



**The
Alan Turing
Institute**

KING'S
College
LONDON



HIRI
Honda Research Institute **EU**

ROS-PyBullet Interface

https://github.com/ros-pybullet/ros_pybullet_interface

`christopher.mower@kcl.ac.uk`

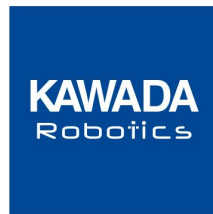
Workshop 26 April 2022

Christopher E. Mower, Theodoros Stouraitis, Lei Yan, João Moura, Christian Rauch
Nazanin Zamani Behabadi, Michael Gienger, Tom Vercauteren, Christos Bergeles, Sethu Vijayakumar



Harmony

Assistive robots for healthcare

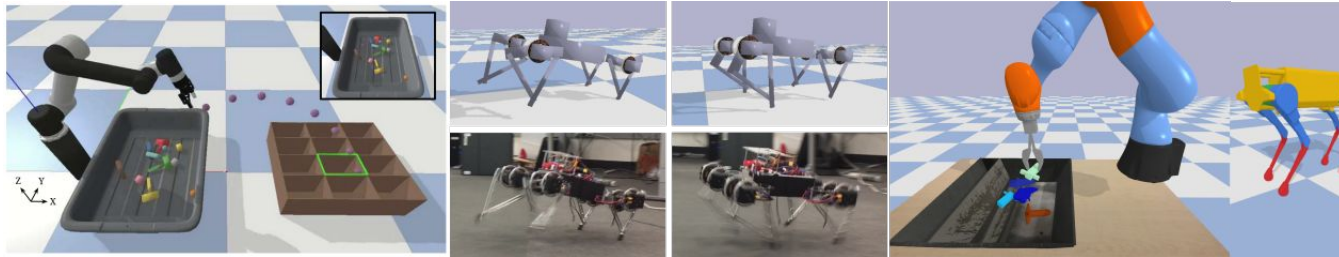
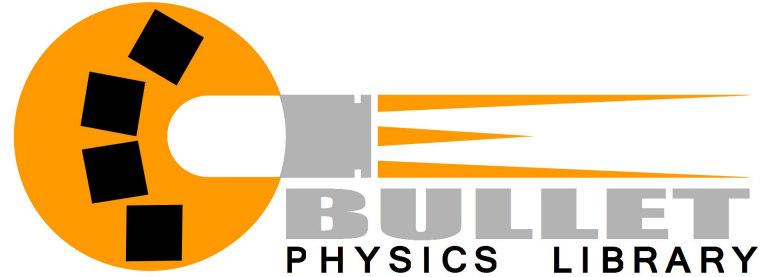


