PyDART2 Documentation

Release 0.3.11

Sehoon Ha

May 25, 2018

Contents

PyDART2 is an open source python binding of DART(6.0.1), an open source physics simulator. Its APIs are designed to provide concise and powerful control on DART physics worlds. Further, a user can write simulations with a numerous python scientific libraries, such as NumPy(linear algebra), SciPy(optimization), scikit-learn (machine learning), PyBrain(machine learning), and so on.

CHAPTER 1

News

- [2016/09/24] PyDART2 now supports both Python2 and Python3.
- [2016/08/05] PyDART2 supports the DART 6.0.1
- [2016/08/05] PyDART is upgraded to PyDART2, for easier installation and richer APIs.

CHAPTER 2

Contents:

2.1 Installation Guide

2.1.1 Install with pip (easy)

In Ubuntu, PyDART2 can be easily installed using PyPI - the Python Package Index. The default Python environment is assumed to be Python3, although PyDART2 is also available in Python2.

The first step is to install DART 6.0.1 (You can skip this if you already have it!). Please use your favorite method to install DART, such as, ..

```
sudo apt-add-repository ppa:dartsim
sudo apt-get update
sudo apt-get install libdart6-all-dev
```

Please refer the official DART installation document (<https://github.com/dartsim/dart/wiki/Installation>) when you have problems.

The next step is to install SWIG, pip3, and PyQt4.

They can be installed by the following command:

```
sudo apt-get install swig python3-pip python3-pyqt4 python3-pyqt4.qtopengl
```

The final step is to install PyDART2 using pip3.

```
sudo pip3 install pydart2
```

All done! Please enjoy the simulation.

```
$ python3
>>> import pydart2 as pydart
>>> pydart.init(verbose=True)
Msg  [pydart2_api] Initialize pydart manager OK
```

For Python2 users, please apply the following commands:

```
sudo apt-get install swig python-pip python-qt4 python-qt4-dev python-qt4-gl
sudo pip install pydart2
```

2.1.2 Install from source code

Sometimes, you want to edit source codes by yourself. For the following steps, I assumed that you already installed the required packages - swig, pip, PyQt4, and so on.

First, please check out the repository.

```
git clone https://github.com/sehoonha/pydart2.git
cd pydart2
```

The next step is to compile the package using setup.py

```
python setup.py build build_ext
```

The final step is to install the python package as a development.

```
python setup.py develop
```

2.1.3 Install using CMake (Old-style)

I also wrote CMakeLists.txt, which is an old-style cross compilation system used in the original PyDART.

2.2 Examples

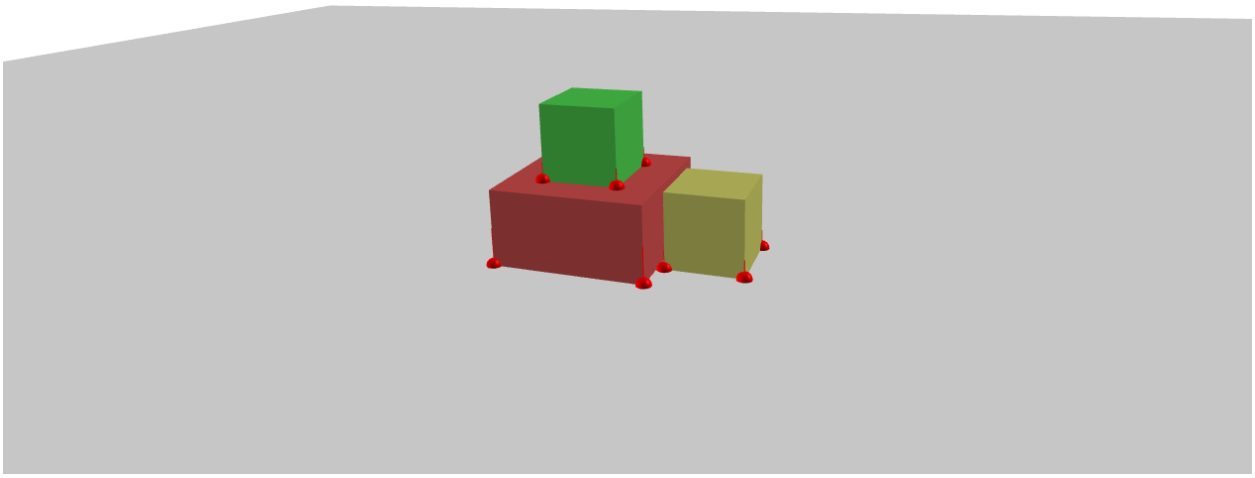
Unfortunately, I cannot write all the documentation by myself. Instead, please use the following examples as tutorial.

