to the broadcast port

~UDPSocket ()

uint32_t **receiveHostIP** (const char **key*, uint32_t &*IP*)
Receive ROS master IP.

Return status (0 invalid, 1 too short, 2 too long, 3 accepted)

Parameters

- *key*: only if this key is matched with the UDP message

bool **broadcastHostIP** (uint32_t *IP*)
Broadcast ROS master IP.

Return successfully broadcasted

Parameters

- *IP*: broadcast this IP

bool **broadcastHostIP** (char **key*, int *length*)
Broadcast ROS master IP of the machine this node was run on.

Return successfully broadcasted

Parameters

- *key*: the key to unlock
- *length*: the length inbytes of the key

bool **receiveSensorData** (uint32_t &*sensorID*, bool &*lighthouse*, bool &*axis*, uint16_t &*sweepDuration*)
Listens for UDP package containing lighthouse tracking data.

Return

Parameters

- *sensorID*: sensor id

- `lighthouse`: witch lighthouse
- `axis`:
- `sweepDuration`:

bool **receiveSensorData** (vector<uint32_t> &*sensorID*, vector<bool> &*lighthouse*, vector<bool> &*axis*, vector<uint32_t> &*sweepDuration*)

bool **convertByte2Text** (uint32_t *inet*, char **inet_str*)
Converts a byte internet address to a human readable address.

Return success

Parameters

- `inet`:
- `inet_str`:

bool **convertText2Byte** (char **inet_str*, uint32_t **inet*)
Converts a human readable address to a byte internet address.

Return success

Parameters

- `inet_str`:
- `inet`:

int **receiveUDP** ()
receive from anyone

Return received number of bytes

int **receiveUDPFromClient** ()
receive from client

Return received number of bytes

bool **sendUDPToClient** ()
send to client

Return success

Public Members

pair<uint32_t, string> **myIP**

pair<uint32_t, string> **myBroadcastIP**

char **buf**[**MAXBUFLLENGTH**]

struct sockaddr_in **client_addr**

ssize_t **numbytes**

Private Functions

bool **setTimeout** (int *usecs*)
Sets the UDP package receive and send timeout.

Return success

Parameters

- *usecs*: time in microseconds

bool **whatMyIP** (string &*IP*, string &*Broadcast_IP*, bool *preferEthernet* = true)
Tries to guess your IP.

Parameters

- *IP*: your IP
- *Broadcast_IP*: your broadcast IP
- *preferEthernet*: if True ethernet will be preferred over wifi
- *success*: (fails if I can't find a valid IP for wifi or ethernet adapter)

bool **broadcastUDP** ()
broadcast

Return success

Private Members

bool **initialized** = false

int **sockfd**

struct sockaddr_in **server_addr**

struct sockaddr_in **broadcast_addr**

socklen_t **client_addr_len**

socklen_t **server_addr_len**

socklen_t **broadcast_addr_len**

struct addrinfo ***servinfo**

bool **exclusive**

struct **ViaPoint**

Public Functions

ViaPoint (string *link_name*, Vector3d &*local_coordinates*, bool *fixed_to_world* = false)

Parameters

- *link_name*:
- *local_coordinates*:

- `fixed_to_world`:

Public Members

string **link_name**

int **link_index**
the name of the link the viapoint belongs to

bool **fixed_to_world**
the index of the link the viapoint belongs to wrt. the robot model

Vector3d **local_coordinates**
indicates if the viapoint is moving with the link or is fixed to the world

Vector3d **global_coordinates**
the local coordinates in the link frame

class VRpuppet : public *cardsflow::kinyon::Robot*

Public Functions

VRpuppet (string *urdf*, string *cardsflow_xml*)
Constructor.

Parameters

- *urdf*: path to urdf
- *cardsflow_xml*: path to cardsflow xml

void **read** ()
Updates the robot model and integrates the robot model using the forwardKinematics function with a small step length.

void **write** ()
Sends motor commands to the real robot.

Public Members

ros::NodeHandlePtr **nh**

ros::Publisher **motor_command**
ROS nodehandle.

ros::ServiceClient **motor_config**
motor command publisher

ros::ServiceClient **sphere_left_axis0_params**

ros::ServiceClient **sphere_left_axis1_params**

ros::ServiceClient **sphere_left_axis2_params**

map<string, ros::ServiceClient> **motor_control_mode**

vector<string> **endeffectors** = {"spine_right"}

map<string, vector<string>> **endeffector_jointnames**

```

bool external_robot_state

map<string,vector<short unsigned int> > VRpuppet::motors= {           {"head", {9,10,1
    indicates if we get the robot state externally

map<string,vector<short unsigned int> > VRpuppet::sim_motors= {           {"head", {36
map<string,vector<double> > VRpuppet::l_offset= {           {"head", {0,0,0,0,0,0}},
class Yatr:public cardsflow::kindyn::Robot

```

Public Functions

Yatr (string *urdf*, string *cardsflow_xml*)

void **read** ()

Updates the robot model and if we do not use gazebo for simulation, we integrate using the forwardKinematics function with a small step length.

void **write** ()

Sends motor commands to the real robot.

Public Members

bool **external_robot_state**

ros::NodeHandlePtr **nh**

indicates if we get the robot state externally

ros::Publisher **motor_command**

ROS nodehandle.

namespace **capture_pedal_trajectory**

Functions

getPositionLeftFoot ()

UTILITY FUNCTIONS ###.

getPositionRightFoot ()

setJointControllerParameters (proportionalVal proportionalVal, derivativeVal derivativeVal)

jointStateCallback (joint_data joint_data)

getPedalPositions (numSamples numSamples)

plotPedalTrajectories ()

plotEverything (numSamples numSamples, jointAngleDict jointAngleDict)

inverse_kinematics_client (endeffector endeffector, frame frame, x x, y y, z z)

inverse_kinematics_multiple_frames_client (endeffector endeffector, frames frames, x x,
y y, z z, weights weights)

main ()

MAIN ###.

Variables

```

string capture_pedal_trajectory.JSON_FILENAME = "captured_pedal_trajectory_22feb.json"
float capture_pedal_trajectory.JOINT_ANGLE_TOLERANCE_FK = 0.05
int capture_pedal_trajectory.JOINT_KP = 10
int capture_pedal_trajectory.JOINT_KD = 0
float capture_pedal_trajectory.PEDAL_CENTER_OFFSET_X = 0.09223
    MEASURED PARAMETERS ###.
float capture_pedal_trajectory.PEDAL_CENTER_OFFSET_Y = 0.00013
float capture_pedal_trajectory.PEDAL_CENTER_OFFSET_Z = 0.00426
int capture_pedal_trajectory.BIKE_OFFSET_X = 0
int capture_pedal_trajectory.BIKE_OFFSET_Y = 0
int capture_pedal_trajectory.BIKE_OFFSET_Z = 0
float capture_pedal_trajectory.PEDAL_RADIUS = 0.16924
float capture_pedal_trajectory.RIGHT_LEG_OFFSET_Y = 0.15796
float capture_pedal_trajectory.LEFT_LEG_OFFSET_Y = -0.18736
string capture_pedal_trajectory.ROS_JOINT_HIP_RIGHT = "joint_hip_right"
    GLOBAL VARIABLES ###.
string capture_pedal_trajectory.ROS_JOINT_KNEE_RIGHT = "joint_knee_right"
string capture_pedal_trajectory.ROS_JOINT_ANKLE_RIGHT = "joint_foot_right"
string capture_pedal_trajectory.ROS_JOINT_HIP_LEFT = "joint_hip_left"
string capture_pedal_trajectory.ROS_JOINT_KNEE_LEFT = "joint_knee_left"
string capture_pedal_trajectory.ROS_JOINT_ANKLE_LEFT = "joint_foot_left"
string capture_pedal_trajectory.RIGHT_HIP_JOINT = "right_hip"
string capture_pedal_trajectory.RIGHT_KNEE_JOINT = "right_knee"
string capture_pedal_trajectory.RIGHT_ANKLE_JOINT = "right_ankle"
string capture_pedal_trajectory.LEFT_HIP_JOINT = "left_hip"
string capture_pedal_trajectory.LEFT_KNEE_JOINT = "left_knee"
string capture_pedal_trajectory.LEFT_ANKLE_JOINT = "left_ankle"
dictionary capture_pedal_trajectory._jointsStatusData= {    RIGHT_HIP_JOINT: {
namespace capture_pedal_trajectory_left_leg_only

```

Functions

```

get_distance (point1 point1, point2 point2)
    UTILITY FUNCTIONS ###.

```

Documentation for a function

Return the distance between two points where points are a list of two coordinates.

```

getPositionLeftFoot ()

```

```

getPositionRightFoot ()
setJointControllerParameters (proportionalVal proportionalVal, derivativeVal derivativeVal)
jointStateCallback (joint_data joint_data)
getPedalPositions (numSamples numSamples)
plotPedalTrajectories ()
plotEverything (numSamples numSamples, jointAngleDict jointAngleDict)
inverse_kinematics_client (endeffector endeffector, frame frame, x x, y y, z z)
inverse_kinematics_multiple_frames_client (endeffector endeffector, frames frames, x x,
                                           y y, z z, weights weights)

main_notnow ()
    MAIN ###.

main ()

```

Variables

```

string capture_pedal_trajectory_left_leg_only.JSON_FILENAME = "captured_pedal_trajectory_left_leg_only.JSON_FILENAME"
string capture_pedal_trajectory_left_leg_only.TEMP_READFILE = "captured_pedal_trajectory_left_leg_only.TEMP_READFILE"
float capture_pedal_trajectory_left_leg_only.JOINT_ANGLE_TOLERANCE_FK = 0.002
int capture_pedal_trajectory_left_leg_only.JOINT_KP = 200
int capture_pedal_trajectory_left_leg_only.JOINT_KD = 0
float capture_pedal_trajectory_left_leg_only.POINT_REACHED_TOLERANCE = 0.02
capture_pedal_trajectory_left_leg_only.LEFT_HIP_JOINT_TEST_CONFIGURATIONS = np.linspace(0, 2, 10)
capture_pedal_trajectory_left_leg_only.LEFT_KNEE_JOINT_TEST_CONFIGURATIONS = np.linspace(0, 2, 10)
capture_pedal_trajectory_left_leg_only.LEFT_ANKLE_JOINT_TEST_CONFIGURATIONS = np.linspace(0, 2, 10)
float capture_pedal_trajectory_left_leg_only.LEFT_HIP_JOINT_LOWER_LIMIT = -1.9
int capture_pedal_trajectory_left_leg_only.LEFT_KNEE_JOINT_LOWER_LIMIT = 0
float capture_pedal_trajectory_left_leg_only.LEFT_ANKLE_JOINT_LOWER_LIMIT = -0.7
float capture_pedal_trajectory_left_leg_only.LEFT_HIP_JOINT_UPPER_LIMIT = 0.5
float capture_pedal_trajectory_left_leg_only.LEFT_KNEE_JOINT_UPPER_LIMIT = 2.0
float capture_pedal_trajectory_left_leg_only.LEFT_ANKLE_JOINT_UPPER_LIMIT = 0.7
float capture_pedal_trajectory_left_leg_only.PEDAL_CENTER_OFFSET_X = 0.09222872619138
    MEASURED PARAMETERS ###.
float capture_pedal_trajectory_left_leg_only.PEDAL_CENTER_OFFSET_Y = 0.00013171801209
float capture_pedal_trajectory_left_leg_only.PEDAL_CENTER_OFFSET_Z = 0.00427728349654
int capture_pedal_trajectory_left_leg_only.BIKE_OFFSET_X = 0
int capture_pedal_trajectory_left_leg_only.BIKE_OFFSET_Y = 0
int capture_pedal_trajectory_left_leg_only.BIKE_OFFSET_Z = 0
float capture_pedal_trajectory_left_leg_only.PEDAL_RADIUS = 0.16924

```

```

float capture_pedal_trajectory_left_leg_only.RIGHT_LEG_OFFSET_Y = 0.1579606226770175
float capture_pedal_trajectory_left_leg_only.LEFT_LEG_OFFSET_Y = -0.1873561275007122
string capture_pedal_trajectory_left_leg_only.ROS_JOINT_HIP_RIGHT = "joint_hip_right"
GLOBAL VARIABLES ###.
string capture_pedal_trajectory_left_leg_only.ROS_JOINT_KNEE_RIGHT = "joint_knee_right"
string capture_pedal_trajectory_left_leg_only.ROS_JOINT_ANKLE_RIGHT = "joint_foot_right"
string capture_pedal_trajectory_left_leg_only.ROS_JOINT_HIP_LEFT = "joint_hip_left"
string capture_pedal_trajectory_left_leg_only.ROS_JOINT_KNEE_LEFT = "joint_knee_left"
string capture_pedal_trajectory_left_leg_only.ROS_JOINT_ANKLE_LEFT = "joint_foot_left"
string capture_pedal_trajectory_left_leg_only.RIGHT_HIP_JOINT = "right_hip"
string capture_pedal_trajectory_left_leg_only.RIGHT_KNEE_JOINT = "right_knee"
string capture_pedal_trajectory_left_leg_only.RIGHT_ANKLE_JOINT = "right_ankle"
string capture_pedal_trajectory_left_leg_only.LEFT_HIP_JOINT = "left_hip"
string capture_pedal_trajectory_left_leg_only.LEFT_KNEE_JOINT = "left_knee"
string capture_pedal_trajectory_left_leg_only.LEFT_ANKLE_JOINT = "left_ankle"
dictionary capture_pedal_trajectory_left_leg_only._jointsStatusData= { RIGHT_HIP_JOINT
namespace CARDSflow

```

Enums

```
enum ControllerType
```

Values:

```
    cable_length_controller
```

```
    torque_position_controller
```

```
    force_position_controller
```

```
namespace cardsflow
```

```
namespace kindyn
```

Typedefs

```
typedef boost::shared_ptr<ViaPoint> ViaPointPtr
```

```
namespace vrpuppet
```

```
namespace compute_steering_path
```

Functions

```
computeSteeringAngles()
```

UTILITY FUNCTIONS ###.

```
computeHandTrajectories()
```

```
main()  
    MAIN ###.
```

Variables

```
int compute_steering_path.MAX_TURNING_ANGLE = math.pi / 6  
    FUNCTION PARAMETERS ###.  
  
int compute_steering_path.NUM_STEERING_ANGLES = 61  
int compute_steering_path.RIKSHAW_TURN_JOINT_X_OFFSET = 0  
int compute_steering_path.RIKSHAW_TURN_JOINT_Y_OFFSET = 0  
int compute_steering_path.RIKSHAW_TURN_JOINT_Z_OFFSET = 0  
float compute_steering_path.HANDLEBAR_X_OFFSET = 0.728713  
float compute_steering_path.HANDLEBAR_Z_OFFSET = 0.719269  
float compute_steering_path.HAND_Y_OFFSET = 0.125  
list compute_steering_path._steeringAngles = []  
    GLOBAL VARIABLES ###.  
  
list compute_steering_path._rightHandTrajectory = []  
list compute_steering_path._leftHandTrajectory = []  
list compute_steering_path._centerHandlebarTrajectory = []  
  
namespace Eigen  
namespace EigenExtension
```

Functions

Matrix3f **SkewSymmetric** (Vector3f *v*)
Skew symmetric matrix for cross product computation.

Return skew symmetric matrix

Parameters

- *v*: 3d Vector

Matrix3d **SkewSymmetric2** (Vector3d *v*)
Skew symmetric matrix for cross product computation.

Return skew symmetric matrix

Parameters

- *v*: 3d Vector

Matrix<float, 2, 1> **GetLinearSplineCoefficients** (float *q_r_i*, float *q_r_ip1*, double *t*)

Matrix<float, 4, 1> **GetCubicSplineCoefficients** (float *q_r_i*, float *q_d_r_i*, float *q_r_ip1*, float *q_d_r_ip1*, double *t*)

Matrix<float, 6, 1> **GetQuinticSplineCoefficients** (float q_{r_i} , float $q_{d_{r_i}}$, float $q_{dd_{r_i}}$,
float $q_{r_{ip1}}$, float $q_{d_{r_{ip1}}}$, float
 $q_{dd_{r_{ip1}}}$, double t)

void **LinearSplineInterpolate** (float $*q_r$, float $*q_{d_r}$, float $*q_{dd_r}$, Matrix<float, 2, 1> a ,
double t)

void **CubicSplineInterpolate** (float $*q_r$, float $*q_{d_r}$, float $*q_{dd_r}$, Matrix<float, 4, 1> a , dou-
ble t)

void **QuinticSplineInterpolate** (float $*q_r$, float $*q_{d_r}$, float $*q_{dd_r}$, Matrix<float, 6, 1> a ,
double t)

Matrix3f **ComputeRotationMatrix** (AngleAxisf a)

Matrix3f **ComputeRotationMatrixDeriv** (AngleAxisf a , AngleAxisf a_d)

Matrix3f **ComputeRotationMatrixDoubleDeriv** (AngleAxisf a , AngleAxisf a_d , AngleAxisf
 a_{dd})

MatrixXf **Pinv** (MatrixXf A)

Pseudo inverse matrix.

Return Pseudo inverse matrix

Parameters

- A :

MatrixXd **Pinv** (MatrixXd A)

Pseudo inverse matrix.

Return Pseudo inverse matrix

Parameters

- A :

namespace finals_simulation_pedaling

Functions

joint_state_callback ($joint_data\ joint_data$)

Documentation for a function.

This function collects the current status of the joint-angles and saves them in the global dictionary “joint_status_data”.

import_joint_trajectory_record ()

Documentation for a function.

Collects and saves all joint- and pedal-angles from the pre-captured trajectory from the (global variable).

get_joint_position ($jointName\ jointName$)

Documentation for a function.

Return current joint-angle of .

plot_measured_trajectories ($input_float\ input_float$)

get_joint_velocity (*jointName jointName*)

Documentation for a function.

Return current joint-velocity of .

get_position (*endeffector endeffector, frame frame*)

Documentation for a function.

Return the position of the of the correspondent .

set_joint_controller_parameters (*proportionalVal proportionalVal, derivativeVal derivativeVal*)

Documentation for a function.

Sets Kp of joint-controller to Sets Kd of joint-controller to

get_position_left_foot ()

Documentation for a function.

Return the position of the left foot of Roboy.

get_position_right_foot ()

Documentation for a function.

Return the position of the right foot of Roboy.

get_distance (*point1 point1, point2 point2*)

Documentation for a function.

Return the distance between two points where points are a list of two coordinates.

check_output_limits (*inputVal inputVal*)

Documentation for a function.

Checks if is inside possible range of joint-angle velocities. If not, returns max-velocity if too high or min-velocity if too low instead of .

compute_velocity (*joint_name joint_name, next_joint_angle next_joint_angle, current_joint_angle current_joint_angle, end_time end_time*)

Documentation for a function.

Returns ideal joint-velocity for according to and joint-angle-difference between and :

$\text{ideal_velocity} = \text{joint_angle_difference} / (\text{end_time} - \text{current_time})$

check_pedal_angle_valid (*pedal_angle pedal_angle*)

get_joint_angle (*joint_name joint_name, pedal_angle pedal_angle*)

Documentation for a function.

Evaluate and return joint-angle of correspondent to using the interpolated function:

The functions can be used by calling “<function_name>(<pedal_angle>)” ==> returns <joint_angle>

interpolate_functions ()

Documentation for a function.

Initializing interpolated functions for joint-angles using pre-captured set points with pedal-angle as input and joint-angle as output:

The functions can be used by calling “<function_name>(<pedal_angle>)” ==> returns <joint_angle>

evaluate_current_pedal_angle (*current_point current_point*)

Documentation for a function.

Evaluating the current pedal-angle according to the current position of the left foot . Using trigonometric functions for the evaluation of the angle.

publish_velocity (*joint_name joint_name, next_joint_angle next_joint_angle, current_joint_angle current_joint_angle, end_time end_time*)

Documentation for a function.

For joint :

- Evaluate ideal velocity in rad/s according to joint-angle-difference and end-time of transition
- Multiply value with (global variable) to receive velocity in rad/s
- Multiply value with (global variable) to slow down simulation time.
- Multiply value with (global variable) of correspondent joint to erase joint-error
- Publish velocity to joint-controller and sleep until end-time of transition

update_velocity (*velocity_Twist velocity_Twist*)

Documentation for a function.

Updates the global variables , and TRAJECTORY_POINT_DURATION when a bike-velocity gets published to the topic “cmd_vel”.

get_angle_difference (*angle_1 angle_1, angle_2 angle_2*)

Documentation for a function.

Returns the absolute difference of two angles within the interval $[0; 2\pi]$

control_pedaling ()

Documentation for a function.

Controls the whole pedaling-process.

Evaluates next pedal-angle according to current pedal-angle and (global variable). Uses to compute joint-velocities for every joint-angle for every transition between two. trajectory points.

Use Threads to simultaneously publish and control joint-angles.

Computes joint-angle-error and adjusts error-factors for every joint to optimize ideal joint-velocity for further transitions.

Simplified Pseudo-Code:

while bike_velocity = 0:

```
for joint in joints:
    publish(0)
sleep()
```

current_pedal_angle = get_current_pedal_angle(left_foot_position)

next_pedal_angle = current_pedal_angle + (2pi / number_circulation_points)

for joint in joints:

```
next_joint_angle = interpolation_function(next_pedal_angle)
```

```
Thread.publish_velocity(joint_name, current_joint_angle, next_joint_angle,
↪end_time)
```

for Thread in created_threads:

```
Thread.join
```

for joint in joints:

```
update_error_factor()
```

main()

Documentation for a function.

Initializes the Control-Node for Pedaling and starts Pedaling-Algorithm.

Variables

```
bool finals_simulation_pedaling.PRINT_DEBUG = True
int finals_simulation_pedaling.SIMULATION_FACTOR = 100
int finals_simulation_pedaling.NUMBER_CIRCULATION_POINTS = 15
string finals_simulation_pedaling.RECORDED_TRAJECTORY_FILENAME = "trajectory_pedaling"
float finals_simulation_pedaling.JOINT_VELOCITY_FACTOR_SIMULATION = 0.008
int finals_simulation_pedaling.CONTROLLER_FREQUENCY = 100
float finals_simulation_pedaling.MIN_JOINT_VEL = -500.0
float finals_simulation_pedaling.MAX_JOINT_VEL = 500.0
int finals_simulation_pedaling.MIN_PEDAL_ANGLE = 0
float finals_simulation_pedaling.RADIUS_BACK_TIRE = 0.294398
float finals_simulation_pedaling.RADIUS_GEAR_CLUSTER = 0.06
float finals_simulation_pedaling.RADIUS_FRONT_CHAIN_RING = 0.075
float finals_simulation_pedaling.BIKE_VELOCITY = 0.0
float finals_simulation_pedaling.PEDAL_SINGLE_ROTATION_DURATION = 0.0
float finals_simulation_pedaling.TRAJECTORY_POINT_DURATION = 0.0
string finals_simulation_pedaling.ROS_JOINT_HIP_RIGHT = "joint_hip_right"
string finals_simulation_pedaling.ROS_JOINT_KNEE_RIGHT = "joint_knee_right"
string finals_simulation_pedaling.ROS_JOINT_ANKLE_RIGHT = "joint_foot_right"
string finals_simulation_pedaling.ROS_JOINT_HIP_LEFT = "joint_hip_left"
string finals_simulation_pedaling.ROS_JOINT_KNEE_LEFT = "joint_knee_left"
string finals_simulation_pedaling.ROS_JOINT_ANKLE_LEFT = "joint_foot_left"
string finals_simulation_pedaling.RIGHT_HIP_JOINT = "right_hip"
string finals_simulation_pedaling.RIGHT_KNEE_JOINT = "right_knee"
string finals_simulation_pedaling.RIGHT_ANKLE_JOINT = "right_ankle"
string finals_simulation_pedaling.LEFT_HIP_JOINT = "left_hip"
string finals_simulation_pedaling.LEFT_KNEE_JOINT = "left_knee"
string finals_simulation_pedaling.LEFT_ANKLE_JOINT = "left_ankle"
```

```

list finals_simulation_pedaling.x_pedal_record = [ ]
list finals_simulation_pedaling.y_pedal_record = [ ]
dictionary finals_simulation_pedaling.joint_trajectories_recorded= { "pedal_angle"
finals_simulation_pedaling.f_interpolated_hip_right = None
finals_simulation_pedaling.f_interpolated_hip_left = None
finals_simulation_pedaling.f_interpolated_knee_right = None
finals_simulation_pedaling.f_interpolated_knee_left = None
finals_simulation_pedaling.f_interpolated_ankle_right = None
finals_simulation_pedaling.f_interpolated_ankle_left = None
finals_simulation_pedaling.f_interpolated_pedal_angle = None
float finals_simulation_pedaling.velocity_error_factor_hip = 1.0
float finals_simulation_pedaling.velocity_error_factor_knee = 1.0
float finals_simulation_pedaling.velocity_error_factor_ankle = 1.0
int finals_simulation_pedaling.velocity_error_counter = 0
finals_simulation_pedaling.ros_right_hip_publisher = rospy.Publisher('/joint_hip_right')
finals_simulation_pedaling.ros_right_knee_publisher = rospy.Publisher('/joint_knee_right')
finals_simulation_pedaling.ros_right_ankle_publisher = rospy.Publisher('/joint_foot_right')
finals_simulation_pedaling.ros_left_hip_publisher = rospy.Publisher('/joint_hip_left')
finals_simulation_pedaling.ros_left_knee_publisher = rospy.Publisher('/joint_knee_left')
finals_simulation_pedaling.ros_left_ankle_publisher = rospy.Publisher('/joint_foot_left')
float finals_simulation_pedaling.PEDAL_CENTER_OFFSET_X = 0.09222872619138603
float finals_simulation_pedaling.PEDAL_CENTER_OFFSET_Y = 0.00013171801209023543
float finals_simulation_pedaling.PEDAL_CENTER_OFFSET_Z = 0.0042772834965418725
list finals_simulation_pedaling._jointsList= [ LEFT_HIP_JOINT, LEFT_KNEE_JOINT,
dictionary finals_simulation_pedaling._parametersRightHip= { "param_p": 1500.0,
dictionary finals_simulation_pedaling._parametersRightKnee= { "param_p": 2000.0,
dictionary finals_simulation_pedaling._parametersRightAnkle= { "param_p": 1000.0,
dictionary finals_simulation_pedaling._parametersLeftHip= { "param_p": 1500.0,
dictionary finals_simulation_pedaling._parametersLeftKnee= { "param_p": 2000.0,
dictionary finals_simulation_pedaling._parametersLeftAnkle= { "param_p": 1000.0,
dictionary finals_simulation_pedaling._jointsControlData= { RIGHT_HIP_JOINT: _parametersRightHip,
dictionary finals_simulation_pedaling.joint_status_data= { RIGHT_HIP_JOINT: {
int finals_simulation_pedaling.number_imported_trajectory_points = -1
int finals_simulation_pedaling.trajectoryStartingPoint = 0
list finals_simulation_pedaling.pedalTrajectoryRight = [ ]
list finals_simulation_pedaling.pedalAngleTrajectoryLeft = [ ]

```

```
list finals_simulation_pedaling.pedalTrajectoryLeft = [ ]
list finals_simulation_pedaling.hipTrajectoryRight = [ ]
list finals_simulation_pedaling.kneeTrajectoryRight = [ ]
list finals_simulation_pedaling.ankleTrajectoryRight = [ ]
list finals_simulation_pedaling.hipTrajectoryLeft = [ ]
list finals_simulation_pedaling.kneeTrajectoryLeft = [ ]
list finals_simulation_pedaling.ankleTrajectoryLeft = [ ]
```

```
namespace hardware_interface
```

```
namespace interpolation
```

Functions

```
importJointTrajectoryRecord()
```

Variables

```
string interpolation.RECORDED_TRAJECTORY_FILENAME = "capture_trajectory/captured_traj
bool interpolation.PRINT_DEBUG = False
int interpolation.numTrajectoryPoints = 0
int interpolation.trajectoryStartingPoint = 0
list interpolation.pedalTrajectoryRight = [ ]
list interpolation.pedalAngleTrajectoryRight = []
list interpolation.pedalTrajectoryLeft = [ ]
list interpolation.hipTrajectoryRight = [ ]
list interpolation.kneeTrajectoryRight = [ ]
list interpolation.ankleTrajectoryRight = [ ]
list interpolation.hipTrajectoryLeft = [ ]
list interpolation.kneeTrajectoryLeft = [ ]
list interpolation.ankleTrajectoryLeft = [ ]
list interpolation.pedalTrajectory = pedalTrajectoryRight
list interpolation.hipTrajectory = hipTrajectoryRight
list interpolation.x = []
list interpolation.y = []
list interpolation.z = []
list interpolation.x_new = []
list interpolation.y_new = []
list interpolation.z_new = []
interpolation.a = range(0, 341, 10)
```

```

interpolation.f = interpolate.interp1d(a, hipTrajectory, kind = "cubic")
interpolation.f1 = interpolate.interp2d(x, y, z, kind='cubic')
interpolation.f2 = interpolate.interp2d(x, y, z, kind='linear')
list interpolation.interp1 = []
list interpolation.interp2 = []
list interpolation.interp3 = []
interpolation.a_new = np.arange(0,340, 0.05)
interpolation.some_value = f(a_new)
list interpolation.temp1 = []
list interpolation.temp2 = []
list interpolation.temp3 = []
int interpolation.mean_error1 = 0
interpolation.max_error1 = 0
int interpolation.mean_error2 = 0
interpolation.max_error2 = 0
int interpolation.mean_error3 = 0
int interpolation.max_error3 = 0
interpolation.current_error = np.abs(temp1[i ] - z_new[i ])
namespace joint_angle_velocity_factor_test

```

Functions

```

jointStateCallback (joint_data joint_data)
    UTILITY FUNCTIONS ###.

importJointTrajectoryRecord()

getJointPosition (jointName jointName)

getJointVelocity (jointName jointName)

getPosition (endeffector endeffector, frame frame)

getPositionLeftFoot ()

getPositionRightFoot ()

getDistance (point1 point1, point2 point2)

setPedalSingleRotationDuration (new_duration_seconds new_duration_seconds)

setTrajectoryPointDuration ()

interpolateTrajectoryPoints (value1 value1, value2 value2, startTime startTime, currTime
                             currTime, endTime endTime)
    CONTROL FUNCTIONS ###.

checkOutputLimits (inputVal inputVal)

```

```

computeVelocitySetpoint (jointName jointName, next_joint_angle next_joint_angle, cur-
                        rent_joint_angle current_joint_angle, startTime startTime, currTime
                        currTime, endTime endTime)

get_joint_angle (thisJointName thisJointName, trajectory_angle trajectory_angle)

interpolate_functions ()

evaluate_current_angle (current_point current_point)

publish_velocity (thisJointName thisJointName, next_joint_angle next_joint_angle, cur-
                  rent_joint_angle current_joint_angle, _startTime _startTime, _currTime
                  _currTime, _endTime _endTime)

FSM()

main()
    MAIN ###.