Prerequisites

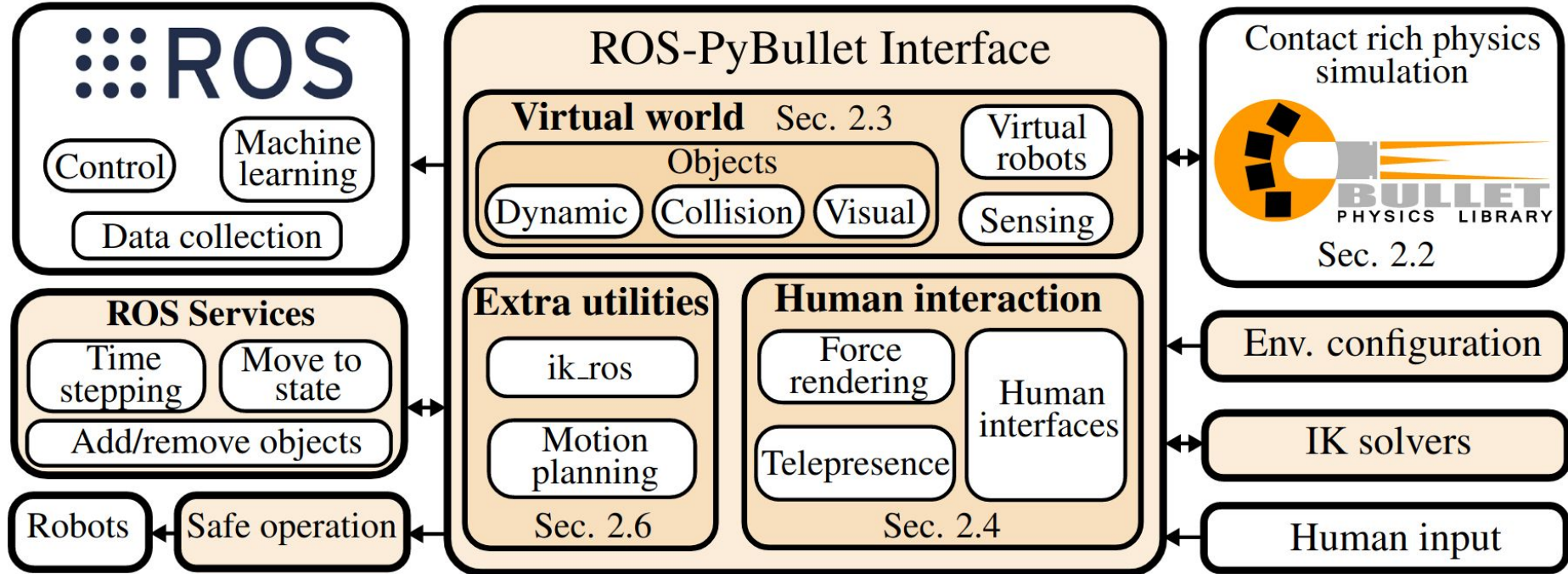
- Knowledge
 - Familiarity with ROS
 - [Topics, Messages, Services, Publishers, Subscribers, Parameters, etc](#)
 - Familiarity with [Pybullet](#) (helps but not essential)
- Setup (ideal)
 - Ubuntu 20.04
 - ROS Noetic
 - Python 3.7>=
 - [catkin tools](#) (\$ sudo pip3 install -U catkin_tools)
- Other setups that *should* work
 - Ubuntu 18.04
 - ROS Melodic
- Not supported!
 - ROS2 (currently)
 - Python 2

Pybullet

- [Pybullet](#) is a python wrapper for the Bullet physics library
- Created by [Erwin Coumans](#)
- Bullet is accepted in community as a reliable simulator for modelling contact/impact



In a nutshell...



History

- In development ~2 years, public release coming soon
- Motivation
 - Reliable contact modelling
 - Connection to ROS
 - Easily interface with hardware
 - Easily develop/prototype
- Started by Chris/Theo
- Other contributors
 - Lei Yan
 - João Moura
 - Christian Rauch
 - Nazanin Zamani Behabadi
- Work utilizing framework
 - [Mower2021a, Moura2021, Li2022, Stouraitis2021, Mower2021b]

[Mower2021a] C. E. Mower, J. Moura, S. Vijayakumar, [Skill-based Shared Control](#), R:SS, 2021.

[Moura2021] J. Moura, T. Stouraitis, S. Vijayakumar, [Non-prehensile Planar Manipulation via Trajectory Optimization with Complementarity Constraints](#), arXiv, 2021 [accepted at ICRA 2022].

[Li2022] S. Li, T. Stouraitis, M. Gienger, S. Vijayakumar, J. A. Shah, [Set-based State Estimation with Probabilistic Consistency Guarantee under Epistemic Uncertainty](#), RAL, 2022.

[Stouraitis2021] T. Stouraitis, [Dyadic collaborative manipulation formalism for optimizing human-robot teaming](#), PhD Thesis, University of Edinburgh, 2021.

[Mower2021b] C. E. Mower, [An Optimization-based Formalism for Shared Autonomy in Dynamic Environments](#), PhD Thesis, University of Edinburgh, 2021.