For source codes and data files, please visit the repository <https://github.com/sehoonha/pydart2>.

2.2.1 Cubes: Hello, PyDART!

This example loads the simulation of cubes and runs it for 2 seconds.

Screenshot



Code

```
import pydart2 as pydart

if __name__ == '__main__':
    print('Hello, PyDART!')

    pydart.init()
    print('pydart initialization OK')

    world = pydart.World(1.0 / 2000.0, './data/skel/cubes.skel')
    print('pydart create_world OK')

    while world.t < 2.0:
        if world.nframes % 100 == 0:
            skel = world.skeletons[-1]
            print("%.4fs: The last cube COM = %s" % (world.t, str(skel.C)))
        world.step()
```

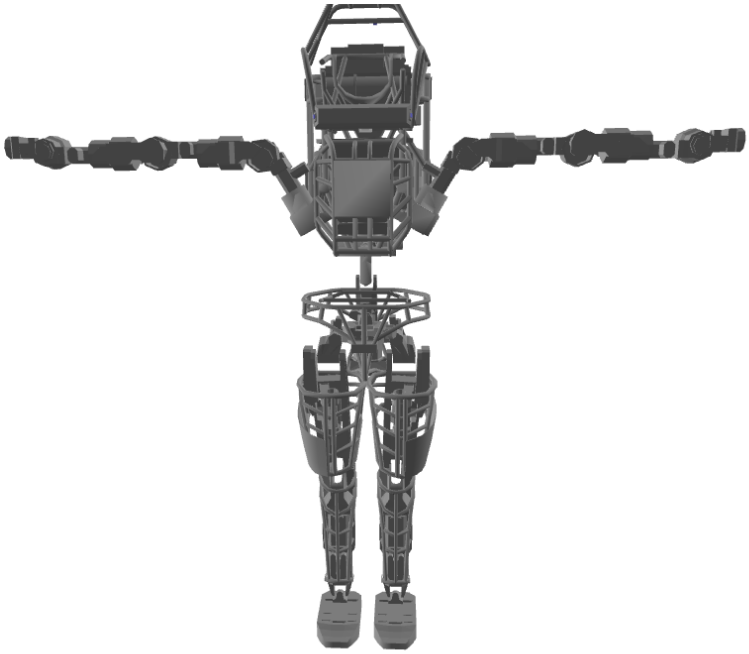
2.2.2 Skeleton Viewer: A basic GUI

This example demonstrates how to load a skeleton, and visualize it using GUI. The following screenshot is the result of the following command.

```
python view_skeleton.py data/sdf/atlas/atlas_v3_no_head.sdf
```

Note: you need to rotate the camera using drag, shift-drag, and control-drag.

Screenshot



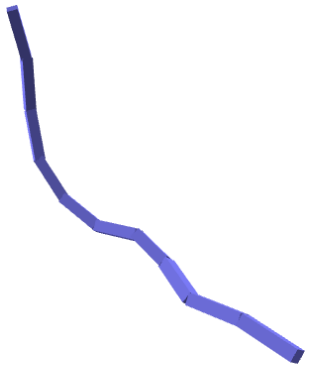
Code

```
if __name__ == '__main__':
    import sys
    import pydart2 as pydart

    if len(sys.argv) != 2:
        print("Usage: view_skeleton.py [*.urdf/*.sdf]")
        exit(0)

    skel_path = sys.argv[1]
    print("skeleton path = %s" % skel_path)

    pydart.init()
    print("Pydart init OK")

    world = pydart.World(1.0 / 1000.0)
    print("World init OK")

    skel = world.add_skeleton(skel_path)
    print("Skeleton add OK")

    print("Camera:")
    print("    drag: rotate camera")
    print("    shift-drag: zoom camera")
    print("    control-drag: translate camera")

    pydart.gui.viewer.launch(world)
```

2.2.3 Chain: A basic controller

This example demonstrates a simple damping controller for a single chain.