```

Variables

```

bool joint_angle_velocity_factor_test.PRINT_DEBUG = False
    MODULE PARAMETERS ###.

string joint_angle_velocity_factor_test.RECORDED_TRAJECTORY_FILENAME = "capture_trajec

float joint_angle_velocity_factor_test.PEDAL_POSITION_ERROR_TOLERANCE = 0.02

float joint_angle_velocity_factor_test.JOINT_TRAJECTORY_ERROR_TOLERANCE = 0.02

int joint_angle_velocity_factor_test.PEDAL_SINGLE_ROTATION_DURATION = 20

int joint_angle_velocity_factor_test.NUMBER_CIRCULATION_POINTS = 72

int joint_angle_velocity_factor_test.TRAJECTORY_POINT_DURATION = 10

int joint_angle_velocity_factor_test.CONTROLLER_FREQUENCY = 100

int joint_angle_velocity_factor_test.MIN_JOINT_VEL = -500

int joint_angle_velocity_factor_test.MAX_JOINT_VEL = 500

int joint_angle_velocity_factor_test.JOINT_VELOCITY_FACTOR = 1000

list joint_angle_velocity_factor_test.x_pedal_record = [ ]
    GLOBAL VARIABLES ###.

list joint_angle_velocity_factor_test.y_pedal_record = [ ]

string joint_angle_velocity_factor_test.ROS_JOINT_HIP_RIGHT = "joint_hip_right"

string joint_angle_velocity_factor_test.ROS_JOINT_KNEE_RIGHT = "joint_knee_right"

string joint_angle_velocity_factor_test.ROS_JOINT_ANKLE_RIGHT = "joint_foot_right"

string joint_angle_velocity_factor_test.ROS_JOINT_HIP_LEFT = "joint_hip_left"

string joint_angle_velocity_factor_test.ROS_JOINT_KNEE_LEFT = "joint_knee_left"

string joint_angle_velocity_factor_test.ROS_JOINT_ANKLE_LEFT = "joint_foot_left"

string joint_angle_velocity_factor_test.RIGHT_HIP_JOINT = "right_hip"

string joint_angle_velocity_factor_test.RIGHT_KNEE_JOINT = "right_knee"

string joint_angle_velocity_factor_test.RIGHT_ANKLE_JOINT = "right_ankle"

string joint_angle_velocity_factor_test.LEFT_HIP_JOINT = "left_hip"

```

```

string joint_angle_velocity_factor_test.LEFT_KNEE_JOINT = "left_knee"
string joint_angle_velocity_factor_test.LEFT_ANKLE_JOINT = "left_ankle"
joint_angle_velocity_factor_test.f_interpolated_hip_right = None
joint_angle_velocity_factor_test.f_interpolated_hip_left = None
joint_angle_velocity_factor_test.f_interpolated_knee_right = None
joint_angle_velocity_factor_test.f_interpolated_knee_left = None
joint_angle_velocity_factor_test.f_interpolated_ankle_right = None
joint_angle_velocity_factor_test.f_interpolated_ankle_left = None
joint_angle_velocity_factor_test.f_interpolated_pedal_angle = None
float joint_angle_velocity_factor_test.velocity_error_factor_hip = 1.0
float joint_angle_velocity_factor_test.velocity_error_factor_knee = 1.0
float joint_angle_velocity_factor_test.velocity_error_factor_ankle = 1.0
joint_angle_velocity_factor_test.ros_right_hip_publisher = rospy.Publisher('/joint_hip')
joint_angle_velocity_factor_test.ros_right_knee_publisher = rospy.Publisher('/joint_knee')
joint_angle_velocity_factor_test.ros_right_ankle_publisher = rospy.Publisher('/joint_ankle')
joint_angle_velocity_factor_test.ros_left_hip_publisher = rospy.Publisher('/joint_hip')
joint_angle_velocity_factor_test.ros_left_knee_publisher = rospy.Publisher('/joint_knee')
joint_angle_velocity_factor_test.ros_left_ankle_publisher = rospy.Publisher('/joint_ankle')
float joint_angle_velocity_factor_test.PEDAL_CENTER_OFFSET_X = 0.20421
float joint_angle_velocity_factor_test.PEDAL_CENTER_OFFSET_Y = -0.00062
float joint_angle_velocity_factor_test.PEDAL_CENTER_OFFSET_Z = 0.2101
list joint_angle_velocity_factor_test._jointsList= [ RIGHT_HIP_JOINT, RIGHT_KNEE_JOINT, RIGHT_ANKLE_JOINT ]
dictionary joint_angle_velocity_factor_test._parametersRightHip= { "param_p": 1500.0 }
dictionary joint_angle_velocity_factor_test._parametersRightKnee= { "param_p": 2000.0 }
dictionary joint_angle_velocity_factor_test._parametersRightAnkle= { "param_p": 1000.0 }
dictionary joint_angle_velocity_factor_test._parametersLeftHip= { "param_p": 1500.0 }
dictionary joint_angle_velocity_factor_test._parametersLeftKnee= { "param_p": 2000.0 }
dictionary joint_angle_velocity_factor_test._parametersLeftAnkle= { "param_p": 1000.0 }
dictionary joint_angle_velocity_factor_test._jointsControlData= { RIGHT_HIP_JOINT: 0.0, RIGHT_KNEE_JOINT: 0.0, RIGHT_ANKLE_JOINT: 0.0, LEFT_HIP_JOINT: 0.0, LEFT_KNEE_JOINT: 0.0, LEFT_ANKLE_JOINT: 0.0 }
dictionary joint_angle_velocity_factor_test._jointsStatusData= { RIGHT_HIP_JOINT: 0.0, RIGHT_KNEE_JOINT: 0.0, RIGHT_ANKLE_JOINT: 0.0, LEFT_HIP_JOINT: 0.0, LEFT_KNEE_JOINT: 0.0, LEFT_ANKLE_JOINT: 0.0 }
int joint_angle_velocity_factor_test.number_imported_trajectory_points = -1
int joint_angle_velocity_factor_test.trajectoryStartingPoint = 0
list joint_angle_velocity_factor_test.pedalTrajectoryRight = [ ]
list joint_angle_velocity_factor_test.pedalAngleTrajectoryRight = [ ]
list joint_angle_velocity_factor_test.pedalTrajectoryLeft = [ ]
list joint_angle_velocity_factor_test.hipTrajectoryRight = [ ]

```

```
list joint_angle_velocity_factor_test.kneeTrajectoryRight = [ ]
list joint_angle_velocity_factor_test.ankleTrajectoryRight = [ ]
list joint_angle_velocity_factor_test.hipTrajectoryLeft = [ ]
list joint_angle_velocity_factor_test.kneeTrajectoryLeft = [ ]
list joint_angle_velocity_factor_test.ankleTrajectoryLeft = [ ]
string joint_angle_velocity_factor_test.INIT = "INIT"
    STATE MACHINE ###.
string joint_angle_velocity_factor_test.PEDAL = "PEDAL"
string joint_angle_velocity_factor_test.UPDATE_PARAMETERS = "UPDATE_PARAMETERS"
namespace kp_kp
```

Functions

```
setJointControllerParameters (proportionalVal proportionalVal, derivativeVal derivativeVal)
main()
```

Variables

```
string kp_kp.JOINT_SHOULDER_AXIS0_RIGHT = "right_shoulder_axis0"
string kp_kp.JOINT_SHOULDER_AXIS1_RIGHT = "right_shoulder_axis1"
string kp_kp.JOINT_SHOULDER_AXIS2_RIGHT = "right_shoulder_axis2"
string kp_kp.JOINT_SHOULDER_AXIS0_LEFT = "left_shoulder_axis0"
string kp_kp.JOINT_SHOULDER_AXIS1_LEFT = "left_shoulder_axis1"
string kp_kp.JOINT_SHOULDER_AXIS2_LEFT = "left_shoulder_axis2"
string kp_kp.JOINT_ELBOW_RIGHT = "elbow_right"
string kp_kp.JOINT_ELBOW_LEFT = "elbow_left"
string kp_kp.JOINT_WRIST_RIGHT_SPHERE_AXIS0 = "wrist_right_sphere_axis0"
string kp_kp.JOINT_WRIST_RIGHT_SPHERE_AXIS1 = "wrist_right_sphere_axis1"
string kp_kp.JOINT_WRIST_RIGHT_SPHERE_AXIS2 = "wrist_right_sphere_axis2"
string kp_kp.JOINT_WRIST_LEFT_SPHERE_AXIS0 = "wrist_left_sphere_axis0"
string kp_kp.JOINT_WRIST_LEFT_SPHERE_AXIS1 = "wrist_left_sphere_axis1"
string kp_kp.JOINT_WRIST_LEFT_SPHERE_AXIS2 = "wrist_left_sphere_axis2"
string kp_kp.ROS_JOINT_HIP_RIGHT = "joint_hip_right"
string kp_kp.ROS_JOINT_KNEE_RIGHT = "joint_knee_right"
string kp_kp.ROS_JOINT_ANKLE_RIGHT = "joint_foot_right"
string kp_kp.ROS_JOINT_HIP_LEFT = "joint_hip_left"
string kp_kp.ROS_JOINT_KNEE_LEFT = "joint_knee_left"
string kp_kp.ROS_JOINT_ANKLE_LEFT = "joint_foot_left"
```

```

computeSteeringAngles ()
    UTILITY FUNCTIONS ###.

computeHandTrajectories ()

getPositionLeftHand ()

getPositionRightHand ()

setJointControllerParameters (proportionalVal proportionalVal, derivativeVal derivativeVal)

jointStateCallback (joint_data joint_data)

inverse_kinematics_client (endeffector endeffector, frame frame, x x, y y, z z, roll roll, pitch
    pitch, yaw yaw)

main ()
    MAIN ###.

```

```
int new_hand_steering_capture_trajectory.MAX_TURNING_ANGLE =  math.pi / 15
FUNCTION PARAMETERS ###.

int new_hand_steering_capture_trajectory.NUM_STEERING_ANGLES = 61

float new_hand_steering_capture_trajectory.RIKSHAW_TURN_JOINT_X_OFFSET = -0.230035468
float new_hand_steering_capture_trajectory.RIKSHAW_TURN_JOINT_Y_OFFSET = -0.010388308
float new_hand_steering_capture_trajectory.RIKSHAW_TURN_JOINT_Z_OFFSET = -0.205270696

int new_hand_steering_capture_trajectory.YAW_RIGHT_HAND_OFFSET =  math.pi / 2 + math.p
int new_hand_steering_capture_trajectory.YAW_LEFT_HAND_OFFSET = 3 * math.pi / 2 + mat
float new_hand_steering_capture_trajectory.HANDLEBAR_X_OFFSET = 0.728713
float new_hand_steering_capture_trajectory.HANDLEBAR_Z_OFFSET = 0.719269
float new_hand_steering_capture_trajectory.HAND_Y_OFFSET = 0.2

string new_hand_steering_capture_trajectory.JSON_FILENAME = "new_hand_steering_trajec
float new_hand_steering_capture_trajectory.JOINT_ANGLE_TOLERANCE_FK = 0.01
string new_hand_steering_capture_trajectory.ENDEFFECTOR_RIGHT = "right_arm"
string new_hand_steering_capture_trajectory.FRAME_RIGHT = "wrist_right_sphere_link1"
string new_hand_steering_capture_trajectory.ENDEFFECTOR_LEFT = "left_arm"
string new_hand_steering_capture_trajectory.FRAME_LEFT = "wrist_left_sphere_link1"

int new_hand_steering_capture_trajectory.BIKE_OFFSET_X = 0
MEASURED PARAMETERS ###.

int new_hand_steering_capture_trajectory.BIKE_OFFSET_Y = 0
int new_hand_steering_capture_trajectory.BIKE_OFFSET_Z = 0
```

```

list new_hand_steering_capture_trajectory._steeringAngles = []
    GLOBAL VARIABLES ###.

list new_hand_steering_capture_trajectory._rightHandTrajectory = []
list new_hand_steering_capture_trajectory._leftHandTrajectory = []
list new_hand_steering_capture_trajectory._centerHandlebarTrajectory = []
string new_hand_steering_capture_trajectory.JOINT_SHOULDER_AXIS0_RIGHT = "right_shoul
string new_hand_steering_capture_trajectory.JOINT_SHOULDER_AXIS1_RIGHT = "right_shoul
string new_hand_steering_capture_trajectory.JOINT_SHOULDER_AXIS2_RIGHT = "right_shoul
string new_hand_steering_capture_trajectory.JOINT_SHOULDER_AXIS0_LEFT = "left_shoulde
string new_hand_steering_capture_trajectory.JOINT_SHOULDER_AXIS1_LEFT = "left_shoulde
string new_hand_steering_capture_trajectory.JOINT_SHOULDER_AXIS2_LEFT = "left_shoulde
string new_hand_steering_capture_trajectory.JOINT_ELBOW_RIGHT = "elbow_right"
string new_hand_steering_capture_trajectory.JOINT_ELBOW_LEFT = "elbow_left"
string new_hand_steering_capture_trajectory.JOINT_WRIST_RIGHT_SPHERE_AXIS0 = "wrist_r
string new_hand_steering_capture_trajectory.JOINT_WRIST_LEFT_SPHERE_AXIS0 = "wrist_le
list new_hand_steering_capture_trajectory.JOINT_LIST= [JOINT_SHOULDER_AXIS0_RIGHT, JO
dictionary new_hand_steering_capture_trajectory._jointsStatusData

namespace pedal_simulation

```

Functions

joint_state_callback (*joint_data joint_data*)

Documentation for a function.

This function collects the current status of the joint-angles and saves them in the global dictionary “joint_status_data”.

import_joint_trajectory_record ()

Documentation for a function.

Collects and saves all joint- and pedal-angles from the pre-captured trajectory from the (global variable).

get_joint_position (*jointName jointName*)

Documentation for a function.

Return current joint-angle of .

get_joint_velocity (*jointName jointName*)

Documentation for a function.

Return current joint-velocity of .

get_position (*endeffector endeffector, frame frame*)

Documentation for a function.

Return the postion of the of the correspondent .

set_joint_controller_parameters (*proportionalVal proportionalVal, derivativeVal deriva-
tiveVal*)

Documentation for a function.

Sets Kp of joint-controller to Sets Kd of joint-controller to

get_position_left_foot ()

Documentation for a function.

Return the position of the left foot of Roboy.

get_position_right_foot ()

Documentation for a function.

Return the position of the right foot of Roboy.

get_distance (*point1 point1, point2 point2*)

Documentation for a function.

Return the distance between two points where points are a list of two coordinates.

check_output_limits (*inputVal inputVal*)

Documentation for a function.

Checks if is inside possible range of joint-angle velocities. If not, returns max-velocity if too high or min-velocity if to low instead of .

compute_velocity (*joint_name joint_name, next_joint_angle next_joint_angle, current_joint_angle current_joint_angle, end_time end_time*)

Documentation for a function.

Returns ideal joint-velocity for according to and joint-angle-difference between and :

$$\text{ideal_velocity} = \text{joint_angle_difference} / (\text{end_time} - \text{current_time})$$

get_joint_angle (*joint_name joint_name, pedal_angle pedal_angle*)

Documentation for a function.

Evaluate and return joint-angle of correspondent to using the interpolated function:

The functions can be used by calling “<function_name>(<pedal_angle>)” ==> returns <joint_angle>

interpolate_functions ()

Documentation for a function.

Initializing interpolated functions for joint-angles using pre-captured set points with pedal-angle as input and joint-angle as output:

The functions can be used by calling “<function_name>(<pedal_angle>)” ==> returns <joint_angle>

evaluate_current_pedal_angle (*current_point current_point*)

Documentation for a function.

Evaluating the current pedal-angle according to the current position of the left foot . Using trigonometric functions for the evaluation of the angle.

publish_velocity (*joint_name joint_name, next_joint_angle next_joint_angle, current_joint_angle current_joint_angle, end_time end_time*)

Documentation for a function.

For joint :

- Evaluate ideal velocity in rad/s according to joint-angle-difference and end-time of transition
- Multiply value with (global variable) to receive velocity in rad/s
- Multiply value with (global variable) to slow down simulation time.
- Multiply value with (global variable) of correspondent joint to erase joint-error
- Publish velocity to joint-controller and sleep until end-time of transition

update_velocity (*velocity_Twist velocity_Twist*)

Documentation for a function.

Updates the global variables , and TRAJECTORY_POINT_DURATION when a bike-velocity gets published to the topic “cmd_vel”.

get_angle_difference (*angle_1 angle_1, angle_2 angle_2*)

Documentation for a function.

Returns the absolute difference of two angles within the interval $[0; 2\pi]$

control_pedaling ()

Documentation for a function.

Controls the whole pedaling-process.

Evaluates next pedal-angle according to current pedal-angle and (global variable). Uses to compute joint-velocities for every joint-angle for every transition between two. trajectory points.

Use Threads to simultaneously publish and control joint-angles.

Computes joint-angle-error and adjusts error-factors for every joint to optimize ideal joint-velocity for further transitions.

Simplified Pseudo-Code:

while bike_velocity = 0:

```
for joint in joints:
    publish(0)
    sleep()
```

current_pedal_angle = get_current_pedal_angle(left_foot_position)

next_pedal_angle = current_pedal_angle + (2pi / number_circulation_points)

for joint in joints:

```
next_joint_angle = interpolation_function(next_pedal_angle)
Thread.publish_velocity(joint_name, current_joint_angle, next_joint_angle,
↪end_time)
```

for Thread in created_threads:

```
Thread.join
```

for joint in joints:

```
update_error_factor()
```

main ()

Documentation for a function.

Initializes the Control-Node for Pedaling and starts Pedaling-Algorithm.

Variables

```
bool pedal_simulation.PRINT_DEBUG = True
```

```
float pedal_simulation.SIMULATION_FACTOR = 100.0
int pedal_simulation.NUMBER_CIRCULATION_POINTS = 30
string pedal_simulation.RECORDED_TRAJECTORY_FILENAME = "trajectory_pedaling/captured_
float pedal_simulation.JOINT_VELOCITY_FACTOR_SIMULATION = 0.01
float pedal_simulation.PEDAL_POSITION_ERROR_TOLERANCE = 0.02
float pedal_simulation.JOINT_TRAJECTORY_ERROR_TOLERANCE = 0.02
int pedal_simulation.CONTROLLER_FREQUENCY = 100
float pedal_simulation.MIN_JOINT_VEL = -500.0
float pedal_simulation.MAX_JOINT_VEL = 500.0
float pedal_simulation.RADIUS_BACK_TIRE = 0.294398
float pedal_simulation.RADIUS_GEAR_CLUSTER = 0.06
float pedal_simulation.RADIUS_FRONT_CHAIN_RING = 0.075
float pedal_simulation.BIKE_VELOCITY = 0.0
float pedal_simulation.PEDAL_SINGLE_ROTATION_DURATION = 0.0
float pedal_simulation.TRAJECTORY_POINT_DURATION = 0.0
list pedal_simulation.x_pedal_record = [ ]
list pedal_simulation.y_pedal_record = [ ]
string pedal_simulation.ROS_JOINT_HIP_RIGHT = "joint_hip_right"
string pedal_simulation.ROS_JOINT_KNEE_RIGHT = "joint_knee_right"
string pedal_simulation.ROS_JOINT_ANKLE_RIGHT = "joint_foot_right"
string pedal_simulation.ROS_JOINT_HIP_LEFT = "joint_hip_left"
string pedal_simulation.ROS_JOINT_KNEE_LEFT = "joint_knee_left"
string pedal_simulation.ROS_JOINT_ANKLE_LEFT = "joint_foot_left"
string pedal_simulation.RIGHT_HIP_JOINT = "right_hip"
string pedal_simulation.RIGHT_KNEE_JOINT = "right_knee"
string pedal_simulation.RIGHT_ANKLE_JOINT = "right_ankle"
string pedal_simulation.LEFT_HIP_JOINT = "left_hip"
string pedal_simulation.LEFT_KNEE_JOINT = "left_knee"
string pedal_simulation.LEFT_ANKLE_JOINT = "left_ankle"
pedal_simulation.f_interpolated_hip_right = None
pedal_simulation.f_interpolated_hip_left = None
pedal_simulation.f_interpolated_knee_right = None
pedal_simulation.f_interpolated_knee_left = None
pedal_simulation.f_interpolated_ankle_right = None
pedal_simulation.f_interpolated_ankle_left = None
pedal_simulation.f_interpolated_pedal_angle = None
```

```

float pedal_simulation.velocity_error_factor_hip = 1.0
float pedal_simulation.velocity_error_factor_knee = 1.0
float pedal_simulation.velocity_error_factor_ankle = 1.0
int pedal_simulation.velocity_error_counter = 0
pedal_simulation.ros_right_hip_publisher = rospy.Publisher('/joint_hip_right/joint_hip')
pedal_simulation.ros_right_knee_publisher = rospy.Publisher('/joint_knee_right/joint_knee')
pedal_simulation.ros_right_ankle_publisher = rospy.Publisher('/joint_foot_right/joint_foot')
pedal_simulation.ros_left_hip_publisher = rospy.Publisher('/joint_hip_left/joint_hip')
pedal_simulation.ros_left_knee_publisher = rospy.Publisher('/joint_knee_left/joint_knee')
pedal_simulation.ros_left_ankle_publisher = rospy.Publisher('/joint_foot_left/joint_foot')
float pedal_simulation.PEDAL_CENTER_OFFSET_X = 0.20421
float pedal_simulation.PEDAL_CENTER_OFFSET_Y = -0.00062
float pedal_simulation.PEDAL_CENTER_OFFSET_Z = 0.2101
list pedal_simulation._jointsList= [ RIGHT_HIP_JOINT, RIGHT_KNEE_JOINT, RIGHT_ANKLE_JOINT, LEFT_HIP_JOINT, LEFT_KNEE_JOINT, LEFT_ANKLE_JOINT ]
dictionary pedal_simulation._parametersRightHip= { "param_p": 1500.0, "param_i": 0.0, "param_d": 0.0 }
dictionary pedal_simulation._parametersRightKnee= { "param_p": 2000.0, "param_i": 0.0, "param_d": 0.0 }
dictionary pedal_simulation._parametersRightAnkle= { "param_p": 1000.0, "param_i": 0.0, "param_d": 0.0 }
dictionary pedal_simulation._parametersLeftHip= { "param_p": 1500.0, "param_i": 0.0, "param_d": 0.0 }
dictionary pedal_simulation._parametersLeftKnee= { "param_p": 2000.0, "param_i": 0.0, "param_d": 0.0 }
dictionary pedal_simulation._parametersLeftAnkle= { "param_p": 1000.0, "param_i": 0.0, "param_d": 0.0 }
dictionary pedal_simulation._jointsControlData= { RIGHT_HIP_JOINT: _parametersRightHip, RIGHT_KNEE_JOINT: _parametersRightKnee, RIGHT_ANKLE_JOINT: _parametersRightAnkle, LEFT_HIP_JOINT: _parametersLeftHip, LEFT_KNEE_JOINT: _parametersLeftKnee, LEFT_ANKLE_JOINT: _parametersLeftAnkle }
dictionary pedal_simulation.joint_status_data= { RIGHT_HIP_JOINT: { "Pos": 0.0, "Vel": 0.0, "Acc": 0.0, "DesiredPos": 0.0, "DesiredVel": 0.0, "DesiredAcc": 0.0 }, RIGHT_KNEE_JOINT: { "Pos": 0.0, "Vel": 0.0, "Acc": 0.0, "DesiredPos": 0.0, "DesiredVel": 0.0, "DesiredAcc": 0.0 }, RIGHT_ANKLE_JOINT: { "Pos": 0.0, "Vel": 0.0, "Acc": 0.0, "DesiredPos": 0.0, "DesiredVel": 0.0, "DesiredAcc": 0.0 }, LEFT_HIP_JOINT: { "Pos": 0.0, "Vel": 0.0, "Acc": 0.0, "DesiredPos": 0.0, "DesiredVel": 0.0, "DesiredAcc": 0.0 }, LEFT_KNEE_JOINT: { "Pos": 0.0, "Vel": 0.0, "Acc": 0.0, "DesiredPos": 0.0, "DesiredVel": 0.0, "DesiredAcc": 0.0 }, LEFT_ANKLE_JOINT: { "Pos": 0.0, "Vel": 0.0, "Acc": 0.0, "DesiredPos": 0.0, "DesiredVel": 0.0, "DesiredAcc": 0.0 } }
int pedal_simulation.number_imported_trajectory_points = -1
int pedal_simulation.trajectoryStartingPoint = 0
list pedal_simulation.pedalTrajectoryRight = [ ]
list pedal_simulation.pedalAngleTrajectoryRight = [ ]
list pedal_simulation.pedalTrajectoryLeft = [ ]
list pedal_simulation.hipTrajectoryRight = [ ]
list pedal_simulation.kneeTrajectoryRight = [ ]
list pedal_simulation.ankleTrajectoryRight = [ ]
list pedal_simulation.hipTrajectoryLeft = [ ]
list pedal_simulation.kneeTrajectoryLeft = [ ]
list pedal_simulation.ankleTrajectoryLeft = [ ]
namespace pedal_simulation_interpolation_cubic_derivative

```

Functions

```

jointStateCallback (joint_data joint_data)
    UTILITY FUNCTIONS ###.

setJointControllerParameters (proportionalVal proportionalVal, derivativeVal derivativeVal)

importJointTrajectoryRecord ()

getJointPosition (jointName jointName)

getJointVelocity (jointName jointName)

getPositionLeftFoot ()

getPositionRightFoot ()

getDistance (point1 point1, point2 point2)

setPedalSingleRotationDuration (new_duration_seconds new_duration_seconds)

setTrajectoryPointDuration ()

setPedalAngularVelocity ()

getCurrentAngle (current_point current_point)

interpolateAllJointPositions ()

printInterpolatedFunctions ()

checkOutputLimits (inputVal inputVal)
    CONTROL FUNCTIONS ###.

FSM ()
    STATE MACHINE ###.

main ()
    MAIN ###.

```

Variables

```

bool pedal_simulation_interpolation_cubic_derivative.PRINT_DEBUG =  True
    MODULE PARAMETERS ###.

string pedal_simulation_interpolation_cubic_derivative.RECORDED_TRAJECTORY_FILENAME =
float pedal_simulation_interpolation_cubic_derivative.PEDAL_POSITION_ERROR_TOLERANCE =
float pedal_simulation_interpolation_cubic_derivative.PEDAL_ANGLE_ERROR_TOLERANCE =  0
float pedal_simulation_interpolation_cubic_derivative.JOINT_TRAJECTORY_ERROR_TOLERANCE =
int pedal_simulation_interpolation_cubic_derivative.PEDAL_SINGLE_ROTATION_DURATION =
int pedal_simulation_interpolation_cubic_derivative.CONTROLLER_FREQUENCY =  100
int pedal_simulation_interpolation_cubic_derivative.MIN_JOINT_VEL =  -50
int pedal_simulation_interpolation_cubic_derivative.MAX_JOINT_VEL =  50
int pedal_simulation_interpolation_cubic_derivative.JOINT_VELOCITY_FACTOR =  1
float pedal_simulation_interpolation_cubic_derivative.PEDAL_CENTER_OFFSET_X =  0.20421
float pedal_simulation_interpolation_cubic_derivative.PEDAL_CENTER_OFFSET_Y =  -0.0006

```

```

float pedal_simulation_interpolation_cubic_derivative.PEDAL_CENTER_OFFSET_Z = 0.2101
list pedal_simulation_interpolation_cubic_derivative.x_pedal_record = []
    GLOBAL VARIABLES ###.
list pedal_simulation_interpolation_cubic_derivative.y_pedal_record = []
string pedal_simulation_interpolation_cubic_derivative.ROS_JOINT_HIP_RIGHT = "joint_h
string pedal_simulation_interpolation_cubic_derivative.ROS_JOINT_KNEE_RIGHT = "joint_
string pedal_simulation_interpolation_cubic_derivative.ROS_JOINT_ANKLE_RIGHT = "joint
string pedal_simulation_interpolation_cubic_derivative.ROS_JOINT_HIP_LEFT = "joint_hip
string pedal_simulation_interpolation_cubic_derivative.ROS_JOINT_KNEE_LEFT = "joint_k
string pedal_simulation_interpolation_cubic_derivative.ROS_JOINT_ANKLE_LEFT = "joint_
string pedal_simulation_interpolation_cubic_derivative.RIGHT_HIP_JOINT = "right_hip"
string pedal_simulation_interpolation_cubic_derivative.RIGHT_KNEE_JOINT = "right_knee
string pedal_simulation_interpolation_cubic_derivative.RIGHT_ANKLE_JOINT = "right_ank
string pedal_simulation_interpolation_cubic_derivative.LEFT_HIP_JOINT = "left_hip"
string pedal_simulation_interpolation_cubic_derivative.LEFT_KNEE_JOINT = "left_knee"
string pedal_simulation_interpolation_cubic_derivative.LEFT_ANKLE_JOINT = "left_ankle
list pedal_simulation_interpolation_cubic_derivative._jointsList = [RIGHT_HIP_JOINT,
list pedal_simulation_interpolation_cubic_derivative._jointsListROS= [ROS_JOINT_HIP_R
dictionary pedal_simulation_interpolation_cubic_derivative._parametersRightHip= {
dictionary pedal_simulation_interpolation_cubic_derivative._parametersRightKnee= {
dictionary pedal_simulation_interpolation_cubic_derivative._parametersRightAnkle= {
dictionary pedal_simulation_interpolation_cubic_derivative._parametersLeftHip= {
dictionary pedal_simulation_interpolation_cubic_derivative._parametersLeftKnee= {
dictionary pedal_simulation_interpolation_cubic_derivative._parametersLeftAnkle= {
dictionary pedal_simulation_interpolation_cubic_derivative._jointsControlData= {
dictionary pedal_simulation_interpolation_cubic_derivative._jointsStatusData= {
int pedal_simulation_interpolation_cubic_derivative._numTrajectoryPoints = -1
int pedal_simulation_interpolation_cubic_derivative._trajectoryStartingPoint = 0
float pedal_simulation_interpolation_cubic_derivative._trajectoryPointDuration = 1.0
float pedal_simulation_interpolation_cubic_derivative._pedalAngularVelocity = 0.1
list pedal_simulation_interpolation_cubic_derivative._pedalTrajectoryRight = []
list pedal_simulation_interpolation_cubic_derivative._pedalAngleTrajectoryRight = []
list pedal_simulation_interpolation_cubic_derivative._hipTrajectoryRight = []
list pedal_simulation_interpolation_cubic_derivative._kneeTrajectoryRight = []
list pedal_simulation_interpolation_cubic_derivative._ankleTrajectoryRight = []
list pedal_simulation_interpolation_cubic_derivative._pedalTrajectoryLeft = []

```

```

list pedal_simulation_interpolation_cubic_derivative._pedalAngleTrajectoryLeft = []
list pedal_simulation_interpolation_cubic_derivative._hipTrajectoryLeft = []
list pedal_simulation_interpolation_cubic_derivative._kneeTrajectoryLeft = []
list pedal_simulation_interpolation_cubic_derivative._ankleTrajectoryLeft = []
namespace pedal_simulation_interpolation_linear_trajectory_points

```

Functions

```

jointStateCallback (joint_data joint_data)
    UTILITY FUNCTIONS ###.

setJointControllerParameters (proportionalVal proportionalVal, derivativeVal derivativeVal)
importJointTrajectoryRecord ()
getJointPosition (jointName jointName)
getJointVelocity (jointName jointName)
getPosition (endeffector endeffector, frame frame)
getPositionLeftFoot ()
getPositionRightFoot ()
getDistance (point1 point1, point2 point2)
setPedalSingleRotationDuration (new_duration_seconds new_duration_seconds)
setTrajectoryPointDuration ()
interpolateTrajectoryPoints (value1 value1, value2 value2, startTime startTime, currTime
                             currTime, endTime endTime)
    CONTROL FUNCTIONS ###.

checkOutputLimits (inputVal inputVal)
computeVelocitySetpoint (jointName jointName, endPos endPos, startTime startTime, currTime
                        currTime, endTime endTime)

FSM ()
main ()
    MAIN ###.