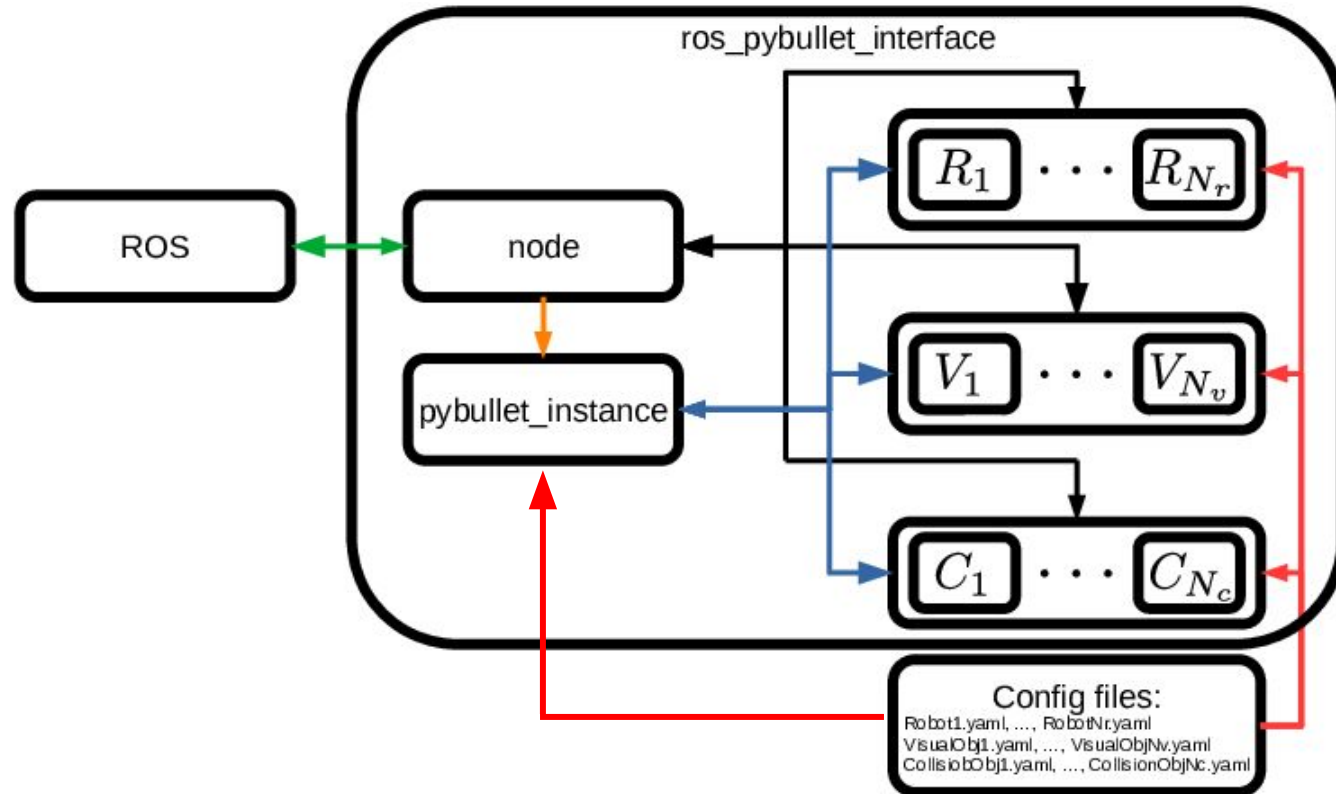
Alternatives?

	ROS	Languages	Deform. Obj.	Hardware	HRI	Photo-realistic
Drake	✗	C++/Python	✓	✗	✗	✗
Gazebo	✓	C++	✗	✓	✗	✗ ³
Nvidia Isaac	✓ ⁴	Python	✗	✓	✗	✓
MuJoCo ⁵	✗	C++/Python	✓	✗	✗	✗
ROS-PyBullet	✓	Python	✓	✓	✓	✗ ⁶

Framework features

1. Online, full-physics simulation of robots using a reliable simulator for contacts.
2. Integration with ROS ecosystem.
3. Simulated sensing (e.g. F-T sensor, RGBD camera).
4. Several interfaces for humans.
5. Modular and extensible design.
6. Easily integrates with hardware.