Screenshot



Code

```
import numpy as np


class DampingController:
    """ Add damping force to the skeleton """
    def __init__(self, skel):
        self.skel = skel

    def compute(self):
        damping = -0.01 * self.skel.dq
        damping[1::3] *= 0.1
        return damping


if __name__ == '__main__':
    import pydart2 as pydart

    pydart.init(verbose=True)
    print('pydart initialization OK')

    world = pydart.World(1.0 / 5000.0, './data/skel/chain.skel')
    print('pydart create_world OK')
```

(continues on next page)

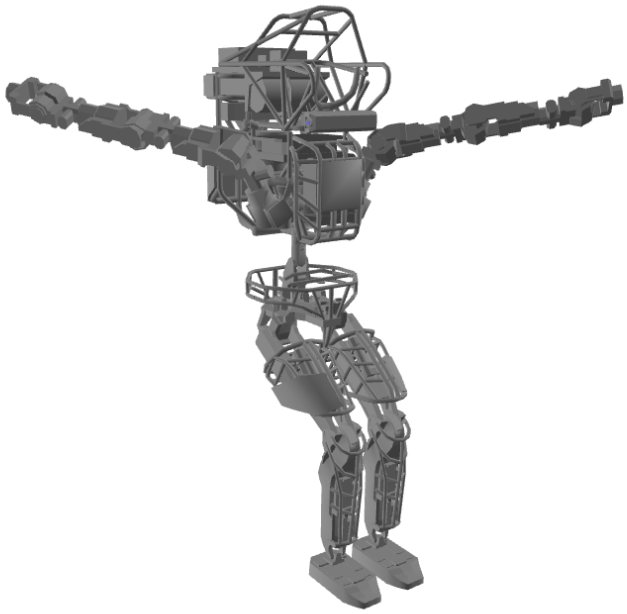
(continued from previous page)

```
    skel = world.skeletons[0]
    skel.q = (np.random.rand(skel.ndofs) - 0.5)
    print('init pose = %s' % skel.q)
    skel.controller = DampingController(skel)

    pydart.gui.viewer.launch(world)
```

2.2.4 Simple Tracking: Robot configuration and PD tracking

This example demonstrates a PD-controller in zero gravity. Please note that the numeric vector of target positions can be configured semantically with names of degrees of freedom.

Screenshot



Code

```
import pydart2 as pydart
import numpy as np


class Controller:
    def __init__(self, skel):
        self.skel = skel
        self.target = None
        self.Kp = np.array([0.0] * 6 + [400.0] * (self.skel.ndofs - 6))
        self.Kd = np.array([0.0] * 6 + [40.0] * (self.skel.ndofs - 6))
```

(continues on next page)

(continued from previous page)

```

    def compute(self):
        return -self.Kp * (self.skel.q - self.target) - self.Kd * self.skel.dq


if __name__ == '__main__':
    print('Example: bipedStand')

    pydart.init()
    print('pydart initialization OK')

    world = pydart.World(1.0 / 2000.0)
    world.set_gravity([0.0, 0.0, 0.0])
    print('World OK')

    skel = world.add_skeleton('./data/sdf/atlas/atlas_v3_no_head.sdf')
    print('Skeleton = ' + str(skel))

    # Set joint damping
    for dof in skel.dofs:
        dof.set_damping_coefficient(80.0)

    # Set target pose
    target = skel.positions()
    target[("l_arm_shy", "r_arm_shy")] = -1.0, 1.0
    target[("l_leg_hpy", "r_leg_hpy")] = -1.0, -1.0
    target[("l_leg_kny", "r_leg_kny")] = 1.0, 1.0

    # Initialize the controller
    controller = Controller(skel)
    controller.target = target
    skel.set_controller(controller)
    print('create controller OK')

    pydart.gui.viewer.launch(world,
                             default_camera=1)  # Use Z-up camera
```

2.2.5 Inspect Skeleton: Navigating Bodynodes, Joints, and Dofs

This example demonstrates how to navigate the components of the given skeleton including body nodes, joints, dofs, shapes, markers, and so on.

```
python inspect_skeleton.py data/sdf/atlas/atlas_v3_no_head.sdf
```

Result

```
----------------------------------------
[Basic information]
        mass = 146.554000
        # DoFs = 33
[BodyNode]
root_bodynode[0] = [BodyNode(0): pelvis]
        [BodyNode(0): pelvis]
```

(continues on next page)

(continued from previous page)

```
                mass = 17.8820Kg
                parent = None
                childs = [[BodyNode(1): ltorso], [BodyNode(9): l_uglut],␣
↪[BodyNode(21): r_uglut]]
                COM = [ 0.0111  0.      0.0271]
                # dependent dofs = 6
                # shapenodes = [[ShapeNode(0:0)], [ShapeNode(0:1)]]
                # markers = 0
        [BodyNode(1): ltorso]
                mass = 2.4090Kg
                parent = [BodyNode(0): pelvis]
                childs = [[BodyNode(2): mtorso]]
                COM = [ -2.37984000e-02  -3.15366000e-06   7.46835000e-02]
                # dependent dofs = 7
                # shapenodes = [[ShapeNode(1:0)], [ShapeNode(1:1)]]
                # markers = 0
        [BodyNode(2): mtorso]
                mass = 0.6900Kg
                parent = [BodyNode(1): ltorso]
                childs = [[BodyNode(3): utorso]]
                COM = [-0.02066266 -0.0131245   0.1925674 ]
                # dependent dofs = 8
                # shapenodes = [[ShapeNode(2:0)], [ShapeNode(2:1)]]
                # markers = 0
...
```