```

Variables

```

bool pedal_simulation_interpolation_linear_trajectory_points.PRINT_DEBUG = True
    MODULE PARAMETERS ###.

string pedal_simulation_interpolation_linear_trajectory_points.RECORDED_TRAJECTORY_FILE
float pedal_simulation_interpolation_linear_trajectory_points.PEDAL_POSITION_ERROR_TOL
float pedal_simulation_interpolation_linear_trajectory_points.JOINT_TRAJECTORY_ERROR_TOL
int pedal_simulation_interpolation_linear_trajectory_points.PEDAL_SINGLE_ROTATION_DURATION
int pedal_simulation_interpolation_linear_trajectory_points.TRAJECTORY_POINT_DURATION
int pedal_simulation_interpolation_linear_trajectory_points.CONTROLLER_FREQUENCY = 10

```

```

int pedal_simulation_interpolation_linear_trajectory_points.MIN_JOINT_VEL = -500
int pedal_simulation_interpolation_linear_trajectory_points.MAX_JOINT_VEL = 500
int pedal_simulation_interpolation_linear_trajectory_points.JOINT_VELOCITY_FACTOR = 1
list pedal_simulation_interpolation_linear_trajectory_points.x_pedal_record = []
    GLOBAL VARIABLES ###.
list pedal_simulation_interpolation_linear_trajectory_points.y_pedal_record = []
string pedal_simulation_interpolation_linear_trajectory_points.ROS_JOINT_HIP_RIGHT =
string pedal_simulation_interpolation_linear_trajectory_points.ROS_JOINT_KNEE_RIGHT =
string pedal_simulation_interpolation_linear_trajectory_points.ROS_JOINT_ANKLE_RIGHT =
string pedal_simulation_interpolation_linear_trajectory_points.ROS_JOINT_HIP_LEFT = "
string pedal_simulation_interpolation_linear_trajectory_points.ROS_JOINT_KNEE_LEFT =
string pedal_simulation_interpolation_linear_trajectory_points.ROS_JOINT_ANKLE_LEFT =
string pedal_simulation_interpolation_linear_trajectory_points.RIGHT_HIP_JOINT = "right"
string pedal_simulation_interpolation_linear_trajectory_points.RIGHT_KNEE_JOINT = "right"
string pedal_simulation_interpolation_linear_trajectory_points.RIGHT_ANKLE_JOINT = "right"
string pedal_simulation_interpolation_linear_trajectory_points.LEFT_HIP_JOINT = "left"
string pedal_simulation_interpolation_linear_trajectory_points.LEFT_KNEE_JOINT = "left"
string pedal_simulation_interpolation_linear_trajectory_points.LEFT_ANKLE_JOINT = "left"
list pedal_simulation_interpolation_linear_trajectory_points._jointsList = [RIGHT_HIP,
list pedal_simulation_interpolation_linear_trajectory_points._jointsListROS = [ROS_JOINT_HIP,
dictionary pedal_simulation_interpolation_linear_trajectory_points._parametersRightHip=
dictionary pedal_simulation_interpolation_linear_trajectory_points._parametersRightKnee=
dictionary pedal_simulation_interpolation_linear_trajectory_points._parametersRightAnkle=
dictionary pedal_simulation_interpolation_linear_trajectory_points._parametersLeftHip=
dictionary pedal_simulation_interpolation_linear_trajectory_points._parametersLeftKnee=
dictionary pedal_simulation_interpolation_linear_trajectory_points._parametersLeftAnkle=
dictionary pedal_simulation_interpolation_linear_trajectory_points._jointsControlData=
dictionary pedal_simulation_interpolation_linear_trajectory_points._jointsStatusData=
int pedal_simulation_interpolation_linear_trajectory_points.numTrajectoryPoints = -1
int pedal_simulation_interpolation_linear_trajectory_points.trajectoryStartingPoint =
list pedal_simulation_interpolation_linear_trajectory_points.pedalTrajectoryRight = [
list pedal_simulation_interpolation_linear_trajectory_points.pedalTrajectoryLeft = [
list pedal_simulation_interpolation_linear_trajectory_points.hipTrajectoryRight = [
list pedal_simulation_interpolation_linear_trajectory_points.kneeTrajectoryRight = [
list pedal_simulation_interpolation_linear_trajectory_points.ankleTrajectoryRight = [
list pedal_simulation_interpolation_linear_trajectory_points.hipTrajectoryLeft = [

```



```

list pedal_simulation_interpolation_linear_trajectory_points.kneeTrajectoryLeft = []
list pedal_simulation_interpolation_linear_trajectory_points.ankleTrajectoryLeft = []
string pedal_simulation_interpolation_linear_trajectory_points.INIT = "INIT"
STATE MACHINE ###.

string pedal_simulation_interpolation_linear_trajectory_points.PEDAL = "PEDAL"
string pedal_simulation_interpolation_linear_trajectory_points.UPDATE_PARAMETERS = "U

namespace pedal_simulation_left_leg_experimental

```

Functions

joint_state_callback (*joint_data joint_data*)

Documentation for a function.

This function collects the current status of the joint-angles and saves them in the global dictionary “joint_status_data”.

import_joint_trajectory_record ()

Documentation for a function.

Collects and saves all joint- and pedal-angles from the pre-captured trajectory from the (global variable).

get_joint_position (*jointName jointName*)

Documentation for a function.

Return current joint-angle of .

get_joint_velocity (*jointName jointName*)

Documentation for a function.

Return current joint-velocity of .

get_position (*endeffector endeffector, frame frame*)

Documentation for a function.

Return the position of the of the correspondent .

set_joint_controller_parameters (*proportionalVal proportionalVal, derivativeVal derivativeVal*)

Documentation for a function.

Sets Kp of joint-controller to Sets Kd of joint-controller to

get_position_left_foot ()

Documentation for a function.

Return the position of the left foot of Roboy.

get_position_right_foot ()

Documentation for a function.

Return the position of the right foot of Roboy.

get_distance (*point1 point1, point2 point2*)

Documentation for a function.

Return the distance between two points where points are a list of two coordinates.

check_output_limits (*inputVal inputVal*)

Documentation for a function.

Checks if is inside possible range of joint-angle velocities. If not, returns max-velocity if too high or min-velocity if to low instead of .

compute_velocity (*joint_name joint_name, next_joint_angle next_joint_angle, current_joint_angle current_joint_angle, end_time end_time*)

Documentation for a function.

Returns ideal joint-velocity for according to and joint-angle-difference between and :

$$\text{ideal_velocity} = \text{joint_angle_difference} / (\text{end_time} - \text{current_time})$$

get_joint_angle (*joint_name joint_name, pedal_angle pedal_angle*)

Documentation for a function.

Evaluate and return joint-angle of correspondent to using the interpolated function:

The functions can be used by calling “<function_name>(<pedal_angle>)” ==> returns <joint_angle>

interpolate_functions ()

Documentation for a function.

Initializing interpolated functions for joint-angles using pre-captured set points with pedal-angle as input and joint-angle as output:

The functions can be used by calling “<function_name>(<pedal_angle>)” ==> returns <joint_angle>

evaluate_current_pedal_angle (*current_point current_point*)

Documentation for a function.

Evaluating the current pedal-angle according to the current position of the left foot . Using trigonometric functions for the evaluation of the angle.

publish_velocity (*joint_name joint_name, next_joint_angle next_joint_angle, current_joint_angle current_joint_angle, end_time end_time*)

Documentation for a function.

For joint :

- Evaluate ideal velocity in rad/s according to joint-angle-difference and end-time of transition
- Multiply value with (global variable) to receive velocity in rad/s
- Multiply value with (global variable) to slow down simulation time.
- Multiply value with (global variable) of correspondent joint to erase joint-error
- Publish velocity to joint-controller and sleep until end-time of transition

update_velocity (*velocity_Twist velocity_Twist*)

Documentation for a function.

Updates the global variables , and TRAJECTORY_POINT_DURATION when a bike-velocity gets published to the topic “cmd_vel”.

get_angle_difference (*angle_1 angle_1, angle_2 angle_2*)

Documentation for a function.

Returns the absolute difference of two angles within the interval [0;2pi]

control_pedaling ()

Documentation for a function.

Controls the whole pedaling-process.

Evaluates next pedal-angle according to current pedal-angle and (global variable). Uses to compute joint-velocities for every joint-angle for every transition between two. trajectory points.

Use Threads to simultaneously publish and control joint-angles.

Computes joint-angle-error and adjusts error-factors for every joint to optimize ideal joint-velocity for further transitions.

Simplified Pseudo-Code:

while bike_velocity = 0:

```
for joint in joints:
    publish(0)
sleep()
```

current_pedal_angle = get_current_pedal_angle(left_foot_position)

next_pedal_angle = current_pedal_angle + (2pi / number_circulation_points)

for joint in joints:

```
next_joint_angle = interpolation_function(next_pedal_angle)
Thread.publish_velocity(joint_name, current_joint_angle, next_joint_angle,
↪end_time)
```

for Thread in created_threads:

```
Thread.join
```

for joint in joints:

```
update_error_factor()
```

main()

Documentation for a function.

Initializes the Control-Node for Pedaling and starts Pedaling-Algorithm.

Variables

```
bool pedal_simulation_left_leg_experimental.PRINT_DEBUG = True
float pedal_simulation_left_leg_experimental.SIMULATION_FACTOR = 100.0
int pedal_simulation_left_leg_experimental.NUMBER_CIRCULATION_POINTS = 30
string pedal_simulation_left_leg_experimental.RECORDED_TRAJECTORY_FILENAME = "traject
float pedal_simulation_left_leg_experimental.JOINT_VELOCITY_FACTOR_SIMULATION = 0.008
float pedal_simulation_left_leg_experimental.PEDAL_POSITION_ERROR_TOLERANCE = 0.02
float pedal_simulation_left_leg_experimental.JOINT_TRAJECTORY_ERROR_TOLERANCE = 0.02
int pedal_simulation_left_leg_experimental.CONTROLLER_FREQUENCY = 100
float pedal_simulation_left_leg_experimental.MIN_JOINT_VEL = -500.0
float pedal_simulation_left_leg_experimental.MAX_JOINT_VEL = 500.0
float pedal_simulation_left_leg_experimental.RADIUS_BACK_TIRE = 0.294398
```

```

float pedal_simulation_left_leg_experimental.RADIUS_GEAR_CLUSTER = 0.06
float pedal_simulation_left_leg_experimental.RADIUS_FRONT_CHAIN_RING = 0.075
float pedal_simulation_left_leg_experimental.BIKE_VELOCITY = 0.0
float pedal_simulation_left_leg_experimental.PEDAL_SINGLE_ROTATION_DURATION = 0.0
float pedal_simulation_left_leg_experimental.TRAJECTORY_POINT_DURATION = 0.0
list pedal_simulation_left_leg_experimental.x_pedal_record = [ ]
list pedal_simulation_left_leg_experimental.y_pedal_record = [ ]
string pedal_simulation_left_leg_experimental.ROS_JOINT_HIP_RIGHT = "joint_hip_right"
string pedal_simulation_left_leg_experimental.ROS_JOINT_KNEE_RIGHT = "joint_knee_right"
string pedal_simulation_left_leg_experimental.ROS_JOINT_ANKLE_RIGHT = "joint_foot_right"
string pedal_simulation_left_leg_experimental.ROS_JOINT_HIP_LEFT = "joint_hip_left"
string pedal_simulation_left_leg_experimental.ROS_JOINT_KNEE_LEFT = "joint_knee_left"
string pedal_simulation_left_leg_experimental.ROS_JOINT_ANKLE_LEFT = "joint_foot_left"
string pedal_simulation_left_leg_experimental.RIGHT_HIP_JOINT = "right_hip"
string pedal_simulation_left_leg_experimental.RIGHT_KNEE_JOINT = "right_knee"
string pedal_simulation_left_leg_experimental.RIGHT_ANKLE_JOINT = "right_ankle"
string pedal_simulation_left_leg_experimental.LEFT_HIP_JOINT = "left_hip"
string pedal_simulation_left_leg_experimental.LEFT_KNEE_JOINT = "left_knee"
string pedal_simulation_left_leg_experimental.LEFT_ANKLE_JOINT = "left_ankle"
pedal_simulation_left_leg_experimental.f_interpolated_hip_right = None
pedal_simulation_left_leg_experimental.f_interpolated_hip_left = None
pedal_simulation_left_leg_experimental.f_interpolated_knee_right = None
pedal_simulation_left_leg_experimental.f_interpolated_knee_left = None
pedal_simulation_left_leg_experimental.f_interpolated_ankle_right = None
pedal_simulation_left_leg_experimental.f_interpolated_ankle_left = None
pedal_simulation_left_leg_experimental.f_interpolated_pedal_angle = None
float pedal_simulation_left_leg_experimental.velocity_error_factor_hip = 1.0
float pedal_simulation_left_leg_experimental.velocity_error_factor_knee = 1.0
float pedal_simulation_left_leg_experimental.velocity_error_factor_ankle = 1.0
int pedal_simulation_left_leg_experimental.velocity_error_counter = 0
pedal_simulation_left_leg_experimental.ros_right_hip_publisher = rospy.Publisher('/jo
pedal_simulation_left_leg_experimental.ros_right_knee_publisher = rospy.Publisher('/j
pedal_simulation_left_leg_experimental.ros_right_ankle_publisher = rospy.Publisher('/
pedal_simulation_left_leg_experimental.ros_left_hip_publisher = rospy.Publisher('/joi
pedal_simulation_left_leg_experimental.ros_left_knee_publisher = rospy.Publisher('/jo
pedal_simulation_left_leg_experimental.ros_left_ankle_publisher = rospy.Publisher('/j

```

```

float pedal_simulation_left_leg_experimental.PEDAL_CENTER_OFFSET_X = 0.00027829138673
float pedal_simulation_left_leg_experimental.PEDAL_CENTER_OFFSET_Y = -0.0351316495558
float pedal_simulation_left_leg_experimental.PEDAL_CENTER_OFFSET_Z = 0.00100933134810
float pedal_simulation_left_leg_experimental.OLD_X_OFFSET = 0.7163902600571725 + 0.23
float pedal_simulation_left_leg_experimental.OLD_Y_OFFSET = -0.010388552466272516 + 0
float pedal_simulation_left_leg_experimental.OLD_Z_OFFSET = 0.2164376942146126 - 0.20
float pedal_simulation_left_leg_experimental.NEW_X_OFFSET = -0.23003546880974085 + (-
float pedal_simulation_left_leg_experimental.NEW_Y_OFFSET = -0.010388308215364166 + (
float pedal_simulation_left_leg_experimental.NEW_Z_OFFSET = -0.2052706960113988 + (0.
float pedal_simulation_left_leg_experimental.DIFF_X_OFFSET = NEW_X_OFFSET - OLD_X_OFF
float pedal_simulation_left_leg_experimental.DIFF_Y_OFFSET = NEW_Y_OFFSET - OLD_Y_OFF
float pedal_simulation_left_leg_experimental.DIFF_Z_OFFSET = NEW_Z_OFFSET - OLD_Z_OFF
list pedal_simulation_left_leg_experimental._jointsList= [ LEFT_HIP_JOINT, LEFT_KNEE
dictionary pedal_simulation_left_leg_experimental._parametersRightHip= { "param_p"
dictionary pedal_simulation_left_leg_experimental._parametersRightKnee= { "param_p
dictionary pedal_simulation_left_leg_experimental._parametersRightAnkle= { "param
dictionary pedal_simulation_left_leg_experimental._parametersLeftHip= { "param_p":
dictionary pedal_simulation_left_leg_experimental._parametersLeftKnee= { "param_p"
dictionary pedal_simulation_left_leg_experimental._parametersLeftAnkle= { "param_p
dictionary pedal_simulation_left_leg_experimental._jointsControlData= { RIGHT_HIP
dictionary pedal_simulation_left_leg_experimental.joint_status_data= { RIGHT_HIP_J
int pedal_simulation_left_leg_experimental.number_imported_trajectory_points = -1
int pedal_simulation_left_leg_experimental.trajectoryStartingPoint = 0
list pedal_simulation_left_leg_experimental.pedalTrajectoryRight = [ ]
list pedal_simulation_left_leg_experimental.pedalAngleTrajectoryLeft = [ ]
list pedal_simulation_left_leg_experimental.pedalTrajectoryLeft = [ ]
list pedal_simulation_left_leg_experimental.hipTrajectoryRight = [ ]
list pedal_simulation_left_leg_experimental.kneeTrajectoryRight = [ ]
list pedal_simulation_left_leg_experimental.ankleTrajectoryRight = [ ]
list pedal_simulation_left_leg_experimental.hipTrajectoryLeft = [ ]
list pedal_simulation_left_leg_experimental.kneeTrajectoryLeft = [ ]
list pedal_simulation_left_leg_experimental.ankleTrajectoryLeft = [ ]
namespace pedal_simulation_left_leg_fk_pos

```

Functions

joint_state_callback (*joint_data joint_data*)

Documentation for a function.

This function collects the current status of the joint-angles and saves them in the global dictionary “joint_status_data”.

import_joint_trajectory_record ()

Documentation for a function.

Collects and saves all joint- and pedal-angles from the pre-captured trajectory from the (global variable).

get_joint_position (*jointName jointName*)

Documentation for a function.

Return current joint-angle of .

get_joint_velocity (*jointName jointName*)

Documentation for a function.

Return current joint-velocity of .

get_position (*endeffector endeffector, frame frame*)

Documentation for a function.

Return the position of the of the correspondent .

set_joint_controller_parameters (*proportionalVal proportionalVal, derivativeVal derivativeVal*)

Documentation for a function.

Sets Kp of joint-controller to Sets Kd of joint-controller to

get_position_left_foot ()

Documentation for a function.

Return the position of the left foot of Roboy.

get_position_left_foot_with_coord_trans ()

get_position_right_foot ()

Documentation for a function.

Return the position of the right foot of Roboy.

get_distance (*point1 point1, point2 point2*)

Documentation for a function.

Return the distance between two points where points are a list of two coordinates.

check_output_limits (*inputVal inputVal*)

Documentation for a function.

Checks if is inside possible range of joint-angle velocities. If not, returns max-velocity if too high or min-velocity if too low instead of .

compute_velocity (*joint_name joint_name, next_joint_angle next_joint_angle, current_joint_angle current_joint_angle, end_time end_time*)

Documentation for a function.

Returns ideal joint-velocity for according to and joint-angle-difference between and :

$ideal_velocity = joint_angle_difference / (end_time - current_time)$

get_joint_angle (*joint_name joint_name, pedal_angle pedal_angle*)

Documentation for a function.

Evaluate and return joint-angle of correspondent to using the interpolated function:

The functions can be used by calling “<function_name>(<pedal_angle>)” ==> returns <joint_angle>

interpolate_functions ()

Documentation for a function.

Initializing interpolated functions for joint-angles using pre-captured set points with pedal-angle as input and joint-angle as output:

The functions can be used by calling “<function_name>(<pedal_angle>)” ==> returns <joint_angle>

evaluate_current_pedal_angle (*current_point current_point*)

Documentation for a function.

Evaluating the current pedal-angle according to the current position of the left foot . Using trigonometric functions for the evaluation of the angle.

publish_velocity (*joint_name joint_name, next_joint_angle next_joint_angle, current_joint_angle current_joint_angle, end_time end_time*)

Documentation for a function.

For joint :

- Evaluate ideal velocity in rad/s according to joint-angle-difference and end-time of transition
- Multiply value with (global variable) to receive velocity in rad/s
- Multiply value with (global variable) to slow down simulation time.
- Multiply value with (global variable) of correspondent joint to erase joint-error
- Publish velocity to joint-controller and sleep until end-time of transition

update_velocity (*velocity_Twist velocity_Twist*)

Documentation for a function.

Updates the global variables , and TRAJECTORY_POINT_DURATION when a bike-velocity gets published to the topic “cmd_vel”.

get_angle_difference (*angle_1 angle_1, angle_2 angle_2*)

Documentation for a function.

Returns the absolute difference of two angles within the interval $[0; 2\pi]$

control_pedaling ()

Documentation for a function.

Controls the whole pedaling-process.

Evaluates next pedal-angle according to current pedal-angle and (global variable). Uses to compute joint-velocities for every joint-angle for every transition between two. trajectory points.

Use Threads to simultaneously publish and control joint-angles.

Computes joint-angle-error and adjusts error-factors for every joint to optimize ideal joint-velocity for further transitions.

Simplified Pseudo-Code:

while bike_velocity = 0:

```
for joint in joints:
```

```
    publish(0)
```

```
sleep()
```

```
current_pedal_angle = get_current_pedal_angle(left_foot_position)
```

```
next_pedal_angle = current_pedal_angle + (2pi / number_circulation_points)
```

```
for joint in joints:
```

```
    next_joint_angle = interpolation_function(next_pedal_angle)
```

```
    Thread.publish_velocity(joint_name, current_joint_angle, next_joint_angle,   
    ↪end_time)
```

```
for Thread in created_threads:
```

```
    Thread.join
```

```
for joint in joints:
```

```
    update_error_factor()
```

```
main()
```

```
    Documentation for a function.
```

```
    Initializes the Control-Node for Pedaling and starts Pedaling-Algorithm.
```

Variables

```
bool pedal_simulation_left_leg_fk_pos.PRINT_DEBUG = True
```

```
float pedal_simulation_left_leg_fk_pos.SIMULATION_FACTOR = 100.0
```

```
int pedal_simulation_left_leg_fk_pos.NUMBER_CIRCULATION_POINTS = 30
```

```
string pedal_simulation_left_leg_fk_pos.RECORDED_TRAJECTORY_FILENAME = "trajectory_pe
```

```
float pedal_simulation_left_leg_fk_pos.JOINT_VELOCITY_FACTOR_SIMULATION = 0.008
```

```
float pedal_simulation_left_leg_fk_pos.PEDAL_POSITION_ERROR_TOLERANCE = 0.02
```

```
float pedal_simulation_left_leg_fk_pos.JOINT_TRAJECTORY_ERROR_TOLERANCE = 0.02
```

```
int pedal_simulation_left_leg_fk_pos.CONTROLLER_FREQUENCY = 100
```

```
float pedal_simulation_left_leg_fk_pos.MIN_JOINT_VEL = -10.0
```

```
float pedal_simulation_left_leg_fk_pos.MAX_JOINT_VEL = 10.0
```

```
float pedal_simulation_left_leg_fk_pos.RADIUS_BACK_TIRE = 0.294398
```

```
float pedal_simulation_left_leg_fk_pos.RADIUS_GEAR_CLUSTER = 0.06
```

```
float pedal_simulation_left_leg_fk_pos.RADIUS_FRONT_CHAIN_RING = 0.075
```

```
float pedal_simulation_left_leg_fk_pos.OLD_X_OFFSET = 0.7163902600571725 + 0.23003546
```

```
float pedal_simulation_left_leg_fk_pos.OLD_Y_OFFSET = -0.010388552466272516 + 0.01038
```

```
float pedal_simulation_left_leg_fk_pos.OLD_Z_OFFSET = 0.2164376942146126 - 0.20527069
```

```

float pedal_simulation_left_leg_fk_pos.NEW_X_OFFSET = -0.23003546880974085 + (-0.7163
float pedal_simulation_left_leg_fk_pos.NEW_Y_OFFSET = -0.010388308215364166 + (0.0103
float pedal_simulation_left_leg_fk_pos.NEW_Z_OFFSET = -0.2052706960113988 + (0.216437
float pedal_simulation_left_leg_fk_pos.DIFF_X_OFFSET = NEW_X_OFFSET - OLD_X_OFFSET
float pedal_simulation_left_leg_fk_pos.DIFF_Y_OFFSET = NEW_Y_OFFSET - OLD_Y_OFFSET
float pedal_simulation_left_leg_fk_pos.DIFF_Z_OFFSET = NEW_Z_OFFSET - OLD_Z_OFFSET
float pedal_simulation_left_leg_fk_pos.BIKE_VELOCITY = 0.0
float pedal_simulation_left_leg_fk_pos.PEDAL_SINGLE_ROTATION_DURATION = 20.0
float pedal_simulation_left_leg_fk_pos.TRAJECTORY_POINT_DURATION = 0.0
list pedal_simulation_left_leg_fk_pos.x_pedal_record = [ ]
list pedal_simulation_left_leg_fk_pos.y_pedal_record = [ ]
string pedal_simulation_left_leg_fk_pos.ROS_JOINT_HIP_RIGHT = "joint_hip_right"
string pedal_simulation_left_leg_fk_pos.ROS_JOINT_KNEE_RIGHT = "joint_knee_right"
string pedal_simulation_left_leg_fk_pos.ROS_JOINT_ANKLE_RIGHT = "joint_foot_right"
string pedal_simulation_left_leg_fk_pos.ROS_JOINT_HIP_LEFT = "joint_hip_left"
string pedal_simulation_left_leg_fk_pos.ROS_JOINT_KNEE_LEFT = "joint_knee_left"
string pedal_simulation_left_leg_fk_pos.ROS_JOINT_ANKLE_LEFT = "joint_foot_left"
string pedal_simulation_left_leg_fk_pos.RIGHT_HIP_JOINT = "right_hip"
string pedal_simulation_left_leg_fk_pos.RIGHT_KNEE_JOINT = "right_knee"
string pedal_simulation_left_leg_fk_pos.RIGHT_ANKLE_JOINT = "right_ankle"
string pedal_simulation_left_leg_fk_pos.LEFT_HIP_JOINT = "left_hip"
string pedal_simulation_left_leg_fk_pos.LEFT_KNEE_JOINT = "left_knee"
string pedal_simulation_left_leg_fk_pos.LEFT_ANKLE_JOINT = "left_ankle"
pedal_simulation_left_leg_fk_pos.f_interpolated_hip_right = None
pedal_simulation_left_leg_fk_pos.f_interpolated_hip_left = None
pedal_simulation_left_leg_fk_pos.f_interpolated_knee_right = None
pedal_simulation_left_leg_fk_pos.f_interpolated_knee_left = None
pedal_simulation_left_leg_fk_pos.f_interpolated_ankle_right = None
pedal_simulation_left_leg_fk_pos.f_interpolated_ankle_left = None
pedal_simulation_left_leg_fk_pos.f_interpolated_pedal_angle = None
float pedal_simulation_left_leg_fk_pos.velocity_error_factor_hip = 1.0
float pedal_simulation_left_leg_fk_pos.velocity_error_factor_knee = 1.0
float pedal_simulation_left_leg_fk_pos.velocity_error_factor_ankle = 1.0
int pedal_simulation_left_leg_fk_pos.velocity_error_counter = 0
pedal_simulation_left_leg_fk_pos.ros_right_hip_publisher = rospy.Publisher('/joint_hi
pedal_simulation_left_leg_fk_pos.ros_right_knee_publisher = rospy.Publisher('/joint_k

```

```

pedal_simulation_left_leg_fk_pos.ros_right_ankle_publisher = rospy.Publisher('/joint_
pedal_simulation_left_leg_fk_pos.ros_left_hip_publisher = rospy.Publisher('/joint_hip
pedal_simulation_left_leg_fk_pos.ros_left_knee_publisher = rospy.Publisher('/joint_kn
pedal_simulation_left_leg_fk_pos.ros_left_ankle_publisher = rospy.Publisher('/joint_f
float pedal_simulation_left_leg_fk_pos.PEDAL_CENTER_OFFSET_X = 0.0002782913867391912
float pedal_simulation_left_leg_fk_pos.PEDAL_CENTER_OFFSET_Y = -0.035131649555820536
float pedal_simulation_left_leg_fk_pos.PEDAL_CENTER_OFFSET_Z = 0.0010093313481022886
list pedal_simulation_left_leg_fk_pos._jointsList= [ LEFT_HIP_JOINT, LEFT_KNEE_JOINT,
dictionary pedal_simulation_left_leg_fk_pos._parametersRightHip= { "param_p": 1500
dictionary pedal_simulation_left_leg_fk_pos._parametersRightKnee= { "param_p": 200
dictionary pedal_simulation_left_leg_fk_pos._parametersRightAnkle= { "param_p": 10
dictionary pedal_simulation_left_leg_fk_pos._parametersLeftHip= { "param_p": 1500.
dictionary pedal_simulation_left_leg_fk_pos._parametersLeftKnee= { "param_p": 2000
dictionary pedal_simulation_left_leg_fk_pos._parametersLeftAnkle= { "param_p": 100
dictionary pedal_simulation_left_leg_fk_pos._jointsControlData= { RIGHT_HIP_JOINT:
dictionary pedal_simulation_left_leg_fk_pos.joint_status_data= { RIGHT_HIP_JOINT:
int pedal_simulation_left_leg_fk_pos.number_imported_trajectory_points = -1
int pedal_simulation_left_leg_fk_pos.trajectoryStartingPoint = 0
list pedal_simulation_left_leg_fk_pos.pedalTrajectoryRight = [ ]
list pedal_simulation_left_leg_fk_pos.pedalAngleTrajectoryLeft = []
list pedal_simulation_left_leg_fk_pos.pedalTrajectoryLeft = [ ]
list pedal_simulation_left_leg_fk_pos.hipTrajectoryRight = [ ]
list pedal_simulation_left_leg_fk_pos.kneeTrajectoryRight = [ ]
list pedal_simulation_left_leg_fk_pos.ankleTrajectoryRight = [ ]
list pedal_simulation_left_leg_fk_pos.hipTrajectoryLeft = [ ]
list pedal_simulation_left_leg_fk_pos.kneeTrajectoryLeft = [ ]
list pedal_simulation_left_leg_fk_pos.ankleTrajectoryLeft = [ ]

```

namespace pedaling

Documentation for this module.

Control of Roboys' hip, knees and feet for pedaling. In order to reach the requested bike-velocity, we compute correspondent angular-velocity for the pedals. The circulation is ideally divided into trajectory points represented by angles within the circulation circle (pedal-angle). With angle-difference between two points and angular-velocity we use the correspondent transition time to control the joints in order to create a pedaling motion for Roboy.