System overview



Robots and building virtual worlds (1/5)

- PybulletObject
 - Initialize common variables
 - Methods to facilitate ROS communication
 - Load configuration files
 - Interface methods to Pybullet

Robots and building virtual worlds (2/5)

- `PybulletRobot(PybulletObject)`
 - Simulates or visualizes robot
 - Supports both fixed based and non fixed based robots
 - Load from URDF (support for SDF and Mujoco files coming soon)
 - Target joint states: position/velocity/torque control are supported
 - Publishes joint/link state
 - Simulate F/T sensors
 - Services
 - Move to joint/end-effector state
 - Robot information

Robots and building virtual worlds (3/5)

- `PybulletVisualObject(PybulletObject)`
 - Simply visualizes objects
 - No interaction with any other body
 - Use-case: visualize real world objects [Moura2021]

Robots and building virtual worlds (4/5)

- `PybulletCollisionObject(PybulletObject)`
 - Model objects such as floors/walls/etc
 - Other bodies react to these objects, but they do not react (massless)
 - Use-case: represent a wiping surface [Mower2021]

Robots and building virtual worlds (5/5)

- `PybulletDynamicObject(PybulletObject)`
 - Objects whose motion is fully defined by Pybullet
 - Initial pose and linear/angular velocity given
 - Pose broadcast to ROS
 - Use-case: simulate a pushing box [Moura2021]

Communication with ROS

- Services
 - Add/remove pybullet objects programmatically
 - Robot
 - Move to joint state
 - Move to end-effector state
 - Robot information
 - Time-step: start, step, stop
- Topics/Messages
 - Status
 - Object pose
 - Dynamic object broadcasts /tf
 - Other objects pose specified by /tf
 - Robot
 - Current joint state /rpbi/ROBOT_NAME/joint_states (sensor_msgs/JointState)
 - Target joint state /rpbi/ROBOT_NAME/joint_states/target (sensor_msgs/JointState)
 - Simulated sensors (FT-sensors, RGBD camera)
 - Pybullet visualizer, i.e. control GUI viewer programmatically

Additional features

- MPC utilities
 - Start, step, stop time via buttons in GUI
- IK utilities
 - https://github.com/cmower/ik_ros provides a standardized IK interface
 - Supports: [trac_ik](#), [EXOTica](#), [RBDL](#), pybullet
- Interpolation utilities
 - Interpolate planned trajectories
- Safe robot utilities
 - https://github.com/cmower/safe_robot provides a generalized safety checking node
 - Checks joint positions/velocities, link limits, self-collisions
 - Easy to plugin custom checks

Summary

- Outline
 - Framework that integrates ROS with a reliable impact/contact simulator Pybullet
 - Supports several robot/tasks/interfaces
 - Open source
 - Easy to install, setup, and use
- Future work
 - Migrate to ROS2
 - Support other Pybullet features, e.g. virtual reality
 - Full documentation

Now

- Install `ros_pybullet_interface` and dependencies
- Basic “hello, world” example
- Adding objects programmatically
- RGBD Sensor
- Inverse Kinematics using `ik_ros`
- Leverage ROS ecosystem (teleop+LfD)