Code

```
if __name__ == '__main__':
    import sys
    import pydart2 as pydart

    if len(sys.argv) != 2:
        print("Usage: inspect_skeleton.py [*.urdf/*.sdf]")
        exit(0)

    skel_path = sys.argv[1]
    print("skeleton path = %s" % skel_path)

    pydart.init()
    print("pydart init OK")

    world = pydart.World(1.0 / 1000.0)
    print("World init OK")

    world.g = [0.0, 0.0, -9.8]
    print("gravity = %s" % str(world.g))

    skel = world.add_skeleton(skel_path)
    print("Skeleton add OK")

    print('----------------------------------------')
    print('[Basic information]')
    print('\tmass = %.6f' % skel.m)
    print('\t# DoFs = %s' % skel.ndofs)
```

(continues on next page)

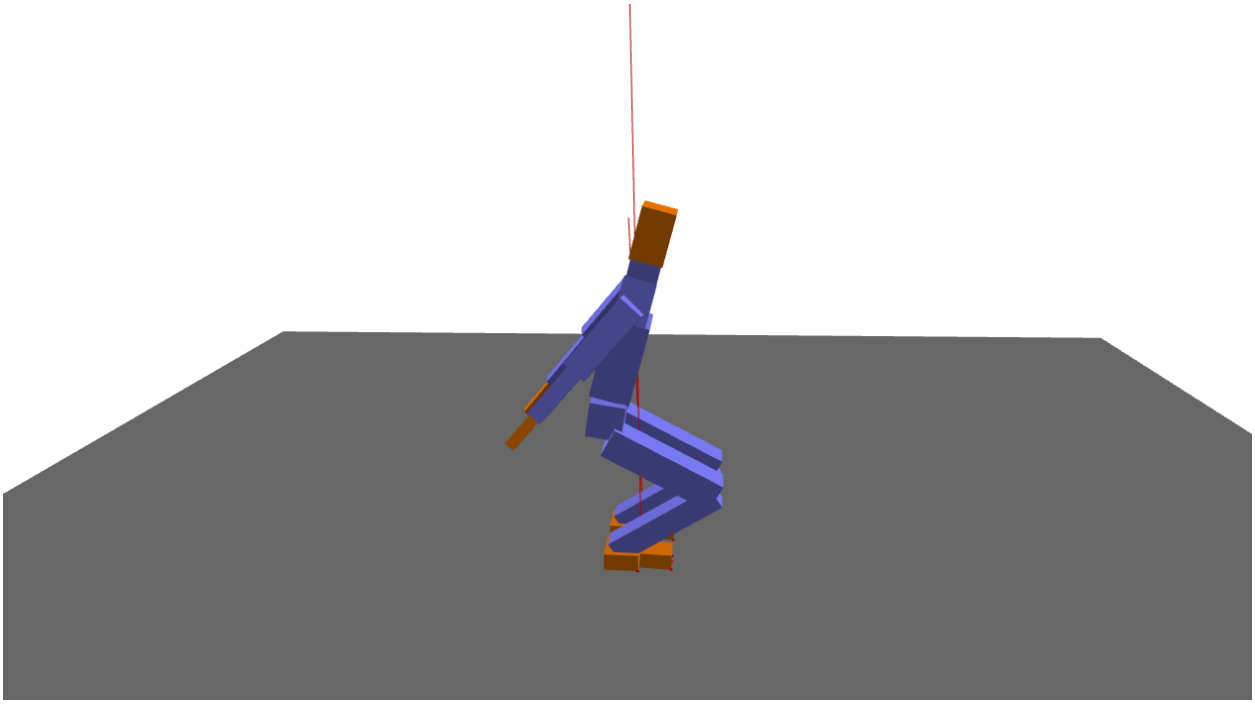
(continued from previous page)

```
    print('[BodyNode]')
    print('root_bodynode[0] = ' + str(skel.root_bodynode(index=0)))
    for body in skel.bodynodes:
        print("\t" + str(body))
        print("\t\tmass = %.4fKg" % body.m)
        print("\t\tparent = " + str(body.parent_bodynode))
        print("\t\tchilds = " + str(body.child_bodynodes))
        print("\t\tCOM = " + str(body.C))
        print("\t\t# dependent dofs = %d" % len(body.dependent_dofs))
        print("\t\t# shapenodes = %s" % str(body.shapenodes))
        print("\t\t# markers = %d" % len(body.markers))
        print("J = %s" % str(body.J))

    print('[DegreeOfFreedom]')
    for dof in skel.dofs:
        print("\t" + str(dof) + " belongs to " + str(dof.joint))
        # print("\t\tindex in skeleton = " + str(dof.index_in_skeleton()))
        # print("\t\tposition = " + str(dof.position()))

    print('[Joint]')
    for joint in skel.joints:
        print("\t" + str(joint))
        print("\t\tparent = " + str(joint.parent_bodynode))
        print("\t\tchild = " + str(joint.child_bodynode))
        print("\t\tdofs = " + str(joint.dofs))

    print('[Markers]')
    for marker in skel.markers:
        print("\t" + str(marker) + " attached to " + str(marker.bodynode))
        print("\t\t" + str(marker.world_position()))

    print('[Position]')
    print('\tpositions = %s' % str(skel.q))
    print('\tvelocities = %s' % str(skel.dq))
    print('\tstates = %s' % str(skel.x))

    print('[Limits]')
    print('\tposition_lower_limits = %s' % str(skel.q_lower))
    print('\tposition_upper_limits = %s' % str(skel.q_upper))
    print('\tforce_lower_limits = %s' % str(skel.tau_lower))
    print('\tforce_upper_limits = %s' % str(skel.tau_upper))

    print('[Lagrangian]')
    print('\tmass matrix = %s' % str(skel.M))
    print('\tcoriolis_and_gravity_forces = %s' % str(skel.c))
    print('\tconstraint_forces = %s' % str(skel.constraint_forces()))
    print('-----------------------------------------')
```