For every joint, we use cubic spline interpolation of pre-captured set points with pedal_angle as input to receive continuous function to be able to get the corresponding joint-angle for every pedal-angle. This enables to use joint-angle-difference between two trajectory-points and transition time to compute and apply velocity-control to the joint-angles to create pedaling-motion

In order to decrease and hopefully erase velocity-error, we multiply the computed (ideal) velocity with an error-factor which is computed and dependent on the velocity-error of previous transitions and measured by pedal-angle-error

Test-script for testing the accuracy of a requested velocity and the transition-time (acceleration) between two different velocities, which is only important for testing in reality, because there is no transition-time between different velocities in simulation.

and determine the time-steps in seconds for the measurement of velocity and determine the size of a step between two different velocities in m/s

```
namespace qpOASES
```

```
namespace robaython
```

Functions

```
compute_leg_length(hip_angle hip_angle, knee_angle knee_angle, ankle_angle ankle_angle)
```

```
get_pedal_position_x(pedal_angle pedal_angle)
```

```
get_pedal_position_y(pedal_angle pedal_angle)
```

```
compute_distance(x1 x1, y1 y1, x2 x2, y2 y2)
```

Variables

```
int robaython.corner_angle = math.pi/2
```

```
float robaython.thigh_along = 309.75
```

```
float robaython.thigh_across = 7.46
```

```
float robaython.leg_along = 347.50
```

```
float robaython.leg_across = 7.0
```

```
float robaython.foot_along = 110.0
```

```
float robaython.foot_across = 47.0
```

```
float robaython.pedal_offset = 11.88
```

```
float robaython.pedal_radius = 169.24
```

```
robaython.maximum_length = compute_leg_length(0, 0, -(math.pi/4))
```

```
robaython.minimum_length = compute_leg_length(math.pi/2, 2*math.pi/3, (math.pi/4))
```

```
robaython.length_difference = maximum_length - minimum_length
```

```
list robaython.candidate_x_list = []
```

```
list robaython.candidate_y_list = []
```

```
list robaython.confirmed_hip_positions = [[]]
```

```
list robaython.confirmed_x_positions = []
```

```
list robaython.confirmed_y_positions = []
```

```
int robaython.N_ANGLES_CHECKED = 36
```

```
int robaython.discarded = 0
```

```

    roboython.pedal_x = get_pedal_position_x(pedal_angle_iter*math.pi*2/N_ANGLES_CHECKED)
    roboython.pedal_y = get_pedal_position_y(pedal_angle_iter * math.pi * 2 / N_ANGLES_CHECKED)
    roboython.this_distance = compute_distance(pedal_x, pedal_y, candidate_x, candidate_y)
namespace state_machine

```

Functions

```

import JointTrajectoryRecord()
    UTILITY FUNCTIONS ###.

import JointPIDParameters()

setJointVelocity(jointName jointName, jointVel jointVel)
getJointVelocity(jointName jointName)
getJointPosition(jointName jointName)
getPedalPosition()
getDistance(point1 point1, point2 point2)
setPedalSingleRotationDuration(new_duration_seconds new_duration_seconds)
setTrajectoryPointDuration()
interpolateTrajectoryPoints(jointName jointName, value1 value1, value2 value2, startTime
                             startTime, currTime currTime, endTime endTime)

FSM()

computeVelocitySetpoint(jointName jointName, currPos currPos, currTime currTime, goalPos
                        goalPos)
    CONTROL FUNCTIONS ###.

main()
    MAIN ###.

```

Variables

```

string state_machine.CONTROL_PARAMETERS_FILENAME = "control_parameters_dds.json"
    MODULE PARAMETERS ###.

string state_machine.RECORDED_TRAJECTORY_FILENAME = "recorded_trajectory_dds.json"

int state_machine.PEDAL_POSITION_ERROR_TOLERANCE = 10
int state_machine.PEDAL_SINGLE_ROTATION_DURATION = 10
int state_machine.TRAJECTORY_POINT_DURATION = 0
int state_machine.CONTROLLER_FREQUENCY = 20
float state_machine.TESTING_VELOCITY1 = 0.0
    GLOBAL VARIABLES ###.

string state_machine.RIGHT_HIP_JOINT = "right_hip"
string state_machine.RIGHT_KNEE_JOINT = "right_knee"
string state_machine.RIGHT_ANKLE_JOINT = "right_ankle"

```


Functions

```

setJointControllerParameters (proportionalVal proportionalVal, derivativeVal derivativeVal)
    UTILITY FUNCTIONS ###.

computeSteeringAngles ()

computeHandTrajectories ()

getPositionLeftHand ()

getPositionRightHand ()

jointStateCallback (joint_data joint_data)

inverse_kinematics_client (endeffector endeffector, frame frame, x x, y y, z z, roll roll, pitch
    pitch, yaw yaw)

main ()
    MAIN ###.

```

Variables

```

int steering_capture_multiple_trajectories.MAX_TURNING_ANGLE = math.pi / 15
    FUNCTION PARAMETERS ###.

int steering_capture_multiple_trajectories.NUM_STEERING_ANGLES = 61

int steering_capture_multiple_trajectories.POINT_MULTIPLICITY = 50

string steering_capture_multiple_trajectories.JSON_FILENAME = "multiple_steering_traj

float steering_capture_multiple_trajectories.JOINT_ANGLE_TOLERANCE_FK = 0.01

string steering_capture_multiple_trajectories.ENDEFFECTOR_RIGHT = "right_hand"

string steering_capture_multiple_trajectories.FRAME_RIGHT = "right_hand"

string steering_capture_multiple_trajectories.ENDEFFECTOR_LEFT = "left_hand"

string steering_capture_multiple_trajectories.FRAME_LEFT = "left_hand"

int steering_capture_multiple_trajectories.BIKE_OFFSET_X = 0
    MEASURED PARAMETERS ###.

int steering_capture_multiple_trajectories.BIKE_OFFSET_Y = 0

int steering_capture_multiple_trajectories.BIKE_OFFSET_Z = 0

float steering_capture_multiple_trajectories.RIKSHAW_TURN_JOINT_X_OFFSET = 0.71639026

float steering_capture_multiple_trajectories.RIKSHAW_TURN_JOINT_Y_OFFSET = -0.0103885

float steering_capture_multiple_trajectories.RIKSHAW_TURN_JOINT_Z_OFFSET = 0.21643769

int steering_capture_multiple_trajectories.YAW_RIGHT_HAND_OFFSET = math.pi / 2 + math

int steering_capture_multiple_trajectories.YAW_LEFT_HAND_OFFSET = 3 * math.pi / 2 + m

float steering_capture_multiple_trajectories.HANDLEBAR_X_OFFSET = 0.728713

float steering_capture_multiple_trajectories.HANDLEBAR_Z_OFFSET = 0.719269

float steering_capture_multiple_trajectories.HAND_Y_OFFSET = 0.2

list steering_capture_multiple_trajectories._steeringAngles = []
    GLOBAL VARIABLES ###.

```

```

list steering_capture_multiple_trajectories._rightHandTrajectory = []
list steering_capture_multiple_trajectories._leftHandTrajectory = []
list steering_capture_multiple_trajectories._centerHandlebarTrajectory = []
string steering_capture_multiple_trajectories.JOINT_SHOULDER_AXIS0_RIGHT = "right_sho
string steering_capture_multiple_trajectories.JOINT_SHOULDER_AXIS1_RIGHT = "right_sho
string steering_capture_multiple_trajectories.JOINT_SHOULDER_AXIS2_RIGHT = "right_sho
string steering_capture_multiple_trajectories.JOINT_SHOULDER_AXIS0_LEFT = "left_shoul
string steering_capture_multiple_trajectories.JOINT_SHOULDER_AXIS1_LEFT = "left_shoul
string steering_capture_multiple_trajectories.JOINT_SHOULDER_AXIS2_LEFT = "left_shoul
string steering_capture_multiple_trajectories.JOINT_ELBOW_ROT0_RIGHT = "elbow_right_r
string steering_capture_multiple_trajectories.JOINT_ELBOW_ROT1_RIGHT = "elbow_right_r
string steering_capture_multiple_trajectories.JOINT_ELBOW_ROT0_LEFT = "elbow_left_rot
string steering_capture_multiple_trajectories.JOINT_ELBOW_ROT1_LEFT = "elbow_left_rot
string steering_capture_multiple_trajectories.JOINT_WRIST_0_RIGHT = "right_wrist_0"
string steering_capture_multiple_trajectories.JOINT_WRIST_1_RIGHT = "right_wrist_1"
string steering_capture_multiple_trajectories.JOINT_WRIST_0_LEFT = "left_wrist_0"
string steering_capture_multiple_trajectories.JOINT_WRIST_1_LEFT = "left_wrist_1"
list steering_capture_multiple_trajectories.JOINT_LIST= [JOINT_SHOULDER_AXIS0_RIGHT,
dictionary steering_capture_multiple_trajectories._jointsStatusData

namespace steering_capture_trajectory

```

Functions

```

computeSteeringAngles()
    UTILITY FUNCTIONS ###.

computeHandTrajectories()

getPositionLeftHand()

getPositionRightHand()

setJointControllerParameters (proportionalVal proportionalVal, derivativeVal derivativeVal)

jointStateCallback (joint_data joint_data)

inverse_kinematics_client (endeffector endeffector, frame frame, x x, y y, z z, roll roll, pitch
    pitch, yaw yaw)

main()
    MAIN ###.

```

Variables

```

int steering_capture_trajectory.MAX_TURNING_ANGLE = math.pi / 15
    FUNCTION PARAMETERS ###.

int steering_capture_trajectory.NUM_STEERING_ANGLES = 61

float steering_capture_trajectory.RIKSHAW_TURN_JOINT_X_OFFSET = 0.7163902600571725 + 0
float steering_capture_trajectory.RIKSHAW_TURN_JOINT_Y_OFFSET = -0.010388552466272516
float steering_capture_trajectory.RIKSHAW_TURN_JOINT_Z_OFFSET = 0.2164376942146126 - 0
int steering_capture_trajectory.YAW_RIGHT_HAND_OFFSET = math.pi / 2 + math.pi
int steering_capture_trajectory.YAW_LEFT_HAND_OFFSET = 3 * math.pi / 2 + math.pi
float steering_capture_trajectory.HANDLEBAR_X_OFFSET = 0.728713
float steering_capture_trajectory.HANDLEBAR_Z_OFFSET = 0.719269
float steering_capture_trajectory.HAND_Y_OFFSET = 0.2
string steering_capture_trajectory.JSON_FILENAME = "steering_trajectory.json"
float steering_capture_trajectory.JOINT_ANGLE_TOLERANCE_FK = 0.01
string steering_capture_trajectory.ENDEFFECTOR_RIGHT = "right_hand"
string steering_capture_trajectory.FRAME_RIGHT = "right_hand"
string steering_capture_trajectory.ENDEFFECTOR_LEFT = "left_hand"
string steering_capture_trajectory.FRAME_LEFT = "left_hand"
int steering_capture_trajectory.BIKE_OFFSET_X = 0
    MEASURED PARAMETERS ###.

int steering_capture_trajectory.BIKE_OFFSET_Y = 0
int steering_capture_trajectory.BIKE_OFFSET_Z = 0
list steering_capture_trajectory._steeringAngles = []
    GLOBAL VARIABLES ###.

list steering_capture_trajectory._rightHandTrajectory = []
list steering_capture_trajectory._leftHandTrajectory = []
list steering_capture_trajectory._centerHandlebarTrajectory = []
string steering_capture_trajectory.JOINT_SHOULDER_AXIS0_RIGHT = "right_shoulder_axis0"
string steering_capture_trajectory.JOINT_SHOULDER_AXIS1_RIGHT = "right_shoulder_axis1"
string steering_capture_trajectory.JOINT_SHOULDER_AXIS2_RIGHT = "right_shoulder_axis2"
string steering_capture_trajectory.JOINT_SHOULDER_AXIS0_LEFT = "left_shoulder_axis0"
string steering_capture_trajectory.JOINT_SHOULDER_AXIS1_LEFT = "left_shoulder_axis1"
string steering_capture_trajectory.JOINT_SHOULDER_AXIS2_LEFT = "left_shoulder_axis2"
string steering_capture_trajectory.JOINT_ELBOW_ROT0_RIGHT = "elbow_right_rot0"
string steering_capture_trajectory.JOINT_ELBOW_ROT1_RIGHT = "elbow_right_rot1"
string steering_capture_trajectory.JOINT_ELBOW_ROT0_LEFT = "elbow_left_rot0"
string steering_capture_trajectory.JOINT_ELBOW_ROT1_LEFT = "elbow_left_rot1"

```



```

string steering_capture_trajectory.JOINT_WRIST_0_RIGHT = "right_wrist_0"
string steering_capture_trajectory.JOINT_WRIST_1_RIGHT = "right_wrist_1"
string steering_capture_trajectory.JOINT_WRIST_0_LEFT = "left_wrist_0"
string steering_capture_trajectory.JOINT_WRIST_1_LEFT = "left_wrist_1"
list steering_capture_trajectory.JOINT_LIST= [JOINT_SHOULDER_AXIS0_RIGHT, JOINT_SHOULDER_AXIS1_RIGHT, JOINT_SHOULDER_AXIS2_RIGHT, JOINT_SHOULDER_AXIS0_LEFT, JOINT_SHOULDER_AXIS1_LEFT, JOINT_SHOULDER_AXIS2_LEFT, JOINT_ELBO
dictionary steering_capture_trajectory._jointsStatusData
namespace steering_interpolate_and_print

```

Functions

```

import JointTrajectoryRecord()
    UTILITY FUNCTIONS ###.

interpolateAllJointPositions()

printInterpolatedFunctions()

main()
    MAIN ###.

```

Variables

```

string steering_interpolate_and_print.RECORDED_TRAJECTORY_FILENAME = "steering_trajectory_recorded_filename"
    MODULE PARAMETERS ###.

bool steering_interpolate_and_print.PRINT_DEBUG = True

string steering_interpolate_and_print.JOINT_SHOULDER_AXIS0_RIGHT = "right_shoulder_axis0"
    GLOBAL VARIABLES ###.

string steering_interpolate_and_print.JOINT_SHOULDER_AXIS1_RIGHT = "right_shoulder_axis1"
string steering_interpolate_and_print.JOINT_SHOULDER_AXIS2_RIGHT = "right_shoulder_axis2"
string steering_interpolate_and_print.JOINT_SHOULDER_AXIS0_LEFT = "left_shoulder_axis0"
string steering_interpolate_and_print.JOINT_SHOULDER_AXIS1_LEFT = "left_shoulder_axis1"
string steering_interpolate_and_print.JOINT_SHOULDER_AXIS2_LEFT = "left_shoulder_axis2"
string steering_interpolate_and_print.JOINT_ELBO_ROT0_RIGHT = "elbow_right_rot0"
string steering_interpolate_and_print.JOINT_ELBO_ROT1_RIGHT = "elbow_right_rot1"
string steering_interpolate_and_print.JOINT_ELBO_ROT0_LEFT = "elbow_left_rot0"
string steering_interpolate_and_print.JOINT_ELBO_ROT1_LEFT = "elbow_left_rot1"
string steering_interpolate_and_print.JOINT_WRIST_0_RIGHT = "right_wrist_0"
string steering_interpolate_and_print.JOINT_WRIST_1_RIGHT = "right_wrist_1"
string steering_interpolate_and_print.JOINT_WRIST_0_LEFT = "left_wrist_0"
string steering_interpolate_and_print.JOINT_WRIST_1_LEFT = "left_wrist_1"
int steering_interpolate_and_print._numTrajectoryPoints = 0
list steering_interpolate_and_print._trajectorySteering = []

```

```
list steering_interpolate_and_print._trajectoryShoulder0Right = []
list steering_interpolate_and_print._trajectoryShoulder1Right = []
list steering_interpolate_and_print._trajectoryShoulder2Right = []
list steering_interpolate_and_print._trajectoryShoulder0Left = []
list steering_interpolate_and_print._trajectoryShoulder1Left = []
list steering_interpolate_and_print._trajectoryShoulder2Left = []
list steering_interpolate_and_print._trajectoryElbow0Right = []
list steering_interpolate_and_print._trajectoryElbow1Right = []
list steering_interpolate_and_print._trajectoryElbow0Left = []
list steering_interpolate_and_print._trajectoryElbow1Left = []
list steering_interpolate_and_print._trajectoryWrist0Right = []
list steering_interpolate_and_print._trajectoryWrist1Right = []
list steering_interpolate_and_print._trajectoryWrist0Left = []
list steering_interpolate_and_print._trajectoryWrist1Left = []
steering_interpolate_and_print._interpolatedShoulder0Right = None
steering_interpolate_and_print._interpolatedShoulder1Right = None
steering_interpolate_and_print._interpolatedShoulder2Right = None
steering_interpolate_and_print._interpolatedShoulder0Left = None
steering_interpolate_and_print._interpolatedShoulder1Left = None
steering_interpolate_and_print._interpolatedShoulder2Left = None
steering_interpolate_and_print._interpolatedElbow0Right = None
steering_interpolate_and_print._interpolatedElbow1Right = None
steering_interpolate_and_print._interpolatedElbow0Left = None
steering_interpolate_and_print._interpolatedElbow1Left = None
steering_interpolate_and_print._interpolatedWrist0Right = None
steering_interpolate_and_print._interpolatedWrist1Right = None
steering_interpolate_and_print._interpolatedWrist0Left = None
steering_interpolate_and_print._interpolatedWrist1Left = None
namespace steering_interpolate_multiple_trajectories_and_print
```

Functions

```
importJointTrajectoryRecord()
    UTILITY FUNCTIONS ###.
regressAllJointPositions(order order)
printRegressedFunctions()
printAllTrajectories()
```

```
saveRegressionToFile (filename filename, order order)
```

```
main ()
    MAIN ###.
```

Variables

```
string steering_interpolate_multiple_trajectories_and_print.RECORDED_TRAJECTORY_FILENAME
    MODULE PARAMETERS ###.
```

```
bool steering_interpolate_multiple_trajectories_and_print.PRINT_DEBUG = True
```

```
int steering_interpolate_multiple_trajectories_and_print.MAX_TURNING_ANGLE = math.pi
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_SHOULDER_AXIS0_RIGHT
    GLOBAL VARIABLES ###.
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_SHOULDER_AXIS1_RIGHT
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_SHOULDER_AXIS2_RIGHT
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_SHOULDER_AXIS0_LEFT
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_SHOULDER_AXIS1_LEFT
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_SHOULDER_AXIS2_LEFT
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_ELBOW_ROT0_RIGHT =
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_ELBOW_ROT1_RIGHT =
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_ELBOW_ROT0_LEFT = "
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_ELBOW_ROT1_LEFT = "
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_WRIST_0_RIGHT = "ri
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_WRIST_1_RIGHT = "ri
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_WRIST_0_LEFT = "lef
```

```
string steering_interpolate_multiple_trajectories_and_print.JOINT_WRIST_1_LEFT = "lef
```

```
int steering_interpolate_multiple_trajectories_and_print._numTrajectoryPoints = 0
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectorySteering = []
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryShoulder0Right =
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryShoulder1Right =
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryShoulder2Right =
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryShoulder0Left =
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryShoulder1Left =
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryShoulder2Left =
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryElbow0Right = []
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryElbow1Right = []
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryElbow0Left = []
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryElbow1Left = []
```

```
list steering_interpolate_multiple_trajectories_and_print._trajectoryWrist0Right = []
```

```

list steering_interpolate_multiple_trajectories_and_print._trajectoryWrist1Right = []
list steering_interpolate_multiple_trajectories_and_print._trajectoryWrist0Left = []
list steering_interpolate_multiple_trajectories_and_print._trajectoryWrist1Left = []
steering_interpolate_multiple_trajectories_and_print._regressedShoulder0Right = None
steering_interpolate_multiple_trajectories_and_print._regressedShoulder1Right = None
steering_interpolate_multiple_trajectories_and_print._regressedShoulder2Right = None
steering_interpolate_multiple_trajectories_and_print._regressedShoulder0Left = None
steering_interpolate_multiple_trajectories_and_print._regressedShoulder1Left = None
steering_interpolate_multiple_trajectories_and_print._regressedShoulder2Left = None
steering_interpolate_multiple_trajectories_and_print._regressedElbow0Right = None
steering_interpolate_multiple_trajectories_and_print._regressedElbow1Right = None
steering_interpolate_multiple_trajectories_and_print._regressedElbow0Left = None
steering_interpolate_multiple_trajectories_and_print._regressedElbow1Left = None
steering_interpolate_multiple_trajectories_and_print._regressedWrist0Right = None
steering_interpolate_multiple_trajectories_and_print._regressedWrist1Right = None
steering_interpolate_multiple_trajectories_and_print._regressedWrist0Left = None
steering_interpolate_multiple_trajectories_and_print._regressedWrist1Left = None
namespace steering_response_test

```

Functions

import_joint_trajectory_record()

Documentation for a function.

Collects and saves all joint- and steering-angles from the pre-captured trajectory from the (global variable).

regress_joint_positions_from_file (*filename filename*)

Documentation for a function.

Initializes the interpolation-functions for every joint-angle using regression. The input value of the function is a steering angle and the output value of the function is the correspondent joint angle.

The functions can be used by calling “<function_name>(<steering_angle>)” ==> returns <joint_angle>

joint_state_callback (*joint_data joint_data*)

Documentation for a function.

This function collects the current status of the joint-angles and saves them in the global dictionary “_jointStatusData”.

check_joint_angle (*joint_name joint_name, steering_angle steering_angle*)

Documentation for a function.

Checks if joint has reached the joint-angle within error-tolerance corresponding to the steering angle and returns if so or if has been reached.

steering_angle_reached (*steering_angle steering_angle*)

Documentation for a function.

Checks if a steering-angle has been reached with checking if all joint-angles have reached the correspondent joint-angles

steering_test (*pub pub*)

Documentation for a function.

Program for testing the transition-time between two different steering-angles. Saves measurements in lists and displays them using pyplot, or just the average values for all steps in

main ()

Documentation for a function.

Initializes the Test-Node for the steering-test

Variables

```
bool steering_response_test.PRINT_DEBUG = False
list steering_response_test.STEPS_DISTRIBUTION_TEST = [2, 4, 6, 8]
bool steering_response_test.SHOW_AVERAGE = True
float steering_response_test.UPDATE_FREQUENCY = 0.01
int steering_response_test.ERROR_TOLERANCE = np.pi/36
int steering_response_test.INITIAL_STARTING_TIME = 10
int steering_response_test.MAX_TRANSITION_TIME = 5
string steering_response_test.JOINT_SHOULDER_AXIS0_RIGHT = "right_shoulder_axis0"
string steering_response_test.JOINT_SHOULDER_AXIS1_RIGHT = "right_shoulder_axis1"
string steering_response_test.JOINT_SHOULDER_AXIS2_RIGHT = "right_shoulder_axis2"
string steering_response_test.JOINT_SHOULDER_AXIS0_LEFT = "left_shoulder_axis0"
string steering_response_test.JOINT_SHOULDER_AXIS1_LEFT = "left_shoulder_axis1"
string steering_response_test.JOINT_SHOULDER_AXIS2_LEFT = "left_shoulder_axis2"
string steering_response_test.JOINT_ELBOW_ROT0_RIGHT = "elbow_right_rot0"
string steering_response_test.JOINT_ELBOW_ROT1_RIGHT = "elbow_right_rot1"
string steering_response_test.JOINT_ELBOW_ROT0_LEFT = "elbow_left_rot0"
string steering_response_test.JOINT_ELBOW_ROT1_LEFT = "elbow_left_rot1"
string steering_response_test.JOINT_WRIST_0_RIGHT = "right_wrist_0"
string steering_response_test.JOINT_WRIST_1_RIGHT = "right_wrist_1"
string steering_response_test.JOINT_WRIST_0_LEFT = "left_wrist_0"
string steering_response_test.JOINT_WRIST_1_LEFT = "left_wrist_1"
dictionary steering_response_test.joint_status_data= { JOINT_SHOULDER_AXIS0_LEFT:
list steering_response_test._joints_list= [JOINT_SHOULDER_AXIS0_RIGHT, JOINT_SHOULDER
steering_response_test.ros_right_shoulder_axis0_pub = rospy.Publisher('/right_shoulder
steering_response_test.ros_right_shoulder_axis1_pub = rospy.Publisher('/right_shoulder
steering_response_test.ros_right_shoulder_axis2_pub = rospy.Publisher('/right_shoulder
```

```
steering_response_test.ros_left_shoulder_axis0_pub = rospy.Publisher('/left_shoulder_
steering_response_test.ros_left_shoulder_axis1_pub = rospy.Publisher('/left_shoulder_
steering_response_test.ros_left_shoulder_axis2_pub = rospy.Publisher('/left_shoulder_
steering_response_test.ros_elbow_right_rot0_pub = rospy.Publisher('/elbow_right_rot0/
steering_response_test.ros_elbow_right_rot1_pub = rospy.Publisher('/elbow_right_rot1/
steering_response_test.ros_elbow_left_rot0_pub = rospy.Publisher('/elbow_left_rot0/el
steering_response_test.ros_elbow_left_rot1_pub = rospy.Publisher('/elbow_left_rot1/el
steering_response_test.ros_right_wrist_0_pub = rospy.Publisher('/right_wrist_0/right_
steering_response_test.ros_right_wrist_1_pub = rospy.Publisher('/right_wrist_1/right_
steering_response_test.ros_left_wrist_0_pub = rospy.Publisher('/left_wrist_0/left_wri
steering_response_test.ros_left_wrist_1_pub = rospy.Publisher('/left_wrist_1/left_wri
int steering_response_test._numTrajectoryPoints = 0
list steering_response_test._trajectorySteering = [ ]
list steering_response_test._trajectoryShoulder0Right = [ ]
list steering_response_test._trajectoryShoulder1Right = [ ]
list steering_response_test._trajectoryShoulder2Right = [ ]
list steering_response_test._trajectoryShoulder0Left = [ ]
list steering_response_test._trajectoryShoulder1Left = [ ]
list steering_response_test._trajectoryShoulder2Left = [ ]
list steering_response_test._trajectoryElbow0Right = [ ]
list steering_response_test._trajectoryElbow1Right = [ ]
list steering_response_test._trajectoryElbow0Left = [ ]
list steering_response_test._trajectoryElbow1Left = [ ]
list steering_response_test._trajectoryWrist0Right = [ ]
list steering_response_test._trajectoryWrist1Right = [ ]
list steering_response_test._trajectoryWrist0Left = [ ]
list steering_response_test._trajectoryWrist1Left = [ ]
steering_response_test._interpolatedShoulder0Right = None
steering_response_test._interpolatedShoulder1Right = None
steering_response_test._interpolatedShoulder2Right = None
steering_response_test._interpolatedShoulder0Left = None
steering_response_test._interpolatedShoulder1Left = None
steering_response_test._interpolatedShoulder2Left = None
steering_response_test._interpolatedElbow0Right = None
steering_response_test._interpolatedElbow1Right = None
steering_response_test._interpolatedElbow0Left = None
```

```

steering_response_test._interpolatedElbow1Left = None
steering_response_test._interpolatedWrist0Right = None
steering_response_test._interpolatedWrist1Right = None
steering_response_test._interpolatedWrist0Left = None
steering_response_test._interpolatedWrist1Left = None
steering_response_test._regressedShoulder0Right = None
steering_response_test._regressedShoulder1Right = None
steering_response_test._regressedShoulder2Right = None
steering_response_test._regressedShoulder0Left = None
steering_response_test._regressedShoulder1Left = None
steering_response_test._regressedShoulder2Left = None
steering_response_test._regressedElbow0Right = None
steering_response_test._regressedElbow1Right = None
steering_response_test._regressedElbow0Left = None
steering_response_test._regressedElbow1Left = None
steering_response_test._regressedWrist0Right = None
steering_response_test._regressedWrist1Right = None
steering_response_test._regressedWrist0Left = None
steering_response_test._regressedWrist1Left = None
string steering_response_test.RECORDED_TRAJECTORY_FILENAME = "trajectory_steering/ste
namespace steering_simulation

```

Functions

joint_state_callback (*joint_data joint_data*)

Documentation for a function.

This function collects the current status of the joint-angles and saves them in the global dictionary “_jointStatusData”.

get_joint_position (*joint_name joint_name*)

Documentation for a function.

Returns current position of joint-angle . .

regress_joint_positions_from_file (*filename filename*)

Documentation for a function.

Initializes the interpolation-functions for every joint-angle using regression. The input value of the function is a steering angle and the output value of the function is the correspondent joint angle.

The functions can be used by calling “<function_name>(<steering_angle>)” ==> returns <joint_angle>

import_joint_trajectory_record ()

Documentation for a function.

Collects and saves all joint- and steering-angles from the pre-captured trajectory from the (global variable).

interpolate_joint_angles()

Documentation for a function.

Initializes the interpolation-functions for every joint-angle using cubic spline interpolation. The input value of the function is a steering angle and the output value of the function the correspondent joint angle.

The functions can be used by calling “<function_name>(<steering_angle>)” ==> returns <joint_angle>

get_angle_difference (*angle_1 angle_1, angle_2 angle_2*)

Documentation for a function.

Returns the absolute difference of two angles within the interval [0;2pi]

set_joint_controller_parameters (*proportional_value proportional_value, derivative_value derivative_value*)

Documentation for a function.

Sets Kp of joint-controller to Sets Kd of joint-controller to

update_steering_angle (*steering_angle_F32 steering_angle_F32*)

Documentation for a function.

Updates the global variable when another node publishes a new requested steering_angle to the topic “cmd_steering_angle_rickshaw”.

check_steering_angle_range (*steering_angle steering_angle*)

Documentation for a function.

Checks if the parameter is within the range of reachable steering-angles of Roboy.

publish_joint_angle (*joint_name joint_name, steering_angle steering_angle*)

Documentation for a function.

Evaluates the correspondent joint-angle of to given using the interpolation-function of

Publishes the computed value to the correspondent ros-topic of to apply position control.

Waits until the joint_angle has reached requested joint_angle within error tolerance.

steering_control()

Documentation for a function.

Controls the whole steering-process. Evaluates the target_steering_angles between requested_steering_angle and current_steering_angle and creates a Thread for every joint-angle with which is responsible to apply correspondent joint-angle given current target_steering_angle.

Simplified Pseudo-code:

while requested_steering_angle = current_steering_angle:

```
sleep()
```

if angle_difference(requested_steering_angle, current_steering_angle) > max_angle_change

```
target_steering_angle = current_steering_angle + max_angle_change
```

else:

```
target_steering_angle = requested_steering_angle
```

for joint in joint_list:

```
Thread.publish_joint_angle(joint_name, target_joint_angle)
```


for Thread in created_threads:

```
Thread.join
```

current_steering_angle = target_steering_angle

main()

Documentation for a function.

Initializes the Control-Node for Steering and starts Steering-Algorithm.