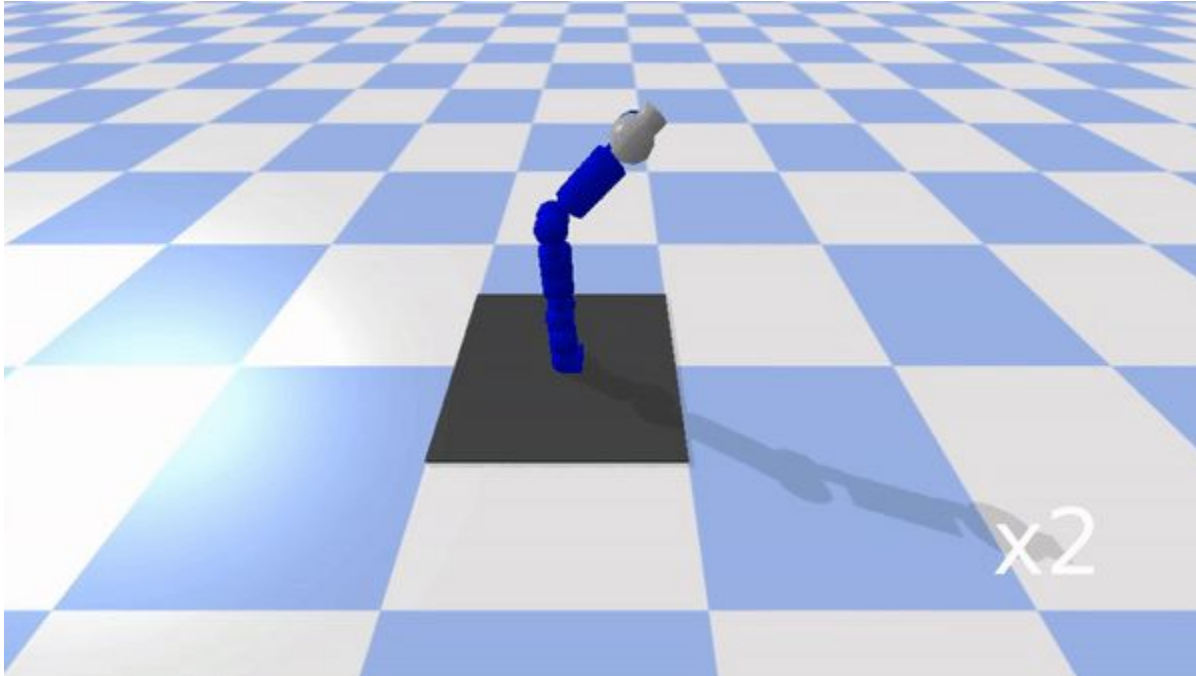
Installation

In theory...

1. [Create catkin workspace](#)
 - a. `$ mkdir -p your_catkin_ws/src && cd your_catkin_ws`
 - b. `$ catkin init`
2. `$ cd src`
3. `$ git clone git@github.com:ros-pybullet/ros_pybullet_interface.git`
4. `$./ros_pybullet_interface/install.sh # automatically calls catkin build`

Basic “hello, world” example

1. `$ roslaunch rpbi_examples basic_example_kuka_lwr.launch`



Note, this records an MP4 video and saves it to `$HOME/basic_example_kuka_lwr.mp4`

Configuring a new robot and environment

- Main config file for `ros_pybullet_interface_node.py`
- Design of parameters:
 - When `camelCase` is used, this matches [Pybullet documentation](#) (if unspecified, default values specified in the documentation are used unless the variable is required, e.g. if `setGravity` is not specified in the config file then `gravD=0.0` for all $D=\{X,Y,Z\}$)
 - When `snake_case` is used, this means it is an interface specific parameters

```
#
# Pybullet instance
#
connect:
  connection_mode: 'GUI'
  options: '--mp4=$HOME/basic_example_kuka_lwr.mp4'

setGravity:
  gravX: 0.0
  gravY: 0.0
  gravZ: -9.81

timeStep: 0.01
start_pybullet_after_initialization: true
status_hz: 50

#
# Pybullet visualizer
#
configureDebugVisualizer:
  enable: 0
  flag: 'COV_ENABLE_GUI'

resetDebugVisualizerCamera:
  cameraDistance: 2.0
  cameraYaw: 0.0
  cameraPitch: -45.0
  cameraTargetPosition: [0.0, 0.0, 0.0]

#
# Pybullet objects
#
collision_objects:
  - "{rpbi_examples}/configs/floor.yaml"
robots:
  - "{rpbi_examples}/configs/basic_example_kuka_lwr/kuka_lwr.yaml"
rpbi_examples/configs/basic_example_kuka_lwr/config.yaml
```

Configuring a new robot and environment

- Pybullet instance
 - General configuration of Pybullet
- Pybullet visualizer
 - Configuration of the visualization GUI
 - Also configurable/updateable via ROS
- Pybullet objects
 - visual_objects
 - collision_objects
 - dynamic_objects
 - sensors
 - robots