2.2.6 Biped Jump: Using Jacobian

This example demonstrates a jumping controller using Jacobian transpose.

Screenshot



Code

```
import pydart2 as pydart
import numpy as np


class JTController:
    """
    # Usage
    self.jt = JTController(self.skel)
    tau += self.jt.apply( ["l_hand", "r_hand"], f )
    """
    def __init__(self, _skel):
        self.skel = _skel

    def apply(self, bodynames, f):
        if not isinstance(bodynames, list):
            bodynames = [bodynames]
        f = np.array(f)

        tau = np.zeros(self.skel.ndofs)
        for bodyname in bodynames:
            J = self.skel.body(bodyname).linear_jacobian()
            JT = np.transpose(J)
            tau += JT.dot(f)
        return tau


class Controller:
```

(continues on next page)

(continued from previous page)

```
    def __init__(self, skel, h):
        self.h = h
        self.skel = skel
        ndofs = self.skel.ndofs
        self.qhat = self.skel.q
        self.Kp = np.diagflat([0.0] * 6 + [600.0] * (ndofs - 6))
        self.Kd = np.diagflat([0.0] * 6 + [40.0] * (ndofs - 6))

        # Init target poses
        self.init_target_poses()
        # Jacobian transpose
        self.jt = JTController(self.skel)

    def init_target_poses(self):
        skel = self.skel
        I_thigh = skel.dof_indices(["j_thigh_left_z", "j_thigh_right_z"])
        I_shin = skel.dof_indices(["j_shin_left", "j_shin_right"])
        I_heel = skel.dof_indices(["j_heel_left_1", "j_heel_right_1"])

        pose0 = self.skel.q
        pose0[I_thigh] = 1.2
        pose0[I_shin] = -2.0
        pose0[I_heel] = 0.8
        pose0[('j_bicep_left_y', 'j_bicep_right_y')] = 0.5, -0.5

        pose1 = self.skel.q
        pose1[('j_bicep_left_y', 'j_bicep_right_y')] = -2.0, 2.0
        pose1[('j_bicep_left_x', 'j_bicep_right_x')] = 0.5, -0.5

        pose2 = self.skel.q
        pose2[I_thigh] = 0.3  # Thighs

        self.target_poses = [pose0, pose1, pose2]
        self.target_times = [0.0, 0.4, 0.8]

    def update_target_pose(self):
        if len(self.target_times) == 0:
            return
        t = self.skel.world.t
        if self.target_times[0] <= t:
            self.qhat = self.target_poses[0]
            print('update pose! at %.4lf' % t)
            self.target_poses.pop(0)
            self.target_times.pop(0)

    def compute(self):
        self.update_target_pose()
        skel = self.skel

        invM = np.linalg.inv(skel.M + self.Kd * self.h)
        p = -self.Kp.dot(skel.q + skel.dq * self.h - self.qhat)
        d = -self.Kd.dot(skel.dq)
        qddot = invM.dot(-skel.c + p + d + skel.constraint_forces())
        tau = p + d - self.Kd.dot(qddot) * self.h

        t = self.skel.world.t
        if 0.3 < t and t < 0.5:  # Jump!
```