Variables

```
bool steering_simulation.PRINT_DEBUG = True
string steering_simulation.RECORDED_TRAJECTORY_FILENAME = "trajectory_steering/steeri
float steering_simulation.JOINT_TARGET_ERROR_TOLERANCE = 0.01
float steering_simulation.UPDATE_FREQUENCY = 0.001
int steering_simulation.MAX_ANGLE_CHANGE = np.pi / 72
float steering_simulation.STEP_TRANSITION_TIME = 2.5
string steering_simulation.JOINT_SHOULDER_AXIS0_RIGHT = "right_shoulder_axis0"
string steering_simulation.JOINT_SHOULDER_AXIS1_RIGHT = "right_shoulder_axis1"
string steering_simulation.JOINT_SHOULDER_AXIS2_RIGHT = "right_shoulder_axis2"
string steering_simulation.JOINT_SHOULDER_AXIS0_LEFT = "left_shoulder_axis0"
string steering_simulation.JOINT_SHOULDER_AXIS1_LEFT = "left_shoulder_axis1"
string steering_simulation.JOINT_SHOULDER_AXIS2_LEFT = "left_shoulder_axis2"
string steering_simulation.JOINT_ELBOW_RIGHT = "elbow_right"
string steering_simulation.JOINT_ELBOW_LEFT = "elbow_left"
string steering_simulation.JOINT_WRIST_RIGHT_SPHERE_AXIS0 = "wrist_right_sphere_axis0"
string steering_simulation.JOINT_WRIST_RIGHT_SPHERE_AXIS1 = "wrist_right_sphere_axis1"
string steering_simulation.JOINT_WRIST_RIGHT_SPHERE_AXIS2 = "wrist_right_sphere_axis2"
string steering_simulation.JOINT_WRIST_LEFT_SPHERE_AXIS0 = "wrist_left_sphere_axis0"
string steering_simulation.JOINT_WRIST_LEFT_SPHERE_AXIS1 = "wrist_left_sphere_axis1"
string steering_simulation.JOINT_WRIST_LEFT_SPHERE_AXIS2 = "wrist_left_sphere_axis2"
string steering_simulation.JOINT_BIKE_FRONT = "joint_front"
list steering_simulation._joints_list= [JOINT_SHOULDER_AXIS0_RIGHT, JOINT_SHOULDER_AX
int steering_simulation._numTrajectoryPoints = 0
list steering_simulation._trajectorySteering = []
list steering_simulation._trajectoryShoulder0Right = []
list steering_simulation._trajectoryShoulder1Right = []
list steering_simulation._trajectoryShoulder2Right = []
list steering_simulation._trajectoryShoulder0Left = []
```

```
list steering_simulation._trajectoryShoulder1Left = []
list steering_simulation._trajectoryShoulder2Left = []
list steering_simulation._trajectoryElbowRight = []
list steering_simulation._trajectoryElbowLeft = []
list steering_simulation._trajectoryWrist0Right = []
list steering_simulation._trajectoryWrist1Right = []
list steering_simulation._trajectoryWrist2Right = []
list steering_simulation._trajectoryWrist0Left = []
list steering_simulation._trajectoryWrist1Left = []
list steering_simulation._trajectoryWrist2Left = []
steering_simulation._interpolatedShoulder0Right = None
steering_simulation._interpolatedShoulder1Right = None
steering_simulation._interpolatedShoulder2Right = None
steering_simulation._interpolatedShoulder0Left = None
steering_simulation._interpolatedShoulder1Left = None
steering_simulation._interpolatedShoulder2Left = None
steering_simulation._interpolatedElbowRight = None
steering_simulation._interpolatedElbowLeft = None
steering_simulation._interpolatedWrist0Right = None
steering_simulation._interpolatedWrist1Right = None
steering_simulation._interpolatedWrist2Right = None
steering_simulation._interpolatedWrist0Left = None
steering_simulation._interpolatedWrist1Left = None
steering_simulation._interpolatedWrist2Left = None
steering_simulation._regressedShoulder0Right = None
steering_simulation._regressedShoulder1Right = None
steering_simulation._regressedShoulder2Right = None
steering_simulation._regressedShoulder0Left = None
steering_simulation._regressedShoulder1Left = None
steering_simulation._regressedShoulder2Left = None
steering_simulation._regressedElbowRight = None
steering_simulation._regressedElbowLeft = None
steering_simulation._regressedWrist0Right = None
steering_simulation._regressedWrist1Right = None
steering_simulation._regressedWrist2Right = None
steering_simulation._regressedWrist0Left = None
```

```

steering_simulation._regressedWrist1Left = None
steering_simulation._regressedWrist2Left = None
dictionary steering_simulation._jointsStatusData
steering_simulation.ros_right_shoulder_axis0_pub= rospy.Publisher('/right_shoulder_ax
steering_simulation.ros_right_shoulder_axis1_pub= rospy.Publisher('/right_shoulder_ax
steering_simulation.ros_right_shoulder_axis2_pub= rospy.Publisher('/right_shoulder_ax
steering_simulation.ros_left_shoulder_axis0_pub = rospy.Publisher('/left_shoulder_axi
steering_simulation.ros_left_shoulder_axis1_pub = rospy.Publisher('/left_shoulder_axi
steering_simulation.ros_left_shoulder_axis2_pub = rospy.Publisher('/left_shoulder_axi
steering_simulation.ros_elbow_right_pub = rospy.Publisher('/elbow_right/elbow_right/t
steering_simulation.ros_elbow_left_pub = rospy.Publisher('/elbow_left/elbow_left/targ
steering_simulation.ros_right_wrist_0_pub = rospy.Publisher('/wrist_right_sphere_axis
steering_simulation.ros_right_wrist_1_pub = rospy.Publisher('/wrist_right_sphere_axis
steering_simulation.ros_right_wrist_2_pub = rospy.Publisher('/wrist_right_sphere_axis
steering_simulation.ros_left_wrist_0_pub = rospy.Publisher('/wrist_left_sphere_axis0/
steering_simulation.ros_left_wrist_1_pub = rospy.Publisher('/wrist_left_sphere_axis1/
steering_simulation.ros_left_wrist_2_pub = rospy.Publisher('/wrist_left_sphere_axis2/
steering_simulation.ros_bike_front_pub = rospy.Publisher('/joint_front/joint_front/ta
steering_simulation.ros_log_error_pub = rospy.Publisher('chatter', String, queue_size
int steering_simulation.requested_steering_angle = 0
bool steering_simulation.angle_change_successful = True
namespace steering_simulation_old_hand

```

Functions

joint_state_callback (*joint_data joint_data*)

Documentation for a function.

This function collects the current status of the joint-angles and saves them in the global dictionary “_jointStatusData”.

get_joint_position (*joint_name joint_name*)

Documentation for a function.

Returns current position of joint-angle . .

regress_joint_positions_from_file (*filename filename*)

Documentation for a function.

Initializes the interpolation-functions for every joint-angle using regression. The input value of the function is a steering angle and the output value of the function is the correspondent joint angle.

The functions can be used by calling “<function_name>(<steering_angle>)” ==> returns <joint_angle>

import_joint_trajectory_record()

Documentation for a function.

Collects and saves all joint- and steering-angles from the pre-captured trajectory from the (global variable).

interpolate_joint_angles()

Documentation for a function.

Initializes the interpolation-functions for every joint-angle using cubic spline interpolation. The input value of the function is a steering angle and the output value of the function the correspondent joint angle.

The functions can be used by calling “<function_name>(<steering_angle>)” ==> returns <joint_angle>

get_angle_difference (*angle_1 angle_1, angle_2 angle_2*)

Documentation for a function.

Returns the absolute difference of two angles within the interval [0;2pi]

set_joint_controller_parameters (*proportional_value proportional_value, derivative_value derivative_value*)

Documentation for a function.

Sets Kp of joint-controller to Sets Kd of joint-controller to

update_steering_angle (*steering_angle_F32 steering_angle_F32*)

Documentation for a function.

Updates the global variable when another node publishes a new requested steering_angle to the topic “cmd_steering_angle_rickshaw”.

check_steering_angle_range (*steering_angle steering_angle*)

Documentation for a function.

Checks if the parameter is within the range of reachable steering-angles of Roboy.

publish_joint_angle (*joint_name joint_name, steering_angle steering_angle*)

Documentation for a function.

Evaluates the correspondent joint-angle of to given using the interpolation-function of

Publishes the computed value to the correspondent ros-topic of to apply position control.

Waits until the joint_angle has reached requested joint_angle within error tolerance.

steering_control()

Documentation for a function.

Controls the whole steering-process. Evaluates the target_steering_angles between requested_steering_angle and current_steering_angle and creates a Thread for every joint-angle with which is responsible to apply correspondent joint-angle given current target_steering_angle.

Simplified Pseudo-code:

while requested_steering_angle = current_steering_angle:

```
sleep()
```

if angle_difference(requested_steering_angle, current_steering_angle) > max_angle_change

```
target_steering_angle = current_steering_angle + max_angle_change
```

else:

```
target_steering_angle = requested_steering_angle
```

```
for joint in joint_list:
```

```
    Thread.publish_joint_angle(joint_name, target_joint_angle)
```

```
for Thread in created_threads:
```

```
    Thread.join
```

```
current_steering_angle = target_steering_angle
```

```
main()
```

```
    Documentation for a function.
```

```
    Initializes the Control-Node for Steering and starts Steering-Algorithm.
```

Variables

```
bool steering_simulation_old_hand.PRINT_DEBUG = True
```

```
string steering_simulation_old_hand.RECORDED_TRAJECTORY_FILENAME = "trajectory_steering"
```

```
float steering_simulation_old_hand.JOINT_TARGET_ERROR_TOLERANCE = 0.01
```

```
float steering_simulation_old_hand.UPDATE_FREQUENCY = 0.001
```

```
int steering_simulation_old_hand.MAX_ANGLE_CHANGE = np.pi / 72
```

```
float steering_simulation_old_hand.STEP_TRANSITION_TIME = 2.5
```

```
string steering_simulation_old_hand.JOINT_SHOULDER_AXIS0_RIGHT = "right_shoulder_axis0"
```

```
string steering_simulation_old_hand.JOINT_SHOULDER_AXIS1_RIGHT = "right_shoulder_axis1"
```

```
string steering_simulation_old_hand.JOINT_SHOULDER_AXIS2_RIGHT = "right_shoulder_axis2"
```

```
string steering_simulation_old_hand.JOINT_SHOULDER_AXIS0_LEFT = "left_shoulder_axis0"
```

```
string steering_simulation_old_hand.JOINT_SHOULDER_AXIS1_LEFT = "left_shoulder_axis1"
```

```
string steering_simulation_old_hand.JOINT_SHOULDER_AXIS2_LEFT = "left_shoulder_axis2"
```

```
string steering_simulation_old_hand.JOINT_ELBOW_ROT0_RIGHT = "elbow_right_rot0"
```

```
string steering_simulation_old_hand.JOINT_ELBOW_ROT1_RIGHT = "elbow_right_rot1"
```

```
string steering_simulation_old_hand.JOINT_ELBOW_ROT0_LEFT = "elbow_left_rot0"
```

```
string steering_simulation_old_hand.JOINT_ELBOW_ROT1_LEFT = "elbow_left_rot1"
```

```
string steering_simulation_old_hand.JOINT_WRIST_0_RIGHT = "right_wrist_0"
```

```
string steering_simulation_old_hand.JOINT_WRIST_1_RIGHT = "right_wrist_1"
```

```
string steering_simulation_old_hand.JOINT_WRIST_0_LEFT = "left_wrist_0"
```

```
string steering_simulation_old_hand.JOINT_WRIST_1_LEFT = "left_wrist_1"
```

```
string steering_simulation_old_hand.JOINT_BIKE_FRONT = "joint_front"
```

```
list steering_simulation_old_hand._joints_list= [JOINT_SHOULDER_AXIS0_RIGHT, JOINT_SHOULDER_AXIS1_RIGHT, JOINT_SHOULDER_AXIS2_RIGHT, JOINT_SHOULDER_AXIS0_LEFT, JOINT_SHOULDER_AXIS1_LEFT, JOINT_SHOULDER_AXIS2_LEFT, JOINT_ELBOW_ROT0_RIGHT, JOINT_ELBOW_ROT1_RIGHT, JOINT_ELBOW_ROT0_LEFT, JOINT_ELBOW_ROT1_LEFT, JOINT_WRIST_0_RIGHT, JOINT_WRIST_1_RIGHT, JOINT_WRIST_0_LEFT, JOINT_WRIST_1_LEFT, JOINT_BIKE_FRONT]
```

```
int steering_simulation_old_hand._numTrajectoryPoints = 0
```

```
list steering_simulation_old_hand._trajectorySteering = []
```

```
list steering_simulation_old_hand._trajectoryShoulder0Right = []
```

```
list steering_simulation_old_hand._trajectoryShoulder1Right = []
list steering_simulation_old_hand._trajectoryShoulder2Right = []
list steering_simulation_old_hand._trajectoryShoulder0Left = []
list steering_simulation_old_hand._trajectoryShoulder1Left = []
list steering_simulation_old_hand._trajectoryShoulder2Left = []
list steering_simulation_old_hand._trajectoryElbow0Right = []
list steering_simulation_old_hand._trajectoryElbow1Right = []
list steering_simulation_old_hand._trajectoryElbow0Left = []
list steering_simulation_old_hand._trajectoryElbow1Left = []
list steering_simulation_old_hand._trajectoryWrist0Right = []
list steering_simulation_old_hand._trajectoryWrist1Right = []
list steering_simulation_old_hand._trajectoryWrist0Left = []
list steering_simulation_old_hand._trajectoryWrist1Left = []
steering_simulation_old_hand._interpolatedShoulder0Right = None
steering_simulation_old_hand._interpolatedShoulder1Right = None
steering_simulation_old_hand._interpolatedShoulder2Right = None
steering_simulation_old_hand._interpolatedShoulder0Left = None
steering_simulation_old_hand._interpolatedShoulder1Left = None
steering_simulation_old_hand._interpolatedShoulder2Left = None
steering_simulation_old_hand._interpolatedElbow0Right = None
steering_simulation_old_hand._interpolatedElbow1Right = None
steering_simulation_old_hand._interpolatedElbow0Left = None
steering_simulation_old_hand._interpolatedElbow1Left = None
steering_simulation_old_hand._interpolatedWrist0Right = None
steering_simulation_old_hand._interpolatedWrist1Right = None
steering_simulation_old_hand._interpolatedWrist0Left = None
steering_simulation_old_hand._interpolatedWrist1Left = None
steering_simulation_old_hand._regressedShoulder0Right = None
steering_simulation_old_hand._regressedShoulder1Right = None
steering_simulation_old_hand._regressedShoulder2Right = None
steering_simulation_old_hand._regressedShoulder0Left = None
steering_simulation_old_hand._regressedShoulder1Left = None
steering_simulation_old_hand._regressedShoulder2Left = None
steering_simulation_old_hand._regressedElbow0Right = None
steering_simulation_old_hand._regressedElbow1Right = None
steering_simulation_old_hand._regressedElbow0Left = None
```

```

steering_simulation_old_hand._regressedElbow1Left = None
steering_simulation_old_hand._regressedWrist0Right = None
steering_simulation_old_hand._regressedWrist1Right = None
steering_simulation_old_hand._regressedWrist0Left = None
steering_simulation_old_hand._regressedWrist1Left = None
dictionary steering_simulation_old_hand._jointsStatusData
steering_simulation_old_hand.ros_right_shoulder_axis0_pub= rospy.Publisher('/right_sho
steering_simulation_old_hand.ros_right_shoulder_axis1_pub= rospy.Publisher('/right_sho
steering_simulation_old_hand.ros_right_shoulder_axis2_pub= rospy.Publisher('/right_sho
steering_simulation_old_hand.ros_left_shoulder_axis0_pub = rospy.Publisher('/left_sho
steering_simulation_old_hand.ros_left_shoulder_axis1_pub = rospy.Publisher('/left_sho
steering_simulation_old_hand.ros_left_shoulder_axis2_pub = rospy.Publisher('/left_sho
steering_simulation_old_hand.ros_elbow_right_rot0_pub = rospy.Publisher('/elbow_right
steering_simulation_old_hand.ros_elbow_right_rot1_pub = rospy.Publisher('/elbow_right
steering_simulation_old_hand.ros_elbow_left_rot0_pub = rospy.Publisher('/elbow_left_r
steering_simulation_old_hand.ros_elbow_left_rot1_pub = rospy.Publisher('/elbow_left_r
steering_simulation_old_hand.ros_right_wrist_0_pub = rospy.Publisher('/right_wrist_0/
steering_simulation_old_hand.ros_right_wrist_1_pub = rospy.Publisher('/right_wrist_1/
steering_simulation_old_hand.ros_left_wrist_0_pub = rospy.Publisher('/left_wrist_0/le
steering_simulation_old_hand.ros_left_wrist_1_pub = rospy.Publisher('/left_wrist_1/le
steering_simulation_old_hand.ros_bike_front_pub = rospy.Publisher('/joint_front/joint
steering_simulation_old_hand.ros_log_error_pub = rospy.Publisher('chatter', String, q
int steering_simulation_old_hand.requested_steering_angle = 0
bool steering_simulation_old_hand.angle_change_successful = True
namespace steering_trajectory_following_test

```

Functions

```

regressFunctionsFromFile (filename_coefficients filename_coefficients)
    UTILITY FUNCTIONS ###.
setJointControllerParameters (proportionalVal proportionalVal, derivativeVal derivativeVal)
computeSteeringAngles ()
computeHandTrajectories ()
getPositionLeftHand ()
getPositionRightHand ()
jointStateCallback (joint_data joint_data)

```

```

recordActualHandTrajectories (ros_right_shoulder0_publisher
                               ros_right_shoulder0_publisher,
                               ros_right_shoulder1_publisher
                               ros_right_shoulder1_publisher,
                               ros_right_shoulder2_publisher
                               ros_right_shoulder2_publisher, ros_right_elbow0_publisher
                               ros_right_elbow0_publisher,   ros_right_elbow1_publisher
                               ros_right_elbow1_publisher,   ros_right_wrist0_publisher
                               ros_right_wrist0_publisher,   ros_right_wrist1_publisher
                               ros_right_wrist1_publisher,   ros_left_shoulder0_publisher
                               ros_left_shoulder0_publisher, ros_left_shoulder1_publisher
                               ros_left_shoulder1_publisher, ros_left_shoulder2_publisher
                               ros_left_shoulder2_publisher, ros_left_elbow0_publisher
                               ros_left_elbow0_publisher,   ros_left_elbow1_publisher
                               ros_left_elbow1_publisher,   ros_left_wrist0_publisher
                               ros_left_wrist0_publisher,   ros_left_wrist1_publisher
                               ros_left_wrist1_publisher)

printPlannedActualHandTrajectories ()

main ()
    MAIN ###.

```

Variables

```

int steering_trajectory_following_test.MAX_TURNING_ANGLE =  math.pi/15
    MODULE PARAMETERS ###.

int steering_trajectory_following_test.NUM_STEERING_ANGLES =  61

float steering_trajectory_following_test.JOINT_ANGLE_TOLERANCE_FK =  0.01

string steering_trajectory_following_test.ENDEFFECTOR_RIGHT =  "right_hand"
string steering_trajectory_following_test.FRAME_RIGHT =  "right_hand"
string steering_trajectory_following_test.ENDEFFECTOR_LEFT =  "left_hand"
string steering_trajectory_following_test.FRAME_LEFT =  "left_hand"

int steering_trajectory_following_test.BIKE_OFFSET_X =  0
    MEASURED PARAMETERS ###.

int steering_trajectory_following_test.BIKE_OFFSET_Y =  0
int steering_trajectory_following_test.BIKE_OFFSET_Z =  0

float steering_trajectory_following_test.RIKSHAW_TURN_JOINT_X_OFFSET =  0.716390260057
float steering_trajectory_following_test.RIKSHAW_TURN_JOINT_Y_OFFSET =  -0.01038855246
float steering_trajectory_following_test.RIKSHAW_TURN_JOINT_Z_OFFSET =  0.216437694214
int steering_trajectory_following_test.YAW_RIGHT_HAND_OFFSET =  math.pi / 2 + math.pi
int steering_trajectory_following_test.YAW_LEFT_HAND_OFFSET =  3 * math.pi / 2 + math.pi
float steering_trajectory_following_test.HANDLEBAR_X_OFFSET =  0.728713
float steering_trajectory_following_test.HANDLEBAR_Z_OFFSET =  0.719269
float steering_trajectory_following_test.HAND_Y_OFFSET =  0.2

```

```

list steering_trajectory_following_test._steeringAngles = []
GLOBAL VARIABLES ###.

list steering_trajectory_following_test._rightHandTrajectoryPlanned = []
list steering_trajectory_following_test._leftHandTrajectoryPlanned = []
list steering_trajectory_following_test._centerHandlebarTrajectoryPlanned = []
list steering_trajectory_following_test._rightHandTrajectoryActual = []
list steering_trajectory_following_test._leftHandTrajectoryActual = []
string steering_trajectory_following_test.JOINT_SHOULDER_AXIS0_RIGHT = "right_shoulder0"
string steering_trajectory_following_test.JOINT_SHOULDER_AXIS1_RIGHT = "right_shoulder1"
string steering_trajectory_following_test.JOINT_SHOULDER_AXIS2_RIGHT = "right_shoulder2"
string steering_trajectory_following_test.JOINT_SHOULDER_AXIS0_LEFT = "left_shoulder0"
string steering_trajectory_following_test.JOINT_SHOULDER_AXIS1_LEFT = "left_shoulder1"
string steering_trajectory_following_test.JOINT_SHOULDER_AXIS2_LEFT = "left_shoulder2"
string steering_trajectory_following_test.JOINT_ELBOW_ROT0_RIGHT = "elbow_right_rot0"
string steering_trajectory_following_test.JOINT_ELBOW_ROT1_RIGHT = "elbow_right_rot1"
string steering_trajectory_following_test.JOINT_ELBOW_ROT0_LEFT = "elbow_left_rot0"
string steering_trajectory_following_test.JOINT_ELBOW_ROT1_LEFT = "elbow_left_rot1"
string steering_trajectory_following_test.JOINT_WRIST_0_RIGHT = "right_wrist_0"
string steering_trajectory_following_test.JOINT_WRIST_1_RIGHT = "right_wrist_1"
string steering_trajectory_following_test.JOINT_WRIST_0_LEFT = "left_wrist_0"
string steering_trajectory_following_test.JOINT_WRIST_1_LEFT = "left_wrist_1"
list steering_trajectory_following_test.JOINT_LIST = [JOINT_SHOULDER_AXIS0_RIGHT, JOINT_SHOULDER_AXIS1_RIGHT, JOINT_SHOULDER_AXIS2_RIGHT, JOINT_SHOULDER_AXIS0_LEFT, JOINT_SHOULDER_AXIS1_LEFT, JOINT_SHOULDER_AXIS2_LEFT, JOINT_ELBOW_ROT0_RIGHT, JOINT_ELBOW_ROT1_RIGHT, JOINT_ELBOW_ROT0_LEFT, JOINT_ELBOW_ROT1_LEFT, JOINT_WRIST_0_RIGHT, JOINT_WRIST_1_RIGHT, JOINT_WRIST_0_LEFT, JOINT_WRIST_1_LEFT]
steering_trajectory_following_test._regressedShoulder0Right = None
steering_trajectory_following_test._regressedShoulder1Right = None
steering_trajectory_following_test._regressedShoulder2Right = None
steering_trajectory_following_test._regressedShoulder0Left = None
steering_trajectory_following_test._regressedShoulder1Left = None
steering_trajectory_following_test._regressedShoulder2Left = None
steering_trajectory_following_test._regressedElbow0Right = None
steering_trajectory_following_test._regressedElbow1Right = None
steering_trajectory_following_test._regressedElbow0Left = None
steering_trajectory_following_test._regressedElbow1Left = None
steering_trajectory_following_test._regressedWrist0Right = None
steering_trajectory_following_test._regressedWrist1Right = None
steering_trajectory_following_test._regressedWrist0Left = None
steering_trajectory_following_test._regressedWrist1Left = None

```

dictionary steering_trajectory_following_test._jointsStatusData
namespace velocity_test

Functions

get_twist (*velocity velocity*)

Documentation for a function.

Return a Twist-msg with the linear velocity #velocity

get_position_left_foot ()

Documentation for a function.

Return the position of the left foot of Roboy.

get_position_right_foot ()

Documentation for a function.

Return the position of the right foot of Roboy.

evaluate_current_pedal_angle (*current_point current_point*)

Documentation for a function.

Evaluating the current pedal-angle according to the current position of the left foot . Using trigonometric functions for the evaluation of the angle.

get_angle_difference (*angle_1 angle_1, angle_2 angle_2*)

Documentation for a function.

Returns the absolute difference of two angles within the interval $[0;2\pi]$

evaluate_error (*velocity velocity, leg leg*)

Documentation for a function.

Evaluate the angle-error for velocity after seconds

simulation_test (*pub pub*)

Documentation for a function.

Test program for evaluating the accuracy of velocity. Saves ten error-results for every target velocity and determines average error and max error for every velocity and displays them using pyplot.

velocity_reached (*velocity velocity*)

Documentation for a function.

Returns if has been reached within the error-tolerance

reality_test_acceleration (*pub pub*)

Documentation for a function.

Test program for evaluating the acceleration time (time needed to change between two velocities.

Uses to determine the difference between two different velocities and measures the time needed to change between the two velocities. Displays all acceleration times using pyplot.

main ()

Documentation for a function.

Initializes the Test-Node for the velocity-tests

Variables

```

float velocity_test.TIME_STEP_SIMULATION = 0.5
float velocity_test.TIME_STEP_REALITY = 0.5
float velocity_test.VELOCITY_STEP_SIMULATION = 0.1
list velocity_test.VELOCITY_STEPS_REALITY = [0.1, 0.2, 0.5]
int velocity_test.ERROR_TOLERANCE_REALITY = 2*np.pi / 720
int velocity_test.MAX_VELOCITY = 5
float velocity_test.PEDAL_CENTER_OFFSET_X = 0.20421
float velocity_test.PEDAL_CENTER_OFFSET_Y = -0.00062
float velocity_test.PEDAL_CENTER_OFFSET_Z = 0.2101
float velocity_test.RADIUS_BACK_TIRE = 0.294398
float velocity_test.RADIUS_GEAR_CLUSTER = 0.06
float velocity_test.RADIUS_FRONT_CHAIN_RING = 0.075
string velocity_test.ROS_JOINT_HIP_RIGHT = "joint_hip_right"
string velocity_test.ROS_JOINT_KNEE_RIGHT = "joint_knee_right"
string velocity_test.ROS_JOINT_ANKLE_RIGHT = "joint_foot_right"
string velocity_test.ROS_JOINT_HIP_LEFT = "joint_hip_left"
string velocity_test.ROS_JOINT_KNEE_LEFT = "joint_knee_left"
string velocity_test.ROS_JOINT_ANKLE_LEFT = "joint_foot_left"
string velocity_test.RIGHT_HIP_JOINT = "right_hip"
string velocity_test.RIGHT_KNEE_JOINT = "right_knee"
string velocity_test.RIGHT_ANKLE_JOINT = "right_ankle"
string velocity_test.LEFT_HIP_JOINT = "left_hip"
string velocity_test.LEFT_KNEE_JOINT = "left_knee"
string velocity_test.LEFT_ANKLE_JOINT = "left_ankle"
dictionary velocity_test.joint_status_data= {    RIGHT_HIP_JOINT: {    "Pos": 0.0
namespace visualization_msgs

file cardsflow_rviz.hpp
#include <QPainter>#include <QCheckBox>#include <QPushButton>#include <QLineEdit>#include
<QSlider>#include <QVBoxLayout>#include <QHBoxLayout>#include <QLabel>#include
<QTableWidget>#include <QComboBox>#include <QTimer>#include <QScrollArea>#include
<QListWidget>#include <QStyledItemDelegate>#include <ros/ros.h>#include <tf/tf.h>#include
<tf/transform_listener.h>#include <tf/transform_broadcaster.h>#include <tf_conversions/tf_eigen.h>#include
<Eigen/Core>#include <Eigen/Dense>#include <rviz/panel.h>#include <pluginlib/class_loader.h>#include
<pluginlib/class_list_macros.h>#include <common_utilities/rviz_visualization.hpp>#include
<geometry_msgs/PoseStamped.h>#include <robo_simulation_msgs/Tendon.h>#include
<moveit_msgs/DisplayRobotState.h>#include <robo_simulation_msgs/JointState.h>#include
<map>#include <thread>

```

```
file cardsflow_rviz.cpp
    #include "include/cardsflow_rviz/cardsflow_rviz.hpp"

file measureExecutionTime.hpp
    #include <string>#include <chrono>#include <boost/shared_ptr.hpp>#include <fstream>
```

Typedefs

```
using Clock = std::chrono::steady_clock

typedef boost::shared_ptr<MeasureExecutionTime> MeasureExecutionTimePtr

file MotorConfig.hpp
    #include <ros/ros.h>#include <map>#include <yaml-cpp/yaml.h>#include <fstream>#include
    <sys/types.h>#include <sys/stat.h>#include <unistd.h>#include <common_utilities/CommonDefinitions.h>

file rviz_visualization.hpp
    #include <ros/ros.h>#include <ros/package.h>#include <visualization_msgs/Marker.h>#include <visu-
    alization_msgs/MarkerArray.h>#include <Eigen/Core>#include <Eigen/Dense>#include <tf/tf.h>#include
    <tf/transform_listener.h>#include <tf/transform_broadcaster.h>#include <tf_conversions/tf_eigen.h>#include
    <interactive_markers/menu_handler.h>#include <interactive_markers/interactive_marker_server.h>#include
    <geometry_msgs/Pose.h>#include <string>#include <sys/stat.h>

file UDPSocket.hpp
    #include <stdlib.h>#include <unistd.h>#include <errno.h>#include <string.h>#include
    <sys/types.h>#include <sys/socket.h>#include <netinet/in.h>#include <arpa/inet.h>#include
    <netdb.h>#include <string>#include <ifaddrs.h>#include <stdio.h>#include <time.h>#include <bit-
    set>#include <iostream>#include <boost/shared_ptr.hpp>#include <vector>#include <ros/ros.h>
```

Defines

```
MAXBUFLLENGTH

BYTE_TO_BINARY_PATTERN

BYTE_TO_BINARY (byte)

BROADCAST_PORT

pack754_32 (f)

pack754_64 (f)

unpack754_32 (i)

unpack754_64 (i)
```

Typedefs

```
typedef boost::shared_ptr<UDPSocket> UDPSocketPtr
```

Functions

```
uint64_t pack754 (long double f, unsigned bits, unsigned expbits)

long double unpack754 (uint64_t i, unsigned bits, unsigned expbits)
```

```
file MotorConfig.cpp
    #include "common_utilities/MotorConfig.hpp"
```

```
file rfid_unlocker.cpp
    #include "common_utilities/UDPSocket.hpp" #include <chrono>
```

Functions

```
int main (int argc, char *argv[])
```

Variables

```
const char* key= "The path of the righteous man is beset on all sides by the inequities of the world and the  
uint8_t user[4] = {0xBF, , , }
```

```
file ROS_MASTER_URI_broadcaster.cpp
    #include <ros/ros.h> #include "common_utilities/UDPSocket.hpp"
```

Functions

```
int main (int argc, char *argv[])
```

Variables

```
const char* key= "The path of the righteous man is beset on all sides by the inequities of the world and the
```

```
file ROS_MASTER_URI_receiver.cpp
    #include <ros/ros.h> #include <common_utilities/UDPSocket.hpp>
```

Functions

```
int main (int argc, char *argv[])
```

Variables

```
const char* key= "The path of the righteous man is beset on all sides by the inequities of the world and the
```

```
file rviz_visualization.cpp
    #include "common_utilities/rviz_visualization.hpp"
```

```
file UDPSocket.cpp
    #include <ifaddrs.h> #include "common_utilities/UDPSocket.hpp"
```