Notice: Specify relative file names using
{ros_package_name}

```
#
# Pybullet instance
#
connect:
  connection_mode: 'GUI'
  options: '--mp4=$HOME/basic_example_kuka_lwr.mp4'

setGravity:
  gravX: 0.0
  gravY: 0.0
  gravZ: -9.81

timeStep: 0.01
start_pybullet_after_initialization: true
status_hz: 50

#
# Pybullet visualizer
#
configureDebugVisualizer:
  enable: 0
  flag: 'COV_ENABLE_GUI'

resetDebugVisualizerCamera:
  cameraDistance: 2.0
  cameraYaw: 0.0
  cameraPitch: -45.0
  cameraTargetPosition: [0.0, 0.0, 0.0]

#
# Pybullet objects
#
collision_objects:
  - "{rpbi_examples}/configs/floor.yaml"
robots:
  - "{rpbi_examples}/configs/basic_example_kuka_lwr/kuka_lwr.yaml"
```

[rpbi_examples/configs/basic_example_kuka_lwr/config.yaml](#)

Configuring a new robot and environment

- Each object must have a unique name
- Initial joint positions
 - Revolute joints assumed to be in radians unless flag is true (as in config)
 - Joints not specified default to 0.0
- Robot links can be broadcast as tf's
- Make robot visual (i.e. simply visualize the robot)
- Simulate F/T sensors

```
name: "kuka_lwr"

loadURDF:
  fileName: "{rpbi_examples}/robots/kuka_lwr.urdf"
  useFixedBase: 1
  basePosition: [0.5, 0., 0.]

initial_joint_position:
  lwr_arm_0_joint: 0.0
  lwr_arm_1_joint: 30.0
  lwr_arm_2_joint: 0.0
  lwr_arm_3_joint: -90.0
  lwr_arm_4_joint: 0.0
  lwr_arm_5_joint: 60.0
  lwr_arm_6_joint: 0.0

initial_revolute_joint_positions_are_deg: true
is_visual_robot: false

setJointMotorControlArrayParameters:
  controlMode: "POSITION_CONTROL"

enabled_joint_force_torque_sensors:
  - "lwr_arm_6_joint"

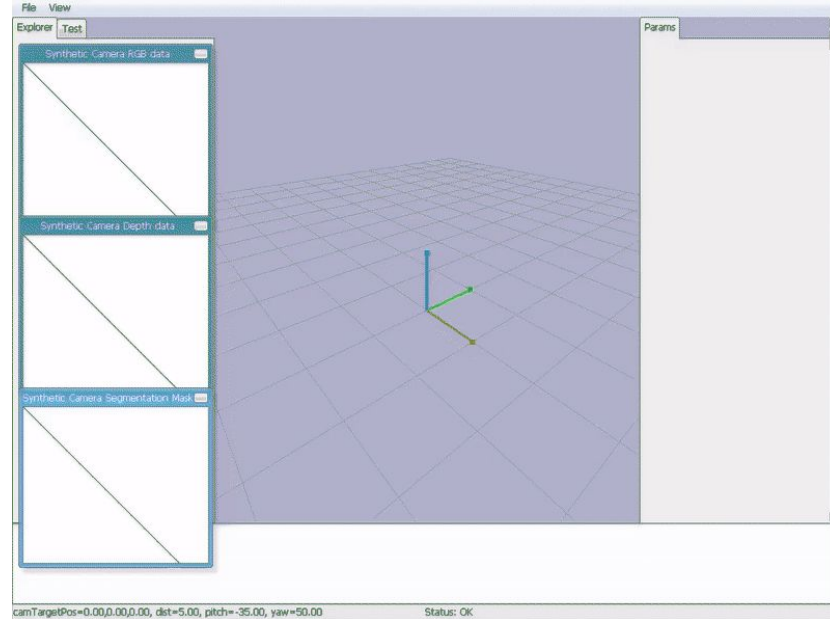
publish_joint_state_frequency: 30

publish_link_states: true
publish_link_states_frequency: 30
```

[rpbi_examples/configs/basic_example_kuka_lwr/kuka_lwr.yaml](#)