(continues on next page)

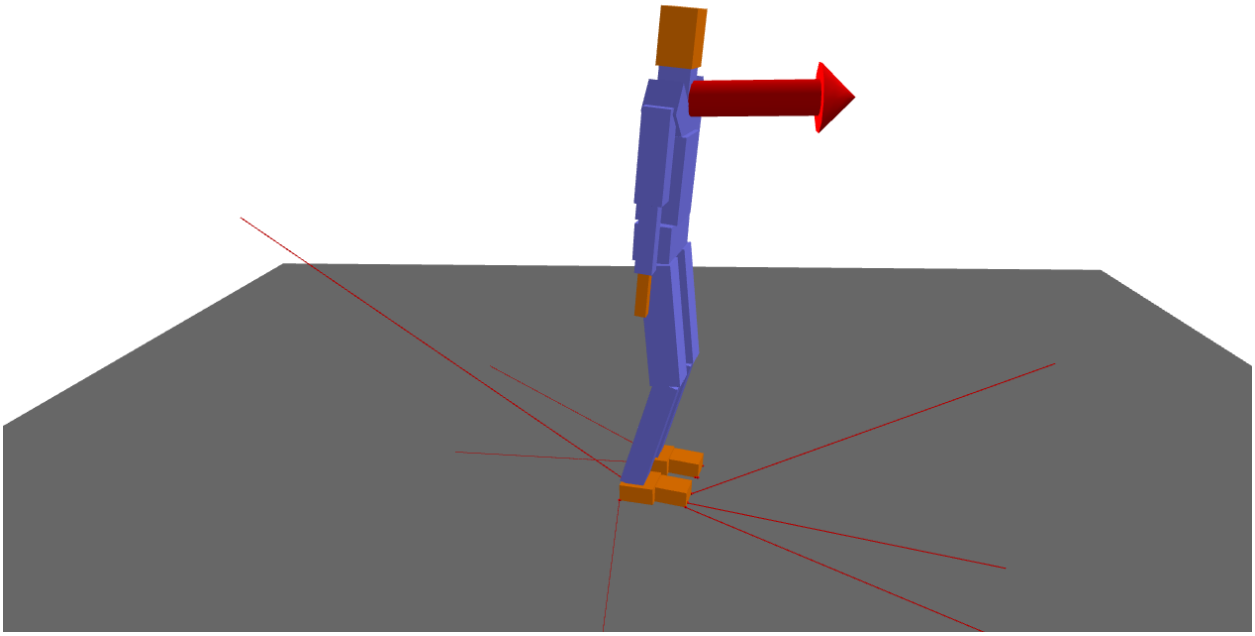
(continued from previous page)

```
            heels = ['h_heel_left', 'h_heel_right']
            vf = self.jt.apply(heels, [0, -700, 0])
            tau += vf

        # Make sure the first six are zero
        tau[:6] = 0
        return tau


if __name__ == '__main__':
    print('Example: bipedJump')

    pydart.init()
    print('pydart initialization OK')

    world = pydart.World(1.0 / 2000.0, './data/skel/fullbody1.skel')
    print('pydart create_world OK')

    # Initialize the pose
    skel = world.skeletons[1]
    q = skel.q
    q[(2, 4, 5)] = [0.02 * np.pi, -0.02, 0]
    skel.set_positions(q)
    print('skeleton position OK')

    # Initialize the controller
    skel.set_controller(Controller(skel, world.dt))
    print('create controller OK')

    pydart.gui.viewer.launch(world)
```

2.2.7 Biped Stand: Balance control + GUI callbacks

This example demonstrates a simple balance controller. The user can interact with a character using keyboards.