Functions

```
uint64_t pack754 (long double f, unsigned bits, unsigned expbits)
long double unpack754 (uint64_t i, unsigned bits, unsigned expbits)
```

```
file cable.hpp
    #include <Eigen/Core> #include <Eigen/Dense> #include <vector>
```

```

file cardsflow_command_interface.hpp
    #include <cassert>#include <string>#include <hardware_interface/internal/hardware_resource_manager.h>#include
    "kindyn/controller/cardsflow_state_interface.hpp"#include <Eigen/Core>#include <Eigen/Dense>

file cardsflow_state_interface.hpp
    #include <hardware_interface/internal/hardware_resource_manager.h>#include <cassert>#include
    <string>#include <Eigen/Core>#include <Eigen/Dense>

file EigenExtension.hpp
    #include <iostream>#include <Eigen/Dense>#include <Eigen/Geometry>#include <Eigen/SVD>

file robot.hpp
    #include <ros/ros.h>#include <Eigen/Core>#include <Eigen/Dense>#include <iDyn-
    Tree/Model/FreeFloatingState.h>#include <iDynTree/KinDynComputations.h>#include <iDyn-
    Tree/ModelIO/ModelLoader.h>#include <iDynTree/Core/EigenHelpers.h>#include <iDyn-
    Tree/InverseKinematics.h>#include "tinyxml.h"#include "kindyn/cable.hpp"#include "kin-
    dyn/EigenExtension.hpp"#include "kindyn/controller/cardsflow_state_interface.hpp"#include "kin-
    dyn/controller/cardsflow_command_interface.hpp"#include <actionlib/server/simple_action_server.h>#include
    <geometry_msgs/PoseStamped.h>#include <geometry_msgs/Vector3.h>#include
    <std_msgs/Float32.h>#include <sensor_msgs/JointState.h>#include <robo_simulation_msgs/Tendon.h>#include
    <robo_simulation_msgs/ControllerType.h>#include <robo_simulation_msgs/JointState.h>#include
    <robo_middleware_msgs/ForwardKinematics.h>#include <robo_middleware_msgs/InverseKinematics.h>#include
    <robo_middleware_msgs/InverseKinematicsMultipleFrames.h>#include <robo_middleware_msgs/MotorCommand.h>#include
    <robo_middleware_msgs/MotorStatus.h>#include <robo_control_msgs/MoveEndEffectorAction.h>#include
    <tf/tf.h>#include <tf/transform_broadcaster.h>#include <tf/transform_listener.h>#include
    <tf_conversions/tf_eigen.h>#include <eigen_conversions/eigen_msg.h>#include
    <qpoASES.hpp>#include <controller_manager/controller_manager.h>#include <con-
    troller_manager_msgs/LoadController.h>#include <hardware_interface/joint_state_interface.h>#include
    <hardware_interface/joint_command_interface.h>#include <hardware_interface/robot_hw.h>#include
    <boost/numeric/odeint.hpp>#include <common_utilities/rviz_visualization.hpp>#include <visualiza-
    tion_msgs/InteractiveMarkerFeedback.h>#include <thread>

```

Typedefs

```
typedef boost::shared_ptr<cardsflow::kindyn::Robot> RobotPtr
```

```

file vrpuppet.hpp
    #include <ros/ros.h>#include <Eigen/Core>#include <Eigen/Dense>#include <iDyn-
    Tree/Model/FreeFloatingState.h>#include <iDynTree/KinDynComputations.h>#include <iDyn-
    Tree/ModelIO/ModelLoader.h>#include <iDynTree/Core/EigenHelpers.h>#include <iDyn-
    Tree/InverseKinematics.h>#include "tinyxml.h"#include "kindyn/cable.hpp"#include "kin-
    dyn/EigenExtension.hpp"#include "kindyn/controller/cardsflow_state_interface.hpp"#include "kin-
    dyn/controller/cardsflow_command_interface.hpp"#include <actionlib/server/simple_action_server.h>#include
    <geometry_msgs/PoseStamped.h>#include <geometry_msgs/Vector3.h>#include
    <std_msgs/Float32.h>#include <sensor_msgs/JointState.h>#include <robo_simulation_msgs/Tendon.h>#include
    <robo_simulation_msgs/ControllerType.h>#include <robo_simulation_msgs/JointState.h>#include
    <robo_middleware_msgs/ForwardKinematics.h>#include <robo_middleware_msgs/InverseKinematics.h>#include
    <robo_middleware_msgs/InverseKinematicsMultipleFrames.h>#include <robo_middleware_msgs/MotorCommand.h>#include
    <robo_middleware_msgs/MotorStatus.h>#include <robo_control_msgs/MoveEndEffectorAction.h>#include
    <tf/tf.h>#include <tf/transform_broadcaster.h>#include <tf/transform_listener.h>#include
    <tf_conversions/tf_eigen.h>#include <eigen_conversions/eigen_msg.h>#include
    <qpoASES.hpp>#include <controller_manager/controller_manager.h>#include <con-
    troller_manager_msgs/LoadController.h>#include <hardware_interface/joint_state_interface.h>#include
    <hardware_interface/joint_command_interface.h>#include <hardware_interface/robot_hw.h>#include

```

```
<boost/numeric/odeint.hpp>#include    <common_utilities/rviz_visualization.hpp>#include    <visualization_msgs/InteractiveMarkerFeedback.h>#include <thread>
```

Typedefs

```
typedef boost::shared_ptr<cardsflow::vrpuppet::Robot> RobotPtr
```

```
file cableLengthController.cpp
```

```
#include    <type_traits>#include    <controller_interface/controller.h>#include    <hardware_interface/joint_command_interface.h>#include    <pluginlib/class_list_macros.h>#include    "kindyn/robot.hpp"#include    "kindyn/controller/cardsflow_state_interface.hpp"#include    <roboym_simulation_msgs/ControllerType.h>#include    <std_msgs/Float32.h>#include    <roboym_control_msgs/SetControllerParameters.h>
```

Functions

```
PLUGINLIB_EXPORT_CLASS (CableLengthController, controller_interface::ControllerBase)
```

```
file cableLengthVelocityController.cpp
```

```
#include    <type_traits>#include    <controller_interface/controller.h>#include    <hardware_interface/joint_command_interface.h>#include    <pluginlib/class_list_macros.h>#include    "kindyn/robot.hpp"#include    "kindyn/controller/cardsflow_state_interface.hpp"#include    <roboym_simulation_msgs/ControllerType.h>#include    <std_msgs/Float32.h>#include    <roboym_control_msgs/SetControllerParameters.h>
```

Functions

```
PLUGINLIB_EXPORT_CLASS (CableLengthVelocityController, controller_interface::ControllerBase)
```

```
file cardsflow_command_interface.cpp
```

```
#include "kindyn/controller/cardsflow_command_interface.hpp"
```

```
file cardsflow_state_interface.cpp
```

```
#include "kindyn/controller/cardsflow_state_interface.hpp"
```

```
file forcePositionController.cpp
```

```
#include    <type_traits>#include    <controller_interface/controller.h>#include    <hardware_interface/joint_command_interface.h>#include    <pluginlib/class_list_macros.h>#include    "kindyn/robot.hpp"#include    "kindyn/controller/cardsflow_state_interface.hpp"#include    <roboym_simulation_msgs/ControllerType.h>#include    <std_msgs/Float32.h>#include    <roboym_control_msgs/SetControllerParameters.h>
```

Functions

```
PLUGINLIB_EXPORT_CLASS (ForcePositionController, controller_interface::ControllerBase)
```

```
file torquePositionController.cpp
```

```
#include    <type_traits>#include    <controller_interface/controller.h>#include    <hardware_interface/joint_command_interface.h>#include    <pluginlib/class_list_macros.h>#include    "kindyn/robot.hpp"#include    "kindyn/controller/cardsflow_command_interface.hpp"#include    <roboym_simulation_msgs/ControllerType.h>#include    <std_msgs/Float32.h>#include    <roboym_control_msgs/SetControllerParameters.h>
```

Functions

PLUGINLIB_EXPORT_CLASS (*TorquePositionController*, controller_interface::ControllerBase)

```
file EigenExtension.cpp
    #include "kindyn/EigenExtension.hpp"

file pedal_simulation.py

file pedal_simulation_interpolation_cubic_derivative.py

file pedal_simulation_interpolation_linear_trajectory_points.py

file pedal_simulation_left_leg_experimental.py

file pedal_simulation_left_leg_fk_pos.py

file finals_simulation_pedaling.py

file joint_angle_velocity_factor_test.py

file velocity_test.py

file capture_pedal_trajectory_left_leg_only.py

file capture_pedal_trajectory.py

file README.md

file robot.cpp
    #include "kindyn/robot.hpp"

file msj_platform.cpp
    #include "kindyn/robot.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h> #include
    <roboy_simulation_msgs/GymStep.h> #include <roboy_simulation_msgs/GymReset.h> #include <com-
    mon_utilities/CommonDefinitions.h>
```

Defines

NUMBER_OF_MOTORS

SPINDLERADIUS

msjMeterPerEncoderTick (encoderTicks)

msjEncoderTicksPerMeter (meter)

Functions

void **update** (controller_manager::ControllerManager *cm)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- cm: pointer to the controller manager

int **main** (int argc, char *argv[])

```
file rickshaw_pedaling.cpp
    #include "kindyn/robot.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h> #include
    <roboy_middleware_msgs/ControlMode.h> #include <common_utilities/CommonDefinitions.h> #include
    <roboy_control_msgs/SetControllerParameters.h>
```


Functions

void **update** (controller_manager::ControllerManager *cm)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- cm: pointer to the controller manager

int **main** (int argc, char *argv[])

file **rikshaw.cpp**

```
#include "kindyn/robot.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h> #include
<roboy_middleware_msgs/ControlMode.h> #include <common_utilities/CommonDefinitions.h> #include
<roboy_control_msgs/SetControllerParameters.h>
```

Functions

void **update** (controller_manager::ControllerManager *cm)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- cm: pointer to the controller manager

int **main** (int argc, char *argv[])

file **rikshaw_new_hands.cpp**

```
#include "kindyn/robot.hpp" #include <thread> #include <std_msgs/Float32.h> #include
<roboy_middleware_msgs/MotorCommand.h> #include <roboy_middleware_msgs/MotorConfig.h> #include
<roboy_middleware_msgs/MotorStatus.h> #include <roboy_middleware_msgs/ControlMode.h> #include
<common_utilities/CommonDefinitions.h> #include <roboy_control_msgs/SetControllerParameters.h>
```

Functions

void **update** (controller_manager::ControllerManager *cm)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- cm: pointer to the controller manager

int **main** (int argc, char *argv[])

file **robot_configuration_estimator.cpp**

```
#include <Eigen/Core> #include <Eigen/Dense> #include <unsupported/Eigen/NonLinearOptimization> #include
<unsupported/Eigen/NumericalDiff> #include <iostream> #include "kindyn/robot.hpp" #include
<std_msgs/Float32.h> #include <controller_manager_msgs/LoadController.h> #include <con-
troller_manager_msgs/SwitchController.h> #include <tf/transform_broadcaster.h>
```

Defines

NUMBER_OF_MOTORS

SPINDLERADIUS

msjMeterPerEncoderTick (encoderTicks)

msjEncoderTicksPerMeter (meter)

Functions

void **update** (controller_manager::ControllerManager *cm)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- cm: pointer to the controller manager

int **main** ()

file **roboy2.cpp**

```
#include "kindyn/vrpuppet.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h> #include
<roboy_middleware_msgs/ControlMode.h> #include <roboy_middleware_msgs/MotorConfigService.h> #include
<common_utilities/CommonDefinitions.h> #include <roboy_control_msgs/SetControllerParameters.h> #include
<std_srvs/Empty.h>
```

Functions

int **main** (int argc, char *argv[])

file **roboy_arcade_maschine.cpp**

```
#include "kindyn/robot.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h>
```

Functions

void **update** (controller_manager::ControllerManager *cm)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- cm: pointer to the controller manager

int **main** (int argc, char *argv[])

file **roboy_head.cpp**

```
#include "kindyn/robot.hpp" #include <thread> #include <std_msgs/Float32.h> #include
<roboy_middleware_msgs/MotorCommand.h> #include <roboy_middleware_msgs/MotorConfig.h> #include
<roboy_middleware_msgs/MotorConfigService.h> #include <roboy_middleware_msgs/MotorStatus.h> #include
<roboy_middleware_msgs/ControlMode.h> #include <common_utilities/CommonDefinitions.h> #include
<roboy_control_msgs/SetControllerParameters.h>
```

Functions

void **update** (controller_manager::ControllerManager *cm)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- cm: pointer to the controller manager

int **main** (int argc, char *argv[])

file **roboy_icecream.cpp**

```
#include "kindyn/vrpuppet.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h> #include
<roboy_middleware_msgs/ControlMode.h> #include <roboy_middleware_msgs/MotorConfigService.h> #include
<common_utilities/CommonDefinitions.h> #include <roboy_control_msgs/SetControllerParameters.h> #include
<std_srvs/Empty.h>
```

Functions

int **main** (int argc, char *argv[])

file **roboy_upper_body.cpp**

```
#include "kindyn/robot.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h> #include
<roboy_middleware_msgs/ControlMode.h> #include <common_utilities/CommonDefinitions.h> #include
<roboy_control_msgs/SetControllerParameters.h>
```

Functions

void **update** (controller_manager::ControllerManager *cm)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- cm: pointer to the controller manager

int **main** (int argc, char *argv[])

file **roboy_xylophone.cpp**

```
#include "kindyn/robot.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h> #include
<common_utilities/CommonDefinitions.h>
```

Functions

void **update** (controller_manager::ControllerManager *cm)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- cm: pointer to the controller manager

int **main** (int argc, char *argv[])

file **shoulder_testbed.cpp**

```
#include "kindyn/robot.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h> #include <common_utilities/CommonDefinitions.h>
```

Defines

NUMBER_OF_MOTORS

Functions

void **update** (controller_manager::ControllerManager **cm*)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- *cm*: pointer to the controller manager

int **main** (int *argc*, char **argv*[])

file **test_robot.cpp**

```
#include "kindyn/robot.hpp" #include <thread>
```

Functions

void **update** (controller_manager::ControllerManager **cm*)

int **main** (int *argc*, char **argv*[])

file **theClaw.cpp**

```
#include "kindyn/robot.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h>
```

Defines

SPINDLERADIUS

FS5103R_MAX_SPEED

FS5103R_FULL_SPEED_BACKWARDS

FS5103R_STOP

FS5103R_FULL_SPEED_FORWARDS

Functions

void **update** (controller_manager::ControllerManager **cm*)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- *cm*: pointer to the controller manager

int **main** (int *argc*, char **argv*[])

file **VRpuppet.cpp**

```
#include "kindyn/robot.hpp" #include <thread> #include <roboy_middleware_msgs/MotorCommand.h> #include
<roboy_middleware_msgs/ControlMode.h> #include <common_utilities/CommonDefinitions.h> #include
<roboy_control_msgs/SetControllerParameters.h>
```

Functions

void **update** (controller_manager::ControllerManager *cm)
controller manager update thread.

Here you can define how fast your controllers should run

Parameters

- cm: pointer to the controller manager

```
int main (int argc, char *argv[])
```

file **vrpuppet.cpp**

```
#include "kindyn/vrpuppet.hpp"
```

file **yatr.cpp**

```
#include "kindyn/robot.hpp" #include <thread>
```

Functions

void **update** (controller_manager::ControllerManager *cm)

```
int main (int argc, char *argv[])
```

file **steering_simulation_old_hand.py**

file **steering_simulation.py**

file **steering_response_test.py**

file **steering_trajectory_following_test.py**

file **compute_steering_path.py**

file **steering_capture_multiple_trajectories.py**

file **steering_capture_trajectory.py**

file **steering_interpolate_and_print.py**

file **steering_interpolate_multiple_trajectories_and_print.py**

file **new_hand_steering_capture_trajectory.py**

file **interpolation.py**

file **kp_kp.py**

file **roboython.py**

file **state_machine.py**

file **test_idyntree.cpp**

```
#include <ros/ros.h> #include <cstdlib> #include <Eigen/Core> #include <iDyn-
Tree/Model/FreeFloatingState.h> #include <iDynTree/KinDynComputations.h> #include <iDyn-
Tree/ModelIO/ModelLoader.h> #include <iDynTree/Core/EigenHelpers.h> #include <iDyn-
Tree/InverseKinematics.h>
```

Functions

```
int main (int argc, char *argv[])
```

page `md_home_docs_checkouts_readthedocs.org_user_builds_cardsflow_checkouts_master_kindyn_s`

This folder contains the control files for CARDSFlow robots.

Doxygen documentation is found in `documentation/html/index.html`.

Before the Python scripts related to the Rickshaw can be run, the Rickshaw robot must be launched. This can at the time of writing be done with: `\roslaunch kindyn robot.launch robot_name:=rikshaw start_controllers:='joint_hip_left joint_hip_right joint_wheel_right joint_wheel_back joint_pedal spine_joint joint_wheel_left joint_front joint_pedal_right joint_pedal_left elbow_right_rot1 joint_foot_left joint_knee_right joint_knee_left joint_foot_right left_shoulder_axis0 left_shoulder_axis1 left_shoulder_axis2 elbow_left_rot1 elbow_left_rot0 left_wrist_0 left_wrist_1 right_shoulder_axis0 right_shoulder_axis2 right_shoulder_axis1 elbow_right_rot0 right_wrist_0 right_wrist_1 head_axis0 head_axis1 head_axis2'`

```
#roslaunch kindyn robot.launch robot_name:=rikshaw start_controllers:='spine_joint sphere_head_axis0
sphere_head_axis1 sphere_head_axis2 joint_hip_left joint_knee_left joint_foot_left joint_hip_right
joint_knee_right joint_foot_right left_shoulder_axis0 left_shoulder_axis1 left_shoulder_axis2 el-
bow_left wrist_left_sphere_axis0 wrist_left_sphere_axis1 wrist_left_sphere_axis2 right_shoulder_axis0
right_shoulder_axis1 right_shoulder_axis2 elbow_right wrist_right_sphere_axis0 wrist_right_sphere_axis1
wrist_right_sphere_axis2 little_right_joint0 ring_right_joint0 middle_right_joint0 index_right_joint0
thumb_right_joint0 little_right_joint1 ring_right_joint1 middle_right_joint1 index_right_joint1
thumb_right_joint1 little_right_joint2 ring_right_joint2 middle_right_joint2 index_right_joint2
thumb_right_joint2 little_left_joint0 ring_left_joint0 middle_left_joint0 index_left_joint0 thumb_left_joint0
little_left_joint1 ring_left_joint1 middle_left_joint1 index_left_joint1 thumb_left_joint1 little_left_joint2
ring_left_joint2 middle_left_joint2 index_left_joint2 thumb_left_joint2 joint_wheel_left joint_wheel_right
joint_front joint_wheel_back joint_pedal joint_pedal_right joint_pedal_left'
```

```
#roslaunch kindyn robot.launch robot_name:=rikshaw start_controllers:='joint_hip_left joint_knee_left
joint_foot_left joint_hip_right joint_knee_right joint_foot_right left_shoulder_axis0 left_shoulder_axis1
left_shoulder_axis2 elbow_left wrist_left_sphere_axis0 right_shoulder_axis0 right_shoulder_axis1
right_shoulder_axis2 elbow_right wrist_right_sphere_axis0 joint_front joint_pedal joint_pedal_right
joint_pedal_left'
```

Below is a content description of each subfolder:

controller \ This folder contains myo-muscle control files.

images \ This folder contains graphs and images relevant to steering and pedaling control of Roboy.

pedaling \ This folder contains files relevant to Roboy pedaling the rickshaw.

robots \ This folder contains the CARDSFlow implementation of each robot.

steering \ This folder contains files relevant to Roboy steering the rickshaw.

utilities \ This folder contains files used in the Rickshaw project.

Todo: \ Add usage information? \ Add description of files inside folders \ Add prerequisites \ Add installation guide

dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/cardsflow_rv

dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/cardsflow_rv

dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/common_utili

dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/common_utili
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/inclu
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/c
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/p
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/s
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/p
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/s
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/common_utili
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/inclu
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/cardsflow_rv
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/inclu
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/p
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/r
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/common_utili
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/cardsflow_rv
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/s
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/p
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/s
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/p
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/s
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/p
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/s
dir /home/docs/checkouts/readthedocs.org/user_builds/cardsflow/checkouts/master/kindyn/src/u

INDEX

B

BROADCAST_PORT (*C macro*), 140
 BYTE_TO_BINARY (*C macro*), 140
 BYTE_TO_BINARY_PATTERN (*C macro*), 140

C

CableLengthController (*C++ class*), 27
 CableLengthController::CableLengthController
 (*C++ function*), 27
 CableLengthController::controller_parameter_srv
 (*C++ member*), 28
 CableLengthController::controller_state
 (*C++ member*), 28
 CableLengthController::init (*C++ function*),
 27
 CableLengthController::joint (*C++ mem-*
 ber), 28
 CableLengthController::joint_command
 (*C++ member*), 28
 CableLengthController::joint_index (*C++*
 member), 28
 CableLengthController::joint_name (*C++*
 member), 28
 CableLengthController::JointPositionCommand
 (*C++ function*), 28
 CableLengthController::Kd (*C++ member*), 28
 CableLengthController::Kp (*C++ member*), 28
 CableLengthController::last_update (*C++*
 member), 28
 CableLengthController::nh (*C++ member*), 28
 CableLengthController::p_error_last
 (*C++ member*), 28
 CableLengthController::setControllerParameters
 (*C++ function*), 28
 CableLengthController::spinner (*C++ mem-*
 ber), 28
 CableLengthController::starting (*C++*
 function), 27
 CableLengthController::stopping (*C++*
 function), 28
 CableLengthController::update (*C++ func-*
 tion), 27
 CableLengthVelocityController (*C++ class*),
 28
 CableLengthVelocityController::CableLengthVelocity
 (*C++ function*), 29
 CableLengthVelocityController::controller_parameter
 (*C++ member*), 30
 CableLengthVelocityController::controller_state
 (*C++ member*), 30
 CableLengthVelocityController::init
 (*C++ function*), 29
 CableLengthVelocityController::joint
 (*C++ member*), 30
 CableLengthVelocityController::joint_command
 (*C++ member*), 30
 CableLengthVelocityController::joint_index
 (*C++ member*), 30
 CableLengthVelocityController::joint_name
 (*C++ member*), 30
 CableLengthVelocityController::JointVelocityCommand
 (*C++ function*), 29
 CableLengthVelocityController::Kd (*C++*
 member), 30
 CableLengthVelocityController::Kp (*C++*
 member), 30
 CableLengthVelocityController::last_update
 (*C++ member*), 30
 CableLengthVelocityController::nh (*C++*
 member), 30
 CableLengthVelocityController::p_error_last
 (*C++ member*), 30
 CableLengthVelocityController::setControllerParameter
 (*C++ function*), 29
 CableLengthVelocityController::spinner
 (*C++ member*), 30
 CableLengthVelocityController::starting
 (*C++ function*), 29
 CableLengthVelocityController::stopping
 (*C++ function*), 29
 CableLengthVelocityController::update
 (*C++ function*), 29
 capture_pedal_trajectory (*built-in class*), 80
 capture_pedal_trajectory.getPedalPositions()

<i>(built-in function)</i> , 80	cardsflow::kindyn::Cable::name (C++ member), 27
capture_pedal_trajectory.getPositionLeftFoot ()	<i>(built-in function)</i> , 80
<i>(built-in function)</i> , 80	cardsflow::kindyn::Cable::viaPoints (C++ member), 27
capture_pedal_trajectory.getPositionRightFoot ()	<i>(built-in function)</i> , 80
<i>(built-in function)</i> , 80	cardsflow::kindyn::Robot (C++ class), 54
capture_pedal_trajectory.inverse_kinematics ()	cardsflow::kindyn::Robot::A (C++ member), 57
<i>(built-in function)</i> , 80	cardsflow::kindyn::Robot::baseVel (C++ member), 55
capture_pedal_trajectory.inverse_kinematics ()	<i>(built-in function)</i> , 80
<i>(built-in function)</i> , 80	cardsflow::kindyn::Robot::bias (C++ member), 56
capture_pedal_trajectory.jointStateCallback ()	<i>(built-in function)</i> , 80
<i>(built-in function)</i> , 80	cardsflow::kindyn::Robot::cable_forces (C++ member), 55
capture_pedal_trajectory.main ()	cardsflow::kindyn::Robot::cable_length_controller (C++ member), 56
<i>(built-in function)</i> , 80	<i>(built-in function)</i> , 80
capture_pedal_trajectory.plotEverything ()	cardsflow::kindyn::Robot::cables (C++ member), 56
<i>(built-in function)</i> , 80	<i>(built-in function)</i> , 80
capture_pedal_trajectory.plotPedalTrajectory ()	cardsflow::kindyn::Robot::cardsflow_command_interface (C++ member), 57
<i>(built-in function)</i> , 80	<i>(built-in function)</i> , 80
capture_pedal_trajectory.setJointControllerParameters ()	cardsflow::kindyn::Robot::cardsflow_state_interface (C++ member), 57
<i>(built-in function)</i> , 80	<i>(built-in function)</i> , 81
capture_pedal_trajectory_left_leg_only	cardsflow::kindyn::Robot::CG (C++ member), 55
<i>(built-in class)</i> , 81	<i>(built-in function)</i> , 82
capture_pedal_trajectory_left_leg_only.gearDistance ()	cardsflow::kindyn::Robot::controller_type (C++ member), 57
<i>(built-in function)</i> , 81	<i>(built-in function)</i> , 81
capture_pedal_trajectory_left_leg_only.gearPedalPos ()	cardsflow::kindyn::Robot::controller_type_sub (C++ member), 59
<i>(built-in function)</i> , 82	<i>(built-in function)</i> , 82
capture_pedal_trajectory_left_leg_only.gearPosLeftFoot ()	cardsflow::kindyn::Robot::endeffector_dof_offset (C++ member), 54
<i>(built-in function)</i> , 81	<i>(built-in function)</i> , 82
capture_pedal_trajectory_left_leg_only.gearPosRightFoot ()	cardsflow::kindyn::Robot::endeffector_index (C++ member), 54
<i>(built-in function)</i> , 81	<i>(built-in function)</i> , 82
capture_pedal_trajectory_left_leg_only.inverse_kinematics ()	cardsflow::kindyn::Robot::endeffector_number_of_dof (C++ member), 54
<i>(built-in function)</i> , 82	<i>(built-in function)</i> , 82
capture_pedal_trajectory_left_leg_only.inverse_kinematics ()	cardsflow::kindyn::Robot::endeffectors (C++ member), 54
<i>(built-in function)</i> , 82	<i>(built-in function)</i> , 82
capture_pedal_trajectory_left_leg_only.jointStateCallback ()	cardsflow::kindyn::Robot::external_robot_target (C++ member), 57
<i>(built-in function)</i> , 82	<i>(built-in function)</i> , 82
capture_pedal_trajectory_left_leg_only.main ()	cardsflow::kindyn::Robot::f_max (C++ member), 57
<i>(built-in function)</i> , 82	<i>(built-in function)</i> , 82
capture_pedal_trajectory_left_leg_only.plotEverything ()	cardsflow::kindyn::Robot::f_min (C++ member), 57
<i>(built-in function)</i> , 82	<i>(built-in function)</i> , 82
capture_pedal_trajectory_left_leg_only.plotPedalTrajectory ()	cardsflow::kindyn::Robot::first_time_solving (C++ member), 57
<i>(built-in function)</i> , 82	<i>(built-in function)</i> , 82
capture_pedal_trajectory_left_leg_only.searchForControlParameters ()	cardsflow::kindyn::Robot::first_update (C++ member), 57
<i>(built-in function)</i> , 82	<i>(built-in function)</i> , 82
CARDSflow (C++ type), 83	cardsflow::kindyn::Robot::fk_srv (C++ member), 60
cardsflow (C++ type), 83	cardsflow::kindyn::Robot::floating_base_sub (C++ member), 60
CARDSflow::cable_length_controller (C++ enumerator), 83	cardsflow::kindyn::Robot::fmt (C++ member), 57
CARDSflow::ControllerType (C++ enum), 83	cardsflow::kindyn::Robot::FOpt (C++ member), 57
CARDSflow::force_position_controller (C++ enumerator), 83	
cardsflow::kindyn (C++ type), 83	
cardsflow::kindyn::Cable (C++ class), 27	

ber), 57
cardsflow::kindyn::Robot::force_position_cardsflow::kindyn::Robot::joint_state_sub
(C++ member), 56 (C++ member), 59
cardsflow::kindyn::Robot::frame_transform_cardsflow::kindyn::Robot::joint_state_target_pub
(C++ member), 55 (C++ member), 59
cardsflow::kindyn::Robot::g (C++ member), cardsflow::kindyn::Robot::kinDynComp
57 (C++ member), 60
cardsflow::kindyn::Robot::gravity (C++ cardsflow::kindyn::Robot::kinDynCompTarget
member), 55 (C++ member), 60
cardsflow::kindyn::Robot::H (C++ member), cardsflow::kindyn::Robot::L (C++ member),
57 56
cardsflow::kindyn::Robot::iDynTreeRobotState_cardsflow::kindyn::Robot::l (C++ member),
(C++ class), 41 55
cardsflow::kindyn::Robot::iDynTreeRobotState_base_cardsflow::kindyn::Robot::l_int (C++
(C++ member), 41 member), 55
cardsflow::kindyn::Robot::iDynTreeRobotState_flow_cardsflow::kindyn::Robot::L_t (C++ mem-
(C++ member), 41 ber), 56
cardsflow::kindyn::Robot::iDynTreeRobotState_flow_cardsflow::kindyn::Robot::l_target (C++
(C++ member), 41 member), 55
cardsflow::kindyn::Robot::iDynTreeRobotState_flow_cardsflow::kindyn::Robot::last_visualization
(C++ member), 41 (C++ member), 57
cardsflow::kindyn::Robot::iDynTreeRobotState_flow_cardsflow::kindyn::Robot::lb (C++ mem-
(C++ function), 41 ber), 57
cardsflow::kindyn::Robot::iDynTreeRobotState_flow_cardsflow::kindyn::Robot::Ld (C++ mem-
(C++ member), 41 ber), 55
cardsflow::kindyn::Robot::ik (C++ mem- cardsflow::kindyn::Robot::ld (C++ mem-
ber), 60 ber), 56
cardsflow::kindyn::Robot::ik_base_link cardsflow::kindyn::Robot::link_index
(C++ member), 60 (C++ member), 56
cardsflow::kindyn::Robot::ik_models cardsflow::kindyn::Robot::link_names
(C++ member), 60 (C++ member), 56
cardsflow::kindyn::Robot::ik_srv (C++ cardsflow::kindyn::Robot::link_to_link_transform
member), 60 (C++ member), 59
cardsflow::kindyn::Robot::ik_two_frames_cardsflow::kindyn::Robot::link_to_world_transform
(C++ member), 60 (C++ member), 55
cardsflow::kindyn::Robot::integration_time_cardsflow::kindyn::Robot::M (C++ member),
(C++ member), 57 55
cardsflow::kindyn::Robot::interactive_markers_cardsflow::kindyn::Robot::Mass (C++ mem-
(C++ member), 60 ber), 56
cardsflow::kindyn::Robot::joint_axis cardsflow::kindyn::Robot::motor_state
(C++ member), 56 (C++ member), 57
cardsflow::kindyn::Robot::joint_command_cardsflow::kindyn::Robot::moveEndEffector_as
(C++ member), 57 (C++ member), 60
cardsflow::kindyn::Robot::joint_command_pub_cardsflow::kindyn::Robot::nh (C++ mem-
(C++ member), 56 ber), 59
cardsflow::kindyn::Robot::joint_index cardsflow::kindyn::Robot::number_of_cables
(C++ member), 56 (C++ member), 55
cardsflow::kindyn::Robot::joint_names cardsflow::kindyn::Robot::number_of_dofs
(C++ member), 56 (C++ member), 54
cardsflow::kindyn::Robot::joint_state cardsflow::kindyn::Robot::number_of_joints
(C++ member), 57 (C++ member), 54
cardsflow::kindyn::Robot::joint_state_interfaces_cardsflow::kindyn::Robot::number_of_links
(C++ member), 57 (C++ member), 55
cardsflow::kindyn::Robot::joint_state_pub_cardsflow::kindyn::Robot::nWSR (C++ mem-