Adding objects programmatically

1. `$ roslaunch rpbi_examples pybullet_objects_example.launch`



Adding objects programmatically

- Services
- rpbi/add_pybullet_object
 - Add pybullet object from file
 - Add pybullet object from config (use [custom_ros_tools.config.config_to_str](#))
- rpbi/remove_pybullet_object
 - Removes object using name

```
from cob_srvs.srv import SetString, SetStringRequest
from ros_pybullet_interface.msg import PybulletObject
from ros_pybullet_interface.srv import AddPybulletObject

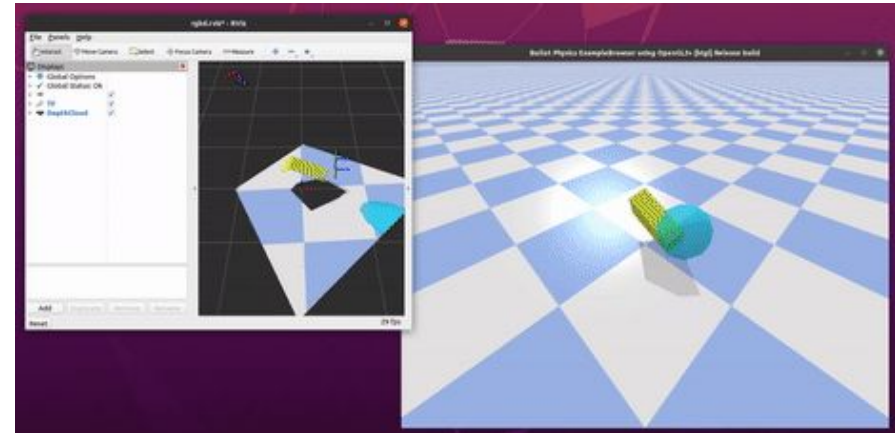
def add_pybullet_object():
    rospy.wait_for_service('rpbi/add_pybullet_object')
    try:
        handle = rospy.ServiceProxy(srv, AddPybulletObject)
        req = PybulletObject()
        req.object_type = PybulletObject.DYNAMIC
        req.filename = '{rpbi_examples}/configs/pybullet_objects_example/dynamic_box.yaml'
        resp = handle(req)
        if resp.success:
            rospy.loginfo('successfully added object')
        else:
            rospy.logwarn('failed to add object: %s' % resp.message)
    except Exception as e:
        rospy.logerr('failed to add object: %s' % str(e))

def remove_object():
    rospy.wait_for_service('rpbi/remove_pybullet_object')
    try:
        handle = rospy.ServiceProxy(srv, SetString)
        object_name = "dynamic_box"
        req = SetStringRequest(data=object_name)
        resp = handle(req)
        if resp.success:
            rospy.loginfo('successfully removed object')
        else:
            rospy.logwarn('failed to remove object: %s' % resp.message)
    except Exception as e:
        rospy.logerr('failed to remove object: %s' % str(e))
```


RGBD Sensor

1. `$ roslaunch rpbi_examples pybullet_rgbd_example.launch`

- Simulate vision data
- Attach sensor to /tf (e.g. robot link)
- Limitation: only one can be instantiated at a time