Screenshot



Code

```
import pydart2 as pydart
import numpy as np


class Controller:
    def __init__(self, skel, h):
        self.h = h
        self.skel = skel
        ndofs = self.skel.ndofs
        self.qhat = self.skel.q
        self.Kp = np.diagflat([0.0] * 6 + [400.0] * (ndofs - 6))
        self.Kd = np.diagflat([0.0] * 6 + [40.0] * (ndofs - 6))
        self.preoffset = 0.0

    def compute(self):
        skel = self.skel

        # Algorithm:
        # Stable Proportional-Derivative Controllers.
        # Jie Tan, Karen Liu, Greg Turk
        # IEEE Computer Graphics and Applications, 31(4), 2011.

        invM = np.linalg.inv(skel.M + self.Kd * self.h)
        p = -self.Kp.dot(skel.q + skel.dq * self.h - self.qhat)
        d = -self.Kd.dot(skel.dq)
        qddot = invM.dot(-skel.c + p + d + skel.constraint_forces())
        tau = p + d - self.Kd.dot(qddot) * self.h
```

(continues on next page)

(continued from previous page)

```
        # Check the balance
        COP = skel.body('h_heel_left').to_world([0.05, 0, 0])
        offset = skel.C[0] - COP[0]
        preoffset = self.preoffset

        # Adjust the target pose -- translated from bipedStand app of DART
        foot = skel.dof_indices(["j_heel_left_1", "j_toe_left",
                                 "j_heel_right_1", "j_toe_right"])
        if 0.0 < offset < 0.1:
            k1, k2, kd = 200.0, 100.0, 10.0
            k = np.array([-k1, -k2, -k1, -k2])
            tau[foot] += k * offset + kd * (preoffset - offset) * np.ones(4)
            self.preoffset = offset
        elif -0.2 < offset < -0.05:
            k1, k2, kd = 2000.0, 100.0, 100.0
            k = np.array([-k1, -k2, -k1, -k2])
            tau[foot] += k * offset + kd * (preoffset - offset) * np.ones(4)
            self.preoffset = offset

        # Make sure the first six are zero
        tau[:6] = 0
        return tau


class MyWorld(pydart.World):
    def __init__(self, ):
        """
        """
        pydart.World.__init__(self, 1.0 / 2000.0, './data/skel/fullbody1.skel')
        self.force = None
        self.duration = 0

    def on_step_event(self, ):
        if self.force is not None and self.duration >= 0:
            self.duration -= 1
            self.skeletons[1].body('h_spine').add_ext_force(self.force)

    def on_key_press(self, key):
        if key == '1':
            self.force = np.array([50.0, 0.0, 0.0])
            self.duration = 100
            print('push backward: f = %s' % self.force)
        elif key == '2':
            self.force = np.array([-50.0, 0.0, 0.0])
            self.duration = 100
            print('push backward: f = %s' % self.force)

    def render_with_ri(self, ri):
        if self.force is not None and self.duration >= 0:
            p0 = self.skeletons[1].body('h_spine').C
            p1 = p0 + 0.01 * self.force
            ri.set_color(1.0, 0.0, 0.0)
            ri.render_arrow(p0, p1, r_base=0.05, head_width=0.1, head_len=0.1)


if __name__ == '__main__':
```

(continues on next page)

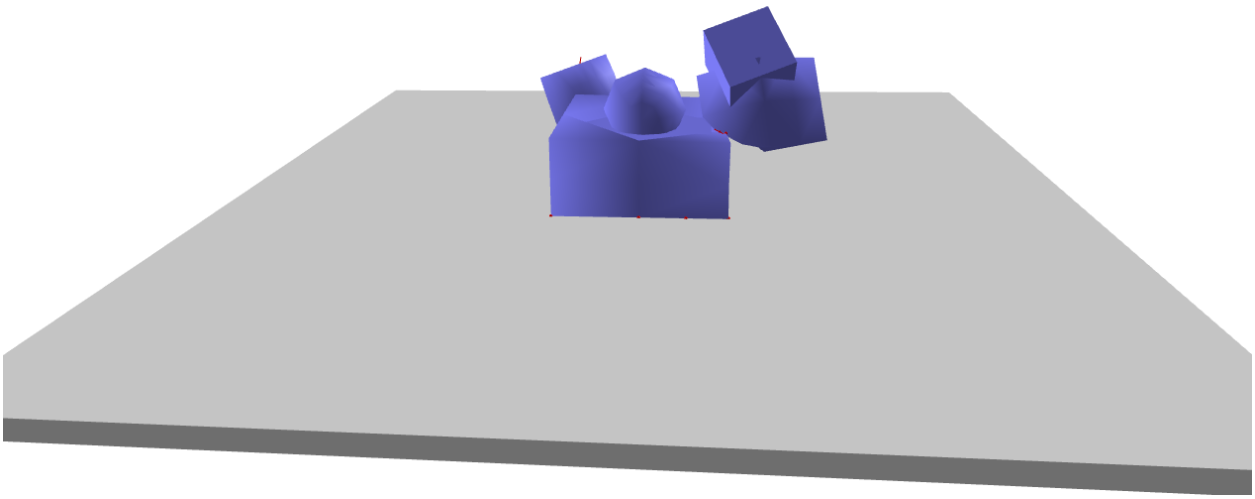
(continued from previous page)

```
    print('Example: bipedStand')

    pydart.init()
    print('pydart initialization OK')

    world = MyWorld()
    print('MyWorld  OK')

    # Use SkelVector to configure the initial pose
    skel = world.skeletons[1]
    q = skel.q
    q["j_pelvis_pos_y"] = -0.05
    q["j_pelvis_rot_y"] = -0.2
    q["j_thigh_left_z", "j_shin_left", "j_heel_left_1"] = 0.15, -0.4, 0.25
    q["j_thigh_right_z", "j_shin_right", "j_heel_right_1"] = 0.15, -0.4, 0.25
    q["j_abdomen_2"] = 0.0
    skel.set_positions(q)
    print('skeleton position OK')

    # Initialize the controller
    skel.set_controller(Controller(skel, world.dt))
    print('create controller OK')

    print("'1'--'2': programmed interaction")
    print("    '1': push forward")
    print("    '2': push backward")
    pydart.gui.viewer.launch(world)
```