ber), 57
cardsflow::kindyn::Robot::P (C++ member), 56
cardsflow::kindyn::Robot::q (C++ member), 55
cardsflow::kindyn::Robot::q_max (C++ member), 55
cardsflow::kindyn::Robot::q_min (C++ member), 55
cardsflow::kindyn::Robot::q_target (C++ member), 55
cardsflow::kindyn::Robot::q_target_prev (C++ member), 55
cardsflow::kindyn::Robot::qd (C++ member), 55
cardsflow::kindyn::Robot::qd_target (C++ member), 55
cardsflow::kindyn::Robot::qd_target_prev (C++ member), 55
cardsflow::kindyn::Robot::qdd (C++ member), 55
cardsflow::kindyn::Robot::qdd_force_control (C++ member), 56
cardsflow::kindyn::Robot::qdd_target (C++ member), 55
cardsflow::kindyn::Robot::qdd_target_prev (C++ member), 55
cardsflow::kindyn::Robot::qdd_torque_control (C++ member), 56
cardsflow::kindyn::Robot::qp_print_level (C++ member), 57
cardsflow::kindyn::Robot::qp_solver (C++ member), 57
cardsflow::kindyn::Robot::read (C++ function), 54
cardsflow::kindyn::Robot::robot_state_pub (C++ member), 59
cardsflow::kindyn::Robot::robot_state_target_pub (C++ member), 59
cardsflow::kindyn::Robot::robotstate (C++ member), 60
cardsflow::kindyn::Robot::S (C++ member), 56
cardsflow::kindyn::Robot::segments (C++ member), 56
cardsflow::kindyn::Robot::spinner (C++ member), 59
cardsflow::kindyn::Robot::state_type (C++ type), 56
cardsflow::kindyn::Robot::tendon_state_pub (C++ member), 59
cardsflow::kindyn::Robot::tendon_state_target_pub (C++ member), 59
cardsflow::kindyn::Robot::torque_position_controller (C++ member), 53
cardsflow::kindyn::Robot::torques (C++ member), 55
cardsflow::kindyn::Robot::ub (C++ member), 57
cardsflow::kindyn::Robot::V (C++ member), 56
cardsflow::kindyn::Robot::W (C++ member), 56
cardsflow::kindyn::Robot::world_H_base (C++ member), 55
cardsflow::kindyn::Robot::world_to_link_transform (C++ member), 55
cardsflow::kindyn::Robot::write (C++ function), 54
cardsflow::kindyn::ViaPoint (C++ class), 78
cardsflow::kindyn::ViaPoint::fixed_to_world (C++ member), 79
cardsflow::kindyn::ViaPoint::global_coordinates (C++ member), 79
cardsflow::kindyn::ViaPoint::link_index (C++ member), 79
cardsflow::kindyn::ViaPoint::link_name (C++ member), 79
cardsflow::kindyn::ViaPoint::local_coordinates (C++ member), 79
cardsflow::kindyn::ViaPoint::ViaPoint (C++ function), 78
cardsflow::kindyn::ViaPointPtr (C++ type), 83
CARDSflow::torque_position_controller (C++ enumerator), 83
cardsflow::vrpuppet (C++ type), 83
cardsflow::vrpuppet::Robot (C++ class), 47
cardsflow::vrpuppet::Robot::~~Robot (C++ function), 47
cardsflow::vrpuppet::Robot::A (C++ member), 50
cardsflow::vrpuppet::Robot::b (C++ member), 51
cardsflow::vrpuppet::Robot::baseVel (C++ member), 48
cardsflow::vrpuppet::Robot::bias (C++ member), 50
cardsflow::vrpuppet::Robot::cable_forces (C++ member), 49
cardsflow::vrpuppet::Robot::cable_length_controller (C++ member), 50
cardsflow::vrpuppet::Robot::cables (C++ member), 50
cardsflow::vrpuppet::Robot::cardsflow_command_inter (C++ member), 51
cardsflow::vrpuppet::Robot::cardsflow_joint_states (C++ member), 53

cardsflow::vrpuppet::Robot::cardsflow_state_interface vrpuppet::Robot::iDynTreeRobotState::jo
 (C++ member), 51 (C++ member), 41

cardsflow::vrpuppet::Robot::CG (C++ mem- cardsflow::vrpuppet::Robot::iDynTreeRobotState::jo
 ber), 48 (C++ member), 41

cardsflow::vrpuppet::Robot::controller_type cardsflow::vrpuppet::Robot::iDynTreeRobotState::res
 (C++ member), 50 (C++ function), 41

cardsflow::vrpuppet::Robot::controller_type cardsflow::vrpuppet::Robot::iDynTreeRobotState::wor
 (C++ member), 53 (C++ member), 41

cardsflow::vrpuppet::Robot::endeffector_def defswt vrpuppet::Robot::ik (C++ mem-
 (C++ member), 48 ber), 53

cardsflow::vrpuppet::Robot::endeffector_index cardsflow::vrpuppet::Robot::ik_base_link
 (C++ member), 48 (C++ member), 53

cardsflow::vrpuppet::Robot::endeffector_name cardsflow::vrpuppet::Robot::ik_models
 (C++ member), 48 (C++ member), 53

cardsflow::vrpuppet::Robot::endeffectorscardsflow::vrpuppet::Robot::ik_srv (C++
 (C++ member), 48 member), 53

cardsflow::vrpuppet::Robot::external_robot cardsflow::vrpuppet::Robot::ik_two_frames_srv
 (C++ member), 49 (C++ member), 53

cardsflow::vrpuppet::Robot::f_max (C++ cardsflow::vrpuppet::Robot::init (C++
 member), 50 function), 47

cardsflow::vrpuppet::Robot::f_min (C++ cardsflow::vrpuppet::Robot::initialized
 member), 50 (C++ member), 49

cardsflow::vrpuppet::Robot::first_time_so cardsflow::vrpuppet::Robot::integration_time
 (C++ member), 50 (C++ member), 50

cardsflow::vrpuppet::Robot::first_update cardsflow::vrpuppet::Robot::interactive_marker_sub
 (C++ member), 51 (C++ member), 53

cardsflow::vrpuppet::Robot::fk_srv (C++ cardsflow::vrpuppet::Robot::InteractiveMarkerFeedba
 member), 53 (C++ function), 52

cardsflow::vrpuppet::Robot::floating_base_sub cardsflow::vrpuppet::Robot::InverseKinematicsMultipl
 (C++ member), 53 (C++ function), 52

cardsflow::vrpuppet::Robot::FloatingBase cardsflow::vrpuppet::Robot::InverseKinematicsServic
 (C++ function), 52 (C++ function), 52

cardsflow::vrpuppet::Robot::fmt (C++ cardsflow::vrpuppet::Robot::joint_axis
 member), 51 (C++ member), 50

cardsflow::vrpuppet::Robot::FOpt (C++ cardsflow::vrpuppet::Robot::joint_command_interface
 member), 51 (C++ member), 51

cardsflow::vrpuppet::Robot::force_position cardsflow::vrpuppet::Robot::joint_command_pub
 (C++ member), 50 (C++ member), 50

cardsflow::vrpuppet::Robot::ForwardKinematics cardsflow::vrpuppet::Robot::joint_index
 (C++ function), 51 (C++ member), 50

cardsflow::vrpuppet::Robot::frame_transform cardsflow::vrpuppet::Robot::joint_names
 (C++ member), 48 (C++ member), 50

cardsflow::vrpuppet::Robot::g (C++ mem- cardsflow::vrpuppet::Robot::joint_state
 ber), 50 (C++ member), 50

cardsflow::vrpuppet::Robot::gravity cardsflow::vrpuppet::Robot::joint_state_interface
 (C++ member), 48 (C++ member), 51

cardsflow::vrpuppet::Robot::H (C++ mem- cardsflow::vrpuppet::Robot::joint_state_pub
 ber), 50 (C++ member), 53

cardsflow::vrpuppet::Robot::iDynTreeRobot cardsflow::vrpuppet::Robot::joint_state_sub
 (C++ class), 40 (C++ member), 53

cardsflow::vrpuppet::Robot::iDynTreeRobot cardsflow::vrpuppet::Robot::joint_state_target_pub
 (C++ member), 41 (C++ member), 53

cardsflow::vrpuppet::Robot::iDynTreeRobot cardsflow::vrpuppet::Robot::joint_target_sub
 (C++ member), 41 (C++ member), 53

cardsflow::vrpuppet::Robot::JointState (C++ function), 52	cardsflow::vrpuppet::Robot::P (C++ member), 49
cardsflow::vrpuppet::Robot::kinDynComp (C++ member), 48	cardsflow::vrpuppet::Robot::parseViapoints (C++ function), 51
cardsflow::vrpuppet::Robot::kinDynCompTargets (C++ member), 48	cardsflow::vrpuppet::Robot::q (C++ member), 48
cardsflow::vrpuppet::Robot::L (C++ member), 49	cardsflow::vrpuppet::Robot::q_max (C++ member), 48
cardsflow::vrpuppet::Robot::l (C++ member), 49	cardsflow::vrpuppet::Robot::q_min (C++ member), 48
cardsflow::vrpuppet::Robot::l_int (C++ member), 49	cardsflow::vrpuppet::Robot::q_target (C++ member), 49
cardsflow::vrpuppet::Robot::L_t (C++ member), 49	cardsflow::vrpuppet::Robot::q_target_prev (C++ member), 49
cardsflow::vrpuppet::Robot::l_target (C++ member), 49	cardsflow::vrpuppet::Robot::qd (C++ member), 48
cardsflow::vrpuppet::Robot::last_visualization (C++ member), 51	cardsflow::vrpuppet::Robot::qd_target (C++ member), 49
cardsflow::vrpuppet::Robot::lb (C++ member), 50	cardsflow::vrpuppet::Robot::qd_target_prev (C++ member), 49
cardsflow::vrpuppet::Robot::Ld (C++ member), 49	cardsflow::vrpuppet::Robot::qdd (C++ member), 48
cardsflow::vrpuppet::Robot::ld (C++ member), 49	cardsflow::vrpuppet::Robot::qdd_force_control (C++ member), 50
cardsflow::vrpuppet::Robot::link_index (C++ member), 50	cardsflow::vrpuppet::Robot::qdd_target (C++ member), 49
cardsflow::vrpuppet::Robot::link_names (C++ member), 50	cardsflow::vrpuppet::Robot::qdd_target_prev (C++ member), 49
cardsflow::vrpuppet::Robot::link_to_link_constraint (C++ member), 52	cardsflow::vrpuppet::Robot::qdd_torque_control (C++ member), 50
cardsflow::vrpuppet::Robot::link_to_world_constraint (C++ member), 48	cardsflow::vrpuppet::Robot::qp_print_level (C++ member), 50
cardsflow::vrpuppet::Robot::M (C++ member), 48	cardsflow::vrpuppet::Robot::qp_solver (C++ member), 50
cardsflow::vrpuppet::Robot::Mass (C++ member), 50	cardsflow::vrpuppet::Robot::read (C++ function), 47
cardsflow::vrpuppet::Robot::motor_state (C++ member), 50	cardsflow::vrpuppet::Robot::resolve_function (C++ function), 52
cardsflow::vrpuppet::Robot::MoveEndEffectors (C++ function), 51	cardsflow::vrpuppet::Robot::Robot (C++ function), 47
cardsflow::vrpuppet::Robot::moveEndEffectors (C++ member), 53	cardsflow::vrpuppet::Robot::robot_state_pub (C++ member), 53
cardsflow::vrpuppet::Robot::nh (C++ member), 52	cardsflow::vrpuppet::Robot::robot_state_target_pub (C++ member), 53
cardsflow::vrpuppet::Robot::number_of_cables (C++ member), 48	cardsflow::vrpuppet::Robot::robotstate (C++ member), 53
cardsflow::vrpuppet::Robot::number_of_dof (C++ member), 48	cardsflow::vrpuppet::Robot::S (C++ member), 49
cardsflow::vrpuppet::Robot::number_of_joints (C++ member), 48	cardsflow::vrpuppet::Robot::segments (C++ member), 49
cardsflow::vrpuppet::Robot::number_of_links (C++ member), 48	cardsflow::vrpuppet::Robot::simulated (C++ member), 49
cardsflow::vrpuppet::Robot::nWSR (C++ member), 50	cardsflow::vrpuppet::Robot::spinner (C++ member), 52

cardsflow::vrpuppet::Robot::state_type 34
(C++ type), 49 CardsflowRviz::robot_state_target (C++
cardsflow::vrpuppet::Robot::tendon_state_pub member), 34
(C++ member), 53 CardsflowRviz::RobotState (C++ function), 33
cardsflow::vrpuppet::Robot::tendon_state CardsflowRviz::RobotStateTarget (C++
(C++ member), 53 function), 33
cardsflow::vrpuppet::Robot::torque_position CardsflowRviz::save (C++ function), 32
(C++ member), 50 CardsflowRviz::show_collision (C++ func-
cardsflow::vrpuppet::Robot::torques tion), 32
(C++ member), 49 CardsflowRviz::show_collision_button
cardsflow::vrpuppet::Robot::ub (C++ mem- (C++ member), 35
ber), 51 CardsflowRviz::show_force (C++ function), 32
cardsflow::vrpuppet::Robot::update (C++ CardsflowRviz::show_force_button (C++
function), 47 member), 35
cardsflow::vrpuppet::Robot::V (C++ mem- CardsflowRviz::show_mesh (C++ function), 32
ber), 49 CardsflowRviz::show_mesh_button (C++
cardsflow::vrpuppet::Robot::W (C++ mem- member), 35
ber), 49 CardsflowRviz::show_target (C++ function),
cardsflow::vrpuppet::Robot::world_H_base 32
(C++ member), 48 CardsflowRviz::show_target_button (C++
cardsflow::vrpuppet::Robot::world_to_link_trans member), 35
(C++ member), 48 CardsflowRviz::show_tendon (C++ function),
cardsflow::vrpuppet::Robot::write (C++ 32
function), 48 CardsflowRviz::show_tendon_button (C++
CardsflowRviz (C++ class), 32 member), 35
CardsflowRviz::~~CardsflowRviz (C++ func- CardsflowRviz::show_tendon_length (C++
tion), 32 function), 32
CardsflowRviz::cable_thickness (C++ mem- CardsflowRviz::show_tendon_length_button
ber), 35 (C++ member), 35
CardsflowRviz::CardsflowRviz (C++ func- CardsflowRviz::show_torque (C++ function),
tion), 32 32
CardsflowRviz::joint_axis (C++ member), 34 CardsflowRviz::show_torque_button (C++
CardsflowRviz::joint_axis_target (C++ member), 35
member), 34 CardsflowRviz::spinner (C++ member), 34
CardsflowRviz::joint_origin (C++ member), CardsflowRviz::Tendon (C++ class), 72
34 CardsflowRviz::tendon (C++ member), 34
CardsflowRviz::joint_origin_target (C++ CardsflowRviz::Tendon::force (C++ mem-
member), 34 ber), 73
CardsflowRviz::joint_state (C++ member), CardsflowRviz::Tendon::l (C++ member), 73
34 CardsflowRviz::Tendon::ld (C++ member), 73
CardsflowRviz::joint_state_target (C++ CardsflowRviz::Tendon::viaPoints (C++
member), 34 member), 73
CardsflowRviz::JointState (C++ function), 34 CardsflowRviz::tendon_length_text_size
CardsflowRviz::JointStateTarget (C++ (C++ member), 35
function), 34 CardsflowRviz::tendon_state (C++ member),
CardsflowRviz::load (C++ function), 32 34
CardsflowRviz::mesh_transparency (C++ CardsflowRviz::tendon_state_target (C++
member), 35 member), 34
CardsflowRviz::model_name (C++ member), 35 CardsflowRviz::tendon_target (C++ mem-
CardsflowRviz::nh (C++ member), 34 ber), 34
CardsflowRviz::pose (C++ member), 34 CardsflowRviz::TendonState (C++ function),
CardsflowRviz::pose_target (C++ member), 33
34 CardsflowRviz::TendonStateTarget (C++
CardsflowRviz::robot_state (C++ member), function), 33

CardsflowRviz::tf_broadcaster (C++ member), 34
 CardsflowRviz::tf_listener (C++ member), 34
 CardsflowRviz::torque (C++ member), 34
 CardsflowRviz::torque_target (C++ member), 34
 CardsflowRviz::visualize_collisions (C++ member), 34
 CardsflowRviz::visualize_force (C++ member), 35
 CardsflowRviz::visualize_force_target (C++ member), 35
 CardsflowRviz::visualize_pose (C++ member), 34
 CardsflowRviz::visualize_targets (C++ member), 35
 CardsflowRviz::visualize_tendon (C++ member), 35
 CardsflowRviz::visualize_tendon_length (C++ member), 35
 CardsflowRviz::visualize_tendon_length_target (C++ member), 35
 CardsflowRviz::visualize_tendon_target (C++ member), 35
 CardsflowRviz::visualize_torque (C++ member), 35
 CardsflowRviz::visualize_torque_target (C++ member), 35
 CardsflowRviz::visualizeCollision (C++ function), 32
 CardsflowRviz::visualizeCollisionSignal (C++ function), 33
 CardsflowRviz::visualizePose (C++ function), 32
 CardsflowRviz::visualizePoseSignal (C++ function), 33
 CardsflowRviz::visualizePoseTarget (C++ function), 32
 CardsflowRviz::visualizePoseTargetSignal (C++ function), 33
 CardsflowRviz::visualizeTargetSignal (C++ function), 33
 CardsflowRviz::visualizeTendon (C++ function), 33
 CardsflowRviz::visualizeTendonSignal (C++ function), 33
 CardsflowRviz::visualizeTendonTarget (C++ function), 33
 CardsflowRviz::visualizeTendonTargetSignal (C++ function), 33
 CardsflowRviz::visualizeTorque (C++ function), 33
 CardsflowRviz::visualizeTorqueSignal (C++ function), 33
 CardsflowRviz::visualizeTorqueTarget (C++ function), 33
 CardsflowRviz::visualizeTorqueTargetSignal (C++ function), 33
 Clock (C++ type), 140
 COLOR (C++ class), 37
 COLOR::a (C++ member), 37
 COLOR::b (C++ member), 37
 COLOR::COLOR (C++ function), 37
 COLOR::g (C++ member), 37
 COLOR::r (C++ member), 37
 COLOR::randColor (C++ function), 37
 compute_steering_path (built-in class), 83
 compute_steering_path.computeHandTrajectories () (built-in function), 83
 compute_steering_path.computeSteeringAngles () (built-in function), 83
 compute_steering_path.main () (built-in function), 83
E
 Eigen (C++ type), 84
 EigenExtension (C++ type), 84
 EigenExtension::ComputeRotationMatrix (C++ function), 85
 EigenExtension::ComputeRotationMatrixDeriv (C++ function), 85
 EigenExtension::ComputeRotationMatrixDoubleDeriv (C++ function), 85
 EigenExtension::CubicSplineInterpolate (C++ function), 85
 EigenExtension::GetCubicSplineCoefficients (C++ function), 84
 EigenExtension::GetLinearSplineCoefficients (C++ function), 84
 EigenExtension::GetQuinticSplineCoefficients (C++ function), 84
 EigenExtension::LinearSplineInterpolate (C++ function), 85
 EigenExtension::Pinv (C++ function), 85
 EigenExtension::QuinticSplineInterpolate (C++ function), 85
 EigenExtension::SkewSymmetric (C++ function), 84
 EigenExtension::SkewSymmetric2 (C++ function), 84
 EigenRobotAcceleration (C++ class), 37
 EigenRobotAcceleration::baseAcc (C++ member), 37
 EigenRobotAcceleration::jointAcc (C++ member), 37
 EigenRobotAcceleration::random (C++ function), 37

EigenRobotAcceleration::resize (C++ function), 37
 EigenRobotState (C++ class), 37
 EigenRobotState::baseVel (C++ member), 38
 EigenRobotState::gravity (C++ member), 38
 EigenRobotState::jointPos (C++ member), 38
 EigenRobotState::jointVel (C++ member), 38
 EigenRobotState::random (C++ function), 38
 EigenRobotState::resize (C++ function), 38
 EigenRobotState::world_H_base (C++ member), 38

F

finals_simulation_pedaling (built-in class), 85
 finals_simulation_pedaling.check_output_time_pos (built-in function), 86
 finals_simulation_pedaling.check_pedal_angle_variation (built-in function), 86
 finals_simulation_pedaling.compute_velocity (built-in function), 86
 finals_simulation_pedaling.control_pedaling (built-in function), 87
 finals_simulation_pedaling.evaluate_current_pedaling_angle (built-in function), 86
 finals_simulation_pedaling.get_angle_difference (built-in function), 87
 finals_simulation_pedaling.get_distance (built-in function), 86
 finals_simulation_pedaling.get_joint_angle (built-in function), 86
 finals_simulation_pedaling.get_joint_position (built-in function), 85
 finals_simulation_pedaling.get_joint_velocity (built-in function), 85
 finals_simulation_pedaling.get_position (built-in function), 86
 finals_simulation_pedaling.get_position_time_pos (built-in function), 86
 finals_simulation_pedaling.get_position_time_vel (built-in function), 86
 finals_simulation_pedaling.import_joint_force_position_controller (built-in function), 85
 finals_simulation_pedaling.interpolate_functions (built-in function), 86
 finals_simulation_pedaling.joint_state_callback (built-in function), 85
 finals_simulation_pedaling.main() (built-in function), 88
 finals_simulation_pedaling.plot_measured_force_pos_inputs (built-in function), 85
 finals_simulation_pedaling.publish_velocity (built-in function), 86
 finals_simulation_pedaling.set_joint_controller_parameters (built-in function), 86
 finals_simulation_pedaling.update_velocity (built-in function), 87
 ForcePositionController (C++ class), 38
 ForcePositionController::controller_parameter_srv (C++ member), 39
 ForcePositionController::controller_state (C++ member), 39
 ForcePositionController::ForcePositionController (C++ function), 38
 ForcePositionController::init (C++ function), 38
 ForcePositionController::joint (C++ member), 39
 ForcePositionController::joint_command (C++ member), 39
 ForcePositionController::joint_index (C++ member), 39
 ForcePositionController::joint_name (C++ member), 39
 ForcePositionController::JointPositionCommand (C++ function), 39
 ForcePositionController::Kd (C++ member), 39
 ForcePositionController::Kp (C++ member), 39
 ForcePositionController::nh (C++ member), 39
 ForcePositionController::p_error_prev (C++ member), 39
 ForcePositionController::q_target (C++ member), 39
 ForcePositionController::setControllerParameters (C++ function), 39
 ForcePositionController::spinner (C++ member), 39
 ForcePositionController::starting (C++ function), 38
 ForcePositionController::stopping (C++ function), 39
 ForcePositionController::update (C++ function), 38
 FS5103R_FULL_SPEED_BACKWARDS (C macro), 148
 FS5103R_FULL_SPEED_FORWARDS (C macro), 148
 FS5103R_MAX_SPEED (C macro), 148
 FS5103R_STOP (C macro), 148
 Functor (C++ class), 40
 Functor::Functor (C++ function), 40
 Functor::Inputs (C++ function), 40
 Functor::InputsAtCompileTime (C++ enumerator), 40
 Functor::InputType (C++ type), 40
 Functor::JacobianType (C++ type), 40

Functor::m_inputs (C++ member), 40
 Functor::m_values (C++ member), 40
 Functor::Scalar (C++ type), 40
 Functor::values (C++ function), 40
 Functor::ValuesAtCompileTime (C++ enumerator), 40
 Functor::ValueType (C++ type), 40
 Functor::[anonymous] (C++ enum), 40

H

hardware_interface (C++ type), 90
 hardware_interface::CardsflowCommandInterface (C++ member), 37
 (C++ class), 30
 hardware_interface::CardsflowHandle (C++ class), 30
 hardware_interface::CardsflowHandle::CardsflowHandle (C++ function), 30
 hardware_interface::CardsflowHandle::getJointPositionsCommand (C++ function), 31
 hardware_interface::CardsflowHandle::getJointTorqueCommand (C++ function), 31
 hardware_interface::CardsflowHandle::getJointVelocityCommand (C++ function), 31
 hardware_interface::CardsflowHandle::joint_position_cmd_ (C++ member), 31

hardware_interface::CardsflowHandle::joint_torque_cmd_ (C++ member), 31

hardware_interface::CardsflowHandle::joint_velocity_cmd_ (C++ member), 31

hardware_interface::CardsflowHandle::motor_cmds_ (C++ member), 31

hardware_interface::CardsflowHandle::setJointPositionsCommand (C++ function), 31

hardware_interface::CardsflowHandle::setJointTorqueCommand (C++ function), 31

hardware_interface::CardsflowHandle::setJointVelocityCommand (C++ function), 31

hardware_interface::CardsflowHandle::setMotorCommand (C++ function), 31

hardware_interface::CardsflowStateHandle (C++ class), 35

hardware_interface::CardsflowStateHandle::acc_ (C++ member), 37

hardware_interface::CardsflowStateHandle::CardsflowStateHandle (C++ function), 35

hardware_interface::CardsflowStateHandle::CG_ (C++ member), 37

hardware_interface::CardsflowStateHandle::getAccelerationImportJointTrajectoryRecord() (C++ function), 36

hardware_interface::CardsflowStateHandle::getCG (C++ function), 36

hardware_interface::CardsflowStateHandle::joint_angle_velocity_factor_test (C++ function), 36

hardware_interface::CardsflowStateHandle::getL (C++ function), 36

hardware_interface::CardsflowStateHandle::getM (C++ function), 36

hardware_interface::CardsflowStateHandle::getName (C++ function), 36

hardware_interface::CardsflowStateHandle::getPosition (C++ function), 36

hardware_interface::CardsflowStateHandle::getVelocity (C++ function), 36

hardware_interface::CardsflowStateHandle::joint_in (C++ member), 37

hardware_interface::CardsflowStateHandle::L_ (C++ member), 37

hardware_interface::CardsflowStateHandle::M_ (C++ member), 37

hardware_interface::CardsflowStateHandle::name_ (C++ member), 37

hardware_interface::CardsflowStateHandle::pos_ (C++ member), 37

hardware_interface::CardsflowStateHandle::vel_ (C++ member), 37

hardware_interface::CardsflowStateInterface (C++ class), 37

iDynTreeRobotAcceleration (C++ class), 40

iDynTreeRobotAcceleration::baseAcc (C++ member), 40

iDynTreeRobotAcceleration::jointAcc (C++ member), 40

iDynTreeRobotAcceleration::resize (C++ function), 40

iDynTreeRobotCommand (C++ class), 41

iDynTreeRobotState::baseVel (C++ member), 41

iDynTreeRobotState::gravity (C++ member), 41

iDynTreeRobotState::jointPos (C++ member), 41

iDynTreeRobotState::jointVel (C++ member), 41

iDynTreeRobotState::resize (C++ function), 41

iDynTreeRobotState::world_H_base (C++ member), 41

interpolation (built-in class), 90

interpolation::importJointTrajectoryRecord() (built-in function), 90

joint_angle_velocity_factor_test (built-in class), 91

joint_angle_velocity_factor_test.checkOutMeasureExecutionTime
(built-in function), 91 (C++ function), 42

joint_angle_velocity_factor_test.computeMeasureSetpointTime::start (C++ function),
(built-in function), 91 42

joint_angle_velocity_factor_test.evaluateMeasureExecutionTime::start_point (C++
(built-in function), 92 member), 42

joint_angle_velocity_factor_test.FSM() MeasureExecutionTime::stop (C++ function),
(built-in function), 92 42

joint_angle_velocity_factor_test.get_joint_angle_execution_time::stop_point (C++
(built-in function), 92 member), 42

joint_angle_velocity_factor_test.getDistanceMeasureExecutionTimePtr (C++ type), 140
(built-in function), 91 MotorConfig (C++ class), 42

joint_angle_velocity_factor_test.getJointMotorConfig::coeffs_displacement2force
(built-in function), 91 (C++ member), 43

joint_angle_velocity_factor_test.getJointMotorConfig::coeffs_force2displacement
(built-in function), 91 (C++ member), 43

joint_angle_velocity_factor_test.getPositionMotorConfig::displacement2force (C++
(built-in function), 91 function), 42

joint_angle_velocity_factor_test.getPositionMotorConfig {fileExists (C++ function), 42
(built-in function), 91 MotorConfig::force2displacement (C++
function), 43

joint_angle_velocity_factor_test.getPositionMotorConfig (C++ function), 42
(built-in function), 91 MotorConfig::MotorConfig (C++ function), 42

joint_angle_velocity_factor_test.importJointTrajectoryRedConfig (C++ function), 42
(built-in function), 91 MotorConfig::writeConfig (C++ function), 42

joint_angle_velocity_factor_test.interpolateEncoderTicksPerMeter (C macro), 144, 146
(built-in function), 92 msjMeterPerEncoderTick (C macro), 144, 146

joint_angle_velocity_factor_test.interpolateTrajectory (C++ class), 43
(built-in function), 91 MsjPlatform::external_robot_state (C++
member), 44

joint_angle_velocity_factor_test.jointStateCall (C++ member), 44
(built-in function), 91 MsjPlatform::gym_reset (C++ member), 44

joint_angle_velocity_factor_test.main() MsjPlatform::gym_step (C++ member), 44
(built-in function), 92 MsjPlatform::GymResetService (C++ func-
tion), 43

joint_angle_velocity_factor_test.publish_velocity (C++ function), 42
(built-in function), 92 MsjPlatform::GymStepService (C++ function),
43

joint_angle_velocity_factor_test.setPedalSingleRotationDuration ()
(built-in function), 91 MsjPlatform::Kp (C++ member), 44

joint_angle_velocity_factor_test.setTrajectoryDuration (C++ member), 44
(built-in function), 91 MsjPlatform::limits (C++ member), 44

MsjPlatform::motor_command (C++ member), 44

K

- `kp_kp` (*built-in class*), 94
- `kp_kp.main()` (*built-in function*), 94
- `kp_kp.setJointControllerParameters()` (*built-in function*), 94

M

main (C++ *function*), [141](#), [144–150](#)
 MAXBUFLNGTH (C *macro*), [140](#)
 MeasureExecutionTime (C++ *class*), [41](#)
 MeasureExecutionTime::~MeasureExecutionTime (C++ *function*), [42](#)
 MeasureExecutionTime::log_file (C++ *member*), [42](#)

```

MsjPlatform::MsjPlatform(C++ function), 43
MsjPlatform::nh(C++ member), 44
MsjPlatform::pnpoly(C++ function), 44
MsjPlatform::pose_thread(C++ member), 44
MsjPlatform::randomPose(C++ function), 44
MsjPlatform::read(C++ function), 43, 44
MsjPlatform::sphere_axis0(C++ member), 44
MsjPlatform::sphere_axis1(C++ member), 44
MsjPlatform::sphere_axis2(C++ member), 44
MsjPlatform::write(C++ function), 43, 44