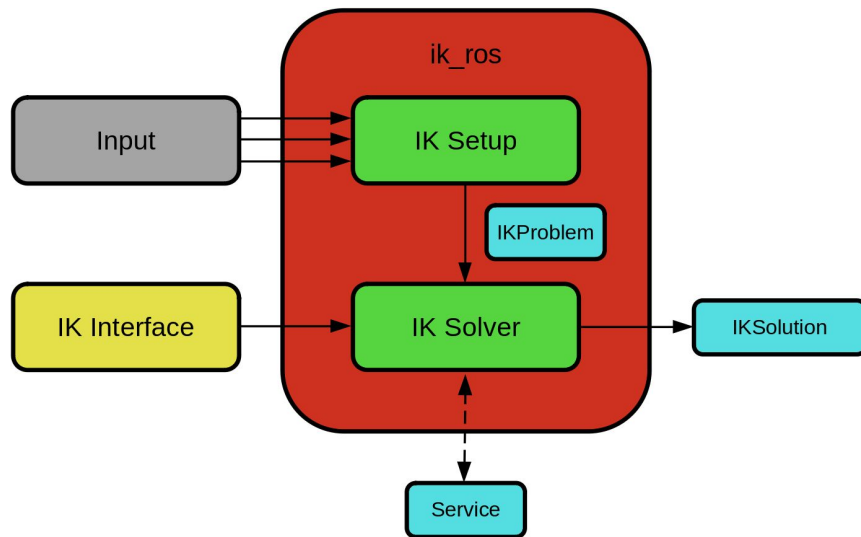
Inverse Kinematics using ik_ros

1. `$ roslaunch ik_ros_examples kuka_lwr_exotica_example.launch`



Inverse Kinematics using ik_ros

- ik_ros is a standardized interface to several IK solvers
- IK Setup node
 - Listens to input streams of data (e.g. /tf) that define the task space goal
 - Packs/publishes an IK problem message
- IK Solver node
 - Interfaces with a given solver (currently supported: EXOTica, trac_ik, pybullet)
 - Solves problem and publishes solution
- Services:
 - Solve ik
 - Toggle IK callback (start/stop)
- ik_ros is extensible: straightforward to add additional/custom IK interfaces



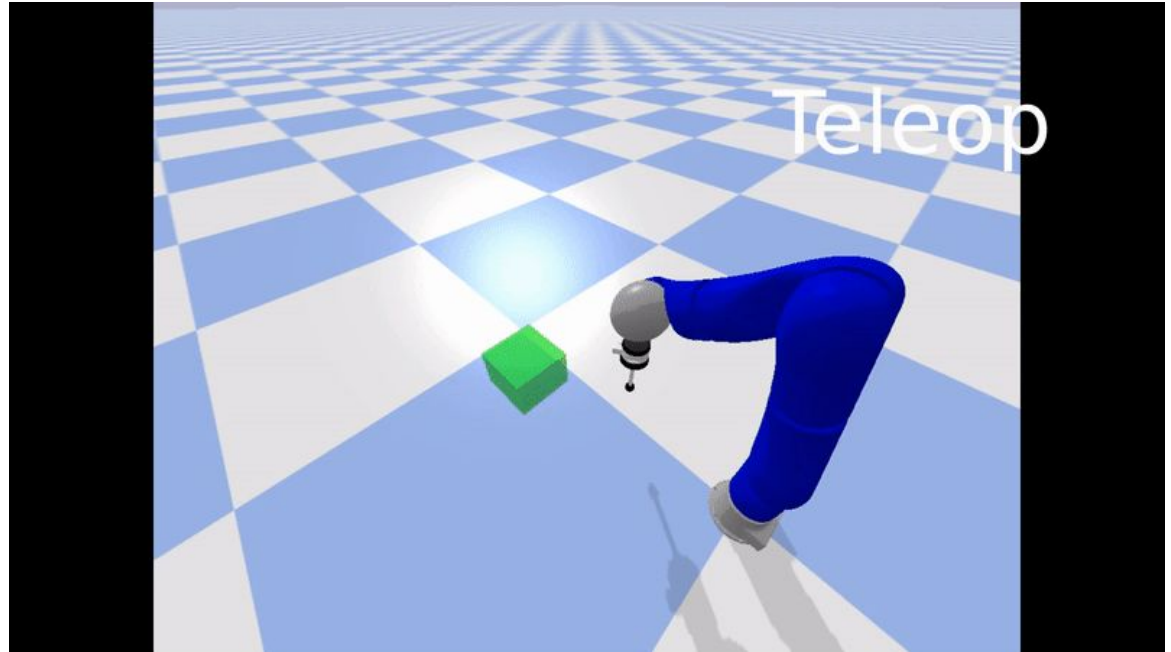
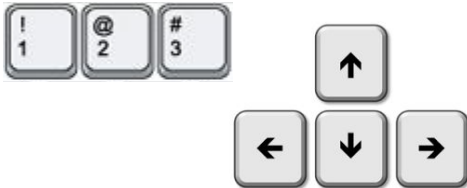
Leverage ROS ecosystem (teleop+LfD)

1. `$ roslaunch rpbi_examples lfd.launch`

Keys:

1. Move robot to start configuration
2. Start/stop teleoperation demo
3. Learn, plan and execute LfD

left/right/up/down direction keys
move end-effector when in
teleoperation demo



Thank you

Questions, feedback, pull requests welcome!