```

N

`new_hand_steering_capture_trajectory`
(*built-in class*), 95

<code>new_hand_steering_capture_trajectory.competeHandAndPublishVelocity()</code> (built-in function), 95	<code>new_hand_steering_capture_trajectory.competeHandAndPublishVelocity()</code> (built-in function), 97
<code>new_hand_steering_capture_trajectory.competeSteeringAnglesSetJointControllerParameters()</code> (built-in function), 95	<code>new_hand_steering_capture_trajectory.competeSteeringAnglesSetJointControllerParameters()</code> (built-in function), 96
<code>new_hand_steering_capture_trajectory.getPedalInfoFromHand()</code> (built-in function), 95	<code>new_hand_steering_capture_trajectory.getPedalInfoFromHand()</code> (built-in function), 98
<code>new_hand_steering_capture_trajectory.getPedalInfoFromRightHand()</code> (built-in function), 95	<code>new_hand_steering_capture_trajectory.getPedalInfoFromRightHand()</code> (built-in class), 100
<code>new_hand_steering_capture_trajectory.inverseKinematicsInterpolationCubicDerivative.ch</code> (built-in function), 95	<code>new_hand_steering_capture_trajectory.inverseKinematicsInterpolationCubicDerivative.ch</code> (built-in function), 101
<code>new_hand_steering_capture_trajectory.joinPedalCallback()</code> (built-in function), 95	<code>new_hand_steering_capture_trajectory.joinPedalCallback()</code> (built-in function), 101
<code>new_hand_steering_capture_trajectory.mainPedalSimulationInterpolationCubicDerivative.get</code> (built-in function), 95	<code>new_hand_steering_capture_trajectory.mainPedalSimulationInterpolationCubicDerivative.get</code> (built-in function), 101
<code>new_hand_steering_capture_trajectory.setPedalControllerParameters()</code> (built-in function), 95	<code>new_hand_steering_capture_trajectory.setPedalControllerParameters()</code> (built-in function), 101
<code>NUMBER_OF_MOTORS (C macro), 144, 146, 148</code>	<code>pedal_simulation_interpolation_cubic_derivative.get</code>

P

pack754 (C++ function), 140, 141	pedal_simulation_interpolation_cubic_derivative.get
pack754_32 (C macro), 140	(built-in function), 101
pack754_64 (C macro), 140	pedal_simulation_interpolation_cubic_derivative.get
pedal_simulation (built-in class), 96	(built-in function), 101
pedal_simulation.check_output_limits()	pedal_simulation_interpolation_cubic_derivative.im
(built-in function), 97	(built-in function), 101
pedal_simulation.compute_velocity()	pedal_simulation_interpolation_cubic_derivative.in
(built-in function), 97	(built-in function), 101
pedal_simulation.control_pedaling()	pedal_simulation_interpolation_cubic_derivative.jo
(built-in function), 98	pedal_simulation_interpolation_cubic_derivative.ma
pedal_simulation.evaluate_current_pedal_angle()	(built-in function), 101
(built-in function), 97	pedal_simulation_interpolation_cubic_derivative.pr
pedal_simulation.get_angle_difference()	(built-in function), 101
(built-in function), 98	pedal_simulation_interpolation_cubic_derivative.se
pedal_simulation.get_distance() (built-in	(built-in function), 101
function), 97	pedal_simulation_interpolation_cubic_derivative.se
pedal_simulation.get_joint_angle()	(built-in function), 101
(built-in function), 97	pedal_simulation_interpolation_cubic_derivative.se
pedal_simulation.get_joint_position()	(built-in function), 101
(built-in function), 96	pedal_simulation_interpolation_cubic_derivative.se
pedal_simulation.get_joint_velocity()	(built-in function), 101
(built-in function), 96	pedal_simulation_interpolation_cubic_derivative.se
pedal_simulation.get_position() (built-in	(built-in function), 101
function), 96	pedal_simulation_interpolation_linear_trajectory_p
pedal_simulation.get_position_left_foot()	(built-in class), 103
(built-in function), 97	pedal_simulation_interpolation_linear_trajectory_p
pedal_simulation.get_position_right_foot()	(built-in function), 103
(built-in function), 97	pedal_simulation_interpolation_linear_trajectory_p
pedal_simulation.import_joint_trajectory_record	(built-in function), 103
(built-in function), 96	pedal_simulation_interpolation_linear_trajectory_p
pedal_simulation.interpolate_functions()	(built-in function), 103
(built-in function), 97	pedal_simulation_interpolation_linear_trajectory_p
pedal_simulation.joint_state_callback()	(built-in function), 103
(built-in function), 96	pedal_simulation_interpolation_linear_trajectory_p
pedal_simulation.main() (built-in function), 98	(built-in function), 103

pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.main() (built-in function), 103
 pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.publish_velocity() (built-in function), 103
 pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.set_joint_controller() (built-in function), 103
 pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.update_velocity() (built-in function), 103
 pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.record() (built-in function), 103
 pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.set_trajectory_limit() (built-in function), 103
 pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.compute_velocity() (built-in function), 103
 pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.set_callback() (built-in function), 103
 pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.set_parameters() (built-in function), 103
 pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.set_single_stage_angle_difference() (built-in function), 103
 pedal_simulation_interpolation_linear_trajectory_simulation_engine_bicycle_experimental.set_trajectory_point_distribution() (built-in function), 103
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.get_joint_angle() (built-in class), 105
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.get_joint_position() (built-in function), 105
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.get_joint_velocity() (built-in function), 106
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.get_position() (built-in function), 106
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.get_position_left() (built-in function), 106
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.get_position_left_difference() (built-in function), 106
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.get_position_right() (built-in function), 105
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.import_joint_trajectory() (built-in function), 106
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.interpolate_function() (built-in function), 105
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.joint_state_callback() (built-in function), 105
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.main() (built-in function), 105
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.publish_velocity() (built-in function), 105
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.set_joint_controller() (built-in function), 105
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.update_velocity() (built-in function), 105
 pedal_simulation_left_leg_experimental.pedal_simulation_left_leg_fk_pos.set_callback() (built-in function), 106
 pedal_simulation_left_leg_experimental.pluginlib_export_class(C++ function), 143, 145
 pedal_simulation_left_leg_experimental.joint_state_callback() (built-in function), 105

Q

qpOASES (C++ type), 115

R

Rickshaw_pedaling (C++ class), 44

Rickshaw_pedaling::endeffector_jointnames
(C++ member), 45Rickshaw_pedaling::endeffectors (C++
member), 45Rickshaw_pedaling::external_robot_state
(C++ member), 45Rickshaw_pedaling::motor_command (C++
member), 45Rickshaw_pedaling::motor_config (C++
member), 45Rickshaw_pedaling::motor_control_mode
(C++ member), 45

Rickshaw_pedaling::nh (C++ member), 45

Rickshaw_pedaling::read (C++ function), 44

Rickshaw_pedaling::Rickshaw_pedaling
(C++ function), 44Rickshaw_pedaling::sphere_left_axis0_params
(C++ member), 45Rickshaw_pedaling::sphere_left_axis1_params
(C++ member), 45Rickshaw_pedaling::sphere_left_axis2_params
(C++ member), 45

Rickshaw_pedaling::write (C++ function), 44

Rikshaw (C++ class), 45

Rikshaw::elbow_left_rot0 (C++ member), 46

Rikshaw::elbow_left_rot1 (C++ member), 46

Rikshaw::elbow_right_rot0 (C++ member), 47

Rikshaw::elbow_right_rot1 (C++ member), 47

Rikshaw::endeffector_jointnames (C++
member), 46

Rikshaw::endeffectors (C++ member), 46

Rikshaw::err_x (C++ member), 47

Rikshaw::error_y (C++ member), 47

Rikshaw::external_robot_state (C++ mem-
ber), 46

Rikshaw::joint_foot_left (C++ member), 46

Rikshaw::joint_foot_right (C++ member), 46

Rikshaw::joint_hip_left (C++ member), 46

Rikshaw::joint_hip_right (C++ member), 46

Rikshaw::joint_knee_left (C++ member), 46

Rikshaw::joint_knee_right (C++ member), 46

Rikshaw::left_shoulder_axis0 (C++ mem-
ber), 46Rikshaw::left_shoulder_axis1 (C++ mem-
ber), 46Rikshaw::left_shoulder_axis2 (C++ mem-
ber), 46

Rikshaw::left_wrist_0 (C++ member), 46

Rikshaw::left_wrist_1 (C++ member), 46

Rikshaw::motor_command (C++ member), 46

Rikshaw::motor_config (C++ member), 46, 47

Rikshaw::motor_control_mode (C++ member),
46

Rikshaw::motor_status (C++ member), 47

Rikshaw::MotorStatus (C++ function), 45

Rikshaw::nh (C++ member), 46

Rikshaw::pitch (C++ member), 47

Rikshaw::pitch_max (C++ member), 47

Rikshaw::pitch_min (C++ member), 47

Rikshaw::read (C++ function), 45

Rikshaw::right_shoulder_axis0 (C++ mem-
ber), 46Rikshaw::right_shoulder_axis1 (C++ mem-
ber), 46Rikshaw::right_shoulder_axis2 (C++ mem-
ber), 47

Rikshaw::right_wrist_0 (C++ member), 47

Rikshaw::right_wrist_1 (C++ member), 47

Rikshaw::Rikshaw (C++ function), 45

Rikshaw::roll (C++ member), 47

Rikshaw::sphere_left_axis0_params (C++
member), 46Rikshaw::sphere_left_axis1_params (C++
member), 46Rikshaw::sphere_left_axis2_params (C++
member), 46

Rikshaw::status_received (C++ member), 47

Rikshaw::write (C++ function), 45, 46

Rikshaw::yaw (C++ member), 47

Rikshaw::yaw_max (C++ member), 47

Rikshaw::yaw_min (C++ member), 47

Robot (C++ class), 53

Robot::~~Robot (C++ function), 54

Robot::controllerType (C++ function), 59

Robot::external_robot_state (C++ member),
54

Robot::FloatingBase (C++ function), 59

Robot::forwardKinematics (C++ function), 54

Robot::ForwardKinematicsService (C++
function), 58

Robot::init (C++ function), 54

Robot::InteractiveMarkerFeedback (C++
function), 58Robot::InverseKinematicsMultipleFramesService
(C++ function), 58Robot::InverseKinematicsService (C++
function), 58

Robot::JointState (C++ function), 58

Robot::JointTarget (C++ function), 52

Robot::MoveEndEffector (C++ function), 58

Robot::nh (C++ member), 54

Robot::parseViapoints (C++ function), 58

Robot::read (C++ function), 53

Robot::resolve_function (C++ function), 59
 Robot::Robot (C++ function), 53, 54
 Robot::update (C++ function), 54
 Robot::update_P (C++ function), 59
 Robot::update_S (C++ function), 59
 Robot::update_V (C++ function), 59
 Robot::write (C++ function), 53
 RobotConfigurationEstimator (C++ class), 60
 RobotConfigurationEstimator::number_of_cables (C++ member), 60
 RobotConfigurationEstimator::number_of_links (C++ member), 60
 RobotConfigurationEstimator::number_of_sensors (C++ member), 60
 RobotConfigurationEstimator::operator () (C++ function), 60
 RobotConfigurationEstimator::robot (C++ member), 60
 RobotConfigurationEstimator::RobotConfigurationEstimatorCoordinates (C++ function), 60
 RobotPtr (C++ type), 142, 143
 Roboy2 (C++ class), 60
 Roboy2::body_parts (C++ member), 61
 Roboy2::bodyPartIDs (C++ member), 61
 Roboy2::displacement (C++ member), 61
 Roboy2::endeffector_jointnames (C++ member), 61
 Roboy2::init_mode (C++ member), 61
 Roboy2::init_pose (C++ member), 61
 Roboy2::init_setpoint (C++ member), 61
 Roboy2::initPose (C++ function), 61
 Roboy2::l_offset (C++ member), 61
 Roboy2::motor_command (C++ member), 61
 Roboy2::motor_config (C++ member), 61
 Roboy2::motor_control_mode (C++ member), 61
 Roboy2::motor_status_received (C++ member), 61
 Roboy2::motor_status_sub (C++ member), 61
 Roboy2::motor_type (C++ member), 61
 Roboy2::MotorStatus (C++ function), 61
 Roboy2::nh (C++ member), 61
 Roboy2::position (C++ member), 61
 Roboy2::read (C++ function), 61
 Roboy2::real_motor_ids (C++ member), 61
 Roboy2::Roboy2 (C++ function), 61
 Roboy2::sim_motor_ids (C++ member), 61
 Roboy2::spinner (C++ member), 61
 Roboy2::use_motor_config (C++ member), 61
 Roboy2::velocity (C++ member), 61
 Roboy2::write (C++ function), 61
 RoboyArcadeMaschine (C++ class), 62
 RoboyArcadeMaschine::external_robot_state (C++ member), 62
 RoboyArcadeMaschine::motor_command (C++ member), 62
 RoboyArcadeMaschine::nh (C++ member), 62
 RoboyArcadeMaschine::read (C++ function), 62
 RoboyArcadeMaschine::RoboyArcadeMaschine (C++ function), 62
 RoboyArcadeMaschine::write (C++ function), 62
 RoboyHead (C++ class), 62
 RoboyHead::endeffector_jointnames (C++ member), 63
 RoboyHead::endeffectors (C++ member), 63
 RoboyHead::err_x (C++ member), 63
 RoboyHead::error_y (C++ member), 63
 RoboyHead::external_robot_state (C++ member), 63
 RoboyHead::face_coordinates (C++ member), 63
 RoboyHead::RoboyHeadCoordinates (C++ function), 62
 RoboyHead::motor_command (C++ member), 63
 RoboyHead::motor_config (C++ member), 63
 RoboyHead::motor_control_mode (C++ member), 63
 RoboyHead::motor_status (C++ member), 63
 RoboyHead::MotorStatus (C++ function), 62
 RoboyHead::nh (C++ member), 63
 RoboyHead::pitch (C++ member), 63
 RoboyHead::pitch_max (C++ member), 63
 RoboyHead::pitch_min (C++ member), 63
 RoboyHead::read (C++ function), 62
 RoboyHead::RoboyHead (C++ function), 62
 RoboyHead::roll (C++ member), 63
 RoboyHead::sphere_head_axis0 (C++ member), 63
 RoboyHead::sphere_head_axis1 (C++ member), 63
 RoboyHead::sphere_head_axis2 (C++ member), 63
 RoboyHead::sphere_left_axis0_params (C++ member), 63
 RoboyHead::sphere_left_axis1_params (C++ member), 63
 RoboyHead::sphere_left_axis2_params (C++ member), 63
 RoboyHead::status_received (C++ member), 63
 RoboyHead::write (C++ function), 62
 RoboyHead::yaw (C++ member), 63
 RoboyHead::yaw_max (C++ member), 63
 RoboyHead::yaw_min (C++ member), 63
 RoboyIcecream (C++ class), 63
 RoboyIcecream::body_parts (C++ member), 64
 RoboyIcecream::bodyPartIDs (C++ member), 64

64
 RoboyIcecream::displacement (C++ member), 64
 RoboyIcecream::endeffector_jointnames (C++ member), 64
 RoboyIcecream::init_mode (C++ member), 64
 RoboyIcecream::init_pose (C++ member), 64
 RoboyIcecream::init_setpoint (C++ member), 64
 RoboyIcecream::initPose (C++ function), 64
 RoboyIcecream::l_offset (C++ member), 64
 RoboyIcecream::motor_command (C++ member), 64
 RoboyIcecream::motor_config (C++ member), 64
 RoboyIcecream::motor_control_mode (C++ member), 64
 RoboyIcecream::motor_status_received (C++ member), 64
 RoboyIcecream::motor_status_sub (C++ member), 64
 RoboyIcecream::motor_type (C++ member), 64
 RoboyIcecream::MotorStatus (C++ function), 64
 RoboyIcecream::nh (C++ member), 64
 RoboyIcecream::position (C++ member), 64
 RoboyIcecream::read (C++ function), 64
 RoboyIcecream::real_motor_ids (C++ member), 64
 RoboyIcecream::RoboyIcecream (C++ function), 64
 RoboyIcecream::sim_motor_ids (C++ member), 64
 RoboyIcecream::spinner (C++ member), 64
 RoboyIcecream::use_motor_config (C++ member), 64
 RoboyIcecream::velocity (C++ member), 64
 RoboyIcecream::write (C++ function), 64
 robaython (built-in class), 115
 robaython.compute_distance() (built-in function), 115
 robaython.compute_leg_length() (built-in function), 115
 robaython.get_pedal_position_x() (built-in function), 115
 robaython.get_pedal_position_y() (built-in function), 115
 RoboyUpperBody (C++ class), 65
 RoboyUpperBody::endeffector_jointnames (C++ member), 65
 RoboyUpperBody::endeffectors (C++ member), 65
 RoboyUpperBody::external_robot_state (C++ member), 65
 RoboyUpperBody::motor_command (C++ member), 65
 RoboyUpperBody::motor_config (C++ member), 65
 RoboyUpperBody::motor_control_mode (C++ member), 65
 RoboyUpperBody::nh (C++ member), 65
 RoboyUpperBody::read (C++ function), 65
 RoboyUpperBody::RoboyUpperBody (C++ function), 65
 RoboyUpperBody::sphere_left_axis0_params (C++ member), 65
 RoboyUpperBody::sphere_left_axis1_params (C++ member), 65
 RoboyUpperBody::sphere_left_axis2_params (C++ member), 65
 RoboyUpperBody::write (C++ function), 65
 RoboyXylophone (C++ class), 65
 RoboyXylophone::external_robot_state (C++ member), 66
 RoboyXylophone::l_offset (C++ member), 66
 RoboyXylophone::motor_command (C++ member), 66
 RoboyXylophone::nh (C++ member), 66
 RoboyXylophone::read (C++ function), 66
 RoboyXylophone::RoboyXylophone (C++ function), 66
 RoboyXylophone::write (C++ function), 66
 rviz_visualization (C++ class), 66
 rviz_visualization::~~rviz_visualization (C++ function), 66
 rviz_visualization::broadcaster (C++ member), 72
 rviz_visualization::clearAll (C++ function), 70
 rviz_visualization::clearMarker (C++ function), 70
 rviz_visualization::convertEigenToGeometry (C++ function), 66
 rviz_visualization::convertGeometryToEigen (C++ function), 66
 rviz_visualization::first (C++ member), 72
 rviz_visualization::getLighthouseTransform (C++ function), 70, 71
 rviz_visualization::getTransform (C++ function), 70, 71
 rviz_visualization::initializeInteractiveMarkerServer (C++ function), 66
 rviz_visualization::interactive_marker_server (C++ member), 72
 rviz_visualization::listener (C++ member), 72
 rviz_visualization::make6DofMarker (C++ function), 66

rviz_visualization::makeBox (C++ function), 66
 rviz_visualization::makeBoxControl (C++ function), 66
 rviz_visualization::marker_array (C++ member), 72
 rviz_visualization::menu_handler (C++ member), 72
 rviz_visualization::nh (C++ member), 72
 rviz_visualization::number_of_markers_to_publish (C++ member), 71
 rviz_visualization::PoseMsgToTF (C++ function), 66
 rviz_visualization::processFeedback (C++ function), 71
 rviz_visualization::publish_as_marker_array (C++ member), 71
 rviz_visualization::publishCube (C++ function), 68
 rviz_visualization::publishCylinder (C++ function), 69
 rviz_visualization::publishMesh (C++ function), 66, 67
 rviz_visualization::publishRay (C++ function), 69
 rviz_visualization::publishSphere (C++ function), 67, 68
 rviz_visualization::publishText (C++ function), 69
 rviz_visualization::publishTransform (C++ function), 71
 rviz_visualization::rviz_visualization (C++ function), 66
 rviz_visualization::visualization_array_publisher (C++ member), 71
 rviz_visualization::visualization_publisher (C++ member), 71
 ShoulderTestbed::write (C++ function), 72
 SPINDLERADIUS (C macro), 144, 146, 148
 state_machine (built-in class), 116
 state_machine.computeVelocitySetpoint() (built-in function), 116
 state_machine.FSM() (built-in function), 116
 state_machine.getDistance() (built-in function), 116
 state_machine.getJointPosition() (built-in function), 116
 state_machine.getJointVelocity() (built-in function), 116
 state_machine.getPedalPosition() (built-in function), 116
 state_machine.importJointPIDParameters() (built-in function), 116
 state_machine.importJointTrajectoryRecord() (built-in function), 116
 state_machine.interpolateTrajectoryPoints() (built-in function), 116
 state_machine.main() (built-in function), 116
 state_machine.setJointVelocity() (built-in function), 116
 state_machine.setPedalSingleRotationDuration() (built-in function), 116
 state_machine.setTrajectoryPointDuration() (built-in function), 116
 std (C++ type), 117
 steering (built-in class), 117
 steering_capture_multiple_trajectories (built-in class), 117
 steering_capture_multiple_trajectories.computeHandTrajectories() (built-in function), 118
 steering_capture_multiple_trajectories.computeSteeringAngles() (built-in function), 118
 steering_capture_multiple_trajectories.getPositionLeftHand() (built-in function), 118
 steering_capture_multiple_trajectories.getPositionRightHand() (built-in function), 118
 steering_capture_multiple_trajectories.inverse_kinematics() (built-in function), 118
 steering_capture_multiple_trajectories.jointStateCallback() (built-in function), 118
 steering_capture_multiple_trajectories.main() (built-in function), 118
 steering_capture_multiple_trajectories.setJointController() (built-in function), 118
 steering_capture_trajectory (built-in class), 119
 steering_capture_trajectory.computeHandTrajectories() (built-in function), 119
 steering_capture_trajectory.computeSteeringAngles() (built-in function), 119
 steering_capture_trajectory.getPositionLeftHand() (built-in function), 119

S

ShoulderTestbed (C++ class), 72
 ShoulderTestbed::external_robot_state (C++ member), 72
 ShoulderTestbed::Kp (C++ member), 72
 ShoulderTestbed::l_offset (C++ member), 72
 ShoulderTestbed::last_update (C++ member), 72
 ShoulderTestbed::motor_command (C++ member), 72
 ShoulderTestbed::nh (C++ member), 72
 ShoulderTestbed::read (C++ function), 72
 ShoulderTestbed::ShoulderTestbed (C++ function), 72
 ShoulderTestbed::winch_radius (C++ member), 72

(built-in function), 119	(built-in function), 128
steering_capture_trajectory.getPositionRightHand(\$simulation.get_joint_position())	(built-in function), 127
(built-in function), 119	(built-in function), 127
steering_capture_trajectory.inverse_kinematics(\$simulation.import_joint_trajectory_record)	(built-in function), 127
(built-in function), 119	(built-in function), 127
steering_capture_trajectory.jointStateCallback(\$simulation.interpolate_joint_angles())	(built-in function), 127
(built-in function), 119	(built-in function), 127
steering_capture_trajectory.main()	steering_simulation.joint_state_callback()
(built-in function), 119	(built-in function), 127
steering_capture_trajectory.setJointController(\$simulation.main())	(built-in function), 129
(built-in function), 119	
steering_interpolate_and_print	(built-in steering_simulation.publish_joint_angle())
class), 121	(built-in function), 128
steering_interpolate_and_print.importJointTrajectoryRecord(\$simulation.regress_joint_positions_from_file)	(built-in function), 127
(built-in function), 121	
steering_interpolate_and_print.interpolateAndInterpolate(\$simulation.set_joint_controller_parameters)	(built-in function), 128
(built-in function), 121	
steering_interpolate_and_print.main()	steering_simulation.steering_control()
(built-in function), 121	(built-in function), 128
steering_interpolate_and_print.printInterpolate(\$simulation.update_steering_angle())	(built-in function), 128
(built-in function), 121	
steering_interpolate_multiple_trajectories(\$simulation.steering_simulation_old_hand)	(built-in class), 131
(built-in class), 122	
steering_interpolate_multiple_trajectories(\$simulation.steering_simulation_old_hand.checkReaching_angle)	(built-in function), 132
(built-in function), 122	
steering_interpolate_multiple_trajectories(\$simulation.steering_simulation_old_hand.get_angle_difference())	(built-in function), 132
(built-in function), 123	
steering_interpolate_multiple_trajectories(\$simulation.steering_simulation_old_hand.get_joint_position())	(built-in function), 131
(built-in function), 122	
steering_interpolate_multiple_trajectories(\$simulation.steering_simulation_old_hand.get_joint_trajectory)	(built-in function), 131
(built-in function), 122	
steering_interpolate_multiple_trajectories(\$simulation.steering_simulation_old_hand.interpolate_joint_angles)	(built-in function), 132
(built-in function), 122	
steering_interpolate_multiple_trajectories(\$simulation.steering_simulation_old_hand.joint_state_callback())	(built-in function), 131
(built-in function), 122	
steering_response_test	(built-in class), 124
(built-in class), 124	steering_simulation_old_hand.main()
steering_response_test.check_joint_angle()	(built-in function), 133
(built-in function), 124	
steering_response_test.import_joint_trajectory	(built-in function), 132
(built-in function), 124	
steering_response_test.joint_state_callback()	(built-in function), 131
(built-in function), 124	
steering_response_test.main()	(built-in function), 132
(built-in function), 125	
steering_response_test.regress_joint_positions	(built-in function), 132
(built-in function), 124	
steering_response_test.steering_angle_reached()	(built-in function), 132
(built-in function), 124	
steering_response_test.steering_test()	steering_trajectory_following_test
(built-in function), 125	(built-in class), 135
steering_simulation	(built-in class), 127
(built-in class), 127	
steering_simulation.check_steering_angles	steering_trajectory_following_test.computeSteeringAngles()
(built-in function), 128	(built-in function), 135
steering_simulation.get_angle_differences	steering_trajectory_following_test.getPositionLeftHandTrajectory()
(built-in function), 128	(built-in function), 135

(built-in function), 135
 steering_trajectory_following_test.getPositionFunction(), 74
 (built-in function), 135
 steering_trajectory_following_test.jointStateCFunction(), 74
 (built-in function), 135
 steering_trajectory_following_test.main() (C++ function), 73
 (built-in function), 136
 steering_trajectory_following_test.printPlannedFunction(), 74
 (built-in function), 136
 steering_trajectory_following_test.recordActualHandTrajectories()
 (built-in function), 135
 steering_trajectory_following_test.regressionFunctionFromFile() (C++ function), 76
 (built-in function), 135
 steering_trajectory_following_test.setJointControllerParameters() (C++ member), 78
 (built-in function), 135
 UDP socket broadcast addr len (C++ member), 78
 UDP socket broadcast host IP (C++ function), 76
 UDP socket broadcast UDP (C++ function), 78
 UDP socket buf (C++ member), 77
 UDP socket client addr (C++ member), 77
 UDP socket client addr len (C++ member), 78
 UDP socket convert byte 2 text (C++ function), 77
 UDP socket convert text 2 byte (C++ function), 77
 UDP socket exclusive (C++ member), 78
 UDP socket initialized (C++ member), 78
 UDP socket my broadcast IP (C++ member), 77
 UDP socket my IP (C++ member), 77
 UDP socket numbytes (C++ member), 77
 UDP socket receive host IP (C++ function), 76
 UDP socket receive sensor data (C++ function), 76, 77
 UDP socket receive UDP (C++ function), 77
 UDP socket receive UDP from client (C++ function), 77
 UDP socket send UDP to client (C++ function), 77
 UDP socket server addr (C++ member), 78
 UDP socket server addr len (C++ member), 78
 UDP socket servinfo (C++ member), 78
 UDP socket set time out (C++ function), 78
 UDP socket sockfd (C++ member), 78
 UDP socket UDP socket (C++ function), 75, 76
 UDP socket whats my IP (C++ function), 78
 UDP socket ptr (C++ type), 140
 unpack754 (C++ function), 140, 141
 unpack754_32 (C macro), 140
 unpack754_64 (C macro), 140
 update (C++ function), 144–149
 user (C++ member), 141

T

TheClaw (C++ class), 73
 TheClaw::external_robot_state (C++ member), 73
 TheClaw::meterPerSecondToServoSpeed (C++ function), 73
 TheClaw::motor_command (C++ member), 73
 TheClaw::nh (C++ member), 73
 TheClaw::read (C++ function), 73
 TheClaw::TheClaw (C++ function), 73
 TheClaw::write (C++ function), 73
 TorquePositionController (C++ class), 73
 TorquePositionController::controller_parameters (C++ member), 75
 TorquePositionController::controller_state (C++ member), 75
 TorquePositionController::init (C++ function), 73
 TorquePositionController::joint (C++ member), 75
 TorquePositionController::joint_command (C++ member), 75
 TorquePositionController::joint_name (C++ member), 75
 TorquePositionController::JointCommand (C++ function), 74
 TorquePositionController::Kd (C++ member), 75
 TorquePositionController::Kp (C++ member), 75
 TorquePositionController::nh (C++ member), 75
 TorquePositionController::q_target (C++ member), 75
 TorquePositionController::setControllerParameters (C++ function), 74
 TorquePositionController::spinner (C++ member), 75

V

`velocity_test` (*built-in class*), 138
`velocity_test.evaluate_current_pedal_angle()`
 (*built-in function*), 138
`velocity_test.evaluate_error()` (*built-in function*), 138
`velocity_test.get_angle_difference()`
 (*built-in function*), 138
`velocity_test.get_position_left_foot()`
 (*built-in function*), 138
`velocity_test.get_position_right_foot()`
 (*built-in function*), 138
`velocity_test.get_twist()` (*built-in function*), 138
`velocity_test.main()` (*built-in function*), 138
`velocity_test.reality_test_acceleration()`
 (*built-in function*), 138
`velocity_test.simulation_test()` (*built-in function*), 138
`velocity_test.velocity_reached()` (*built-in function*), 138
`visualization_msgs` (C++ *type*), 139
`VRpuppet` (C++ *class*), 79
`VRpuppet::endeffector_jointnames` (C++ *member*), 79
`VRpuppet::endeffectors` (C++ *member*), 79
`VRpuppet::external_robot_state` (C++ *member*), 79
`VRpuppet::motor_command` (C++ *member*), 79
`VRpuppet::motor_config` (C++ *member*), 79
`VRpuppet::motor_control_mode` (C++ *member*), 79
`VRpuppet::nh` (C++ *member*), 79
`VRpuppet::read` (C++ *function*), 79
`VRpuppet::sphere_left_axis0_params` (C++ *member*), 79
`VRpuppet::sphere_left_axis1_params` (C++ *member*), 79
`VRpuppet::sphere_left_axis2_params` (C++ *member*), 79
`VRpuppet::VRpuppet` (C++ *function*), 79
`VRpuppet::write` (C++ *function*), 79

Y

`Yatr` (C++ *class*), 80
`Yatr::external_robot_state` (C++ *member*), 80
`Yatr::motor_command` (C++ *member*), 80
`Yatr::nh` (C++ *member*), 80
`Yatr::read` (C++ *function*), 80
`Yatr::write` (C++ *function*), 80
`Yatr::Yatr` (C++ *function*), 80