2.2.8 Soft Bodies: Softbody simulation

This example loads the simulation of soft bodies

Screenshot



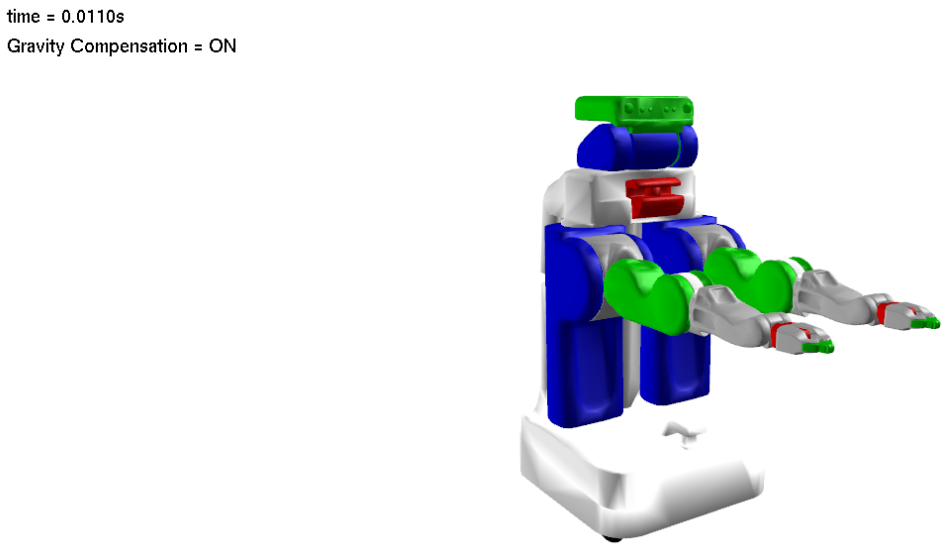
Code

```
import pydart2 as pydart
if __name__ == '__main__':
    pydart.init()
    print('pydart initialization OK')

    world = pydart.World(1.0 / 2000.0, './data/skel/softBodies.skel')
    print('pydart create_world OK')

    pydart.gui.viewer.launch(world)
```

2.2.9 Gravity Compensation Control with a PR2 arm

This example demonstrates a gravity compensation controller with a PR-2 robot arm. Please make sure that you downloaded the URDF files from: https://github.com/sehoonha/pydart2/tree/master/examples/data

Screenshot



Youtube link: https://youtu.be/wzzIyDvc5hQ

Code

```
# Copyright (c) 2015, Disney Research
# All rights reserved.
#
# Author(s): Sehoon Ha <sehoon.ha@disneyresearch.com>
# Disney Research Robotics Group
import pydart2 as pydart
import numpy as np


class GravityCompensationController(object):

    def __init__(self, robot):
        self.robot = robot
        self.g = self.robot.world.gravity()
        self.enabled = True

    def compute(self, ):
        tau = np.zeros(self.robot.num_dofs())
        if not self.enabled:
            return tau

        for body in self.robot.bodynodes:
            m = body.mass()  # Or, simply body.m
            J = body.linear_jacobian(body.local_com())
            tau += J.transpose().dot(-(m * self.g))
        return tau
```

(continues on next page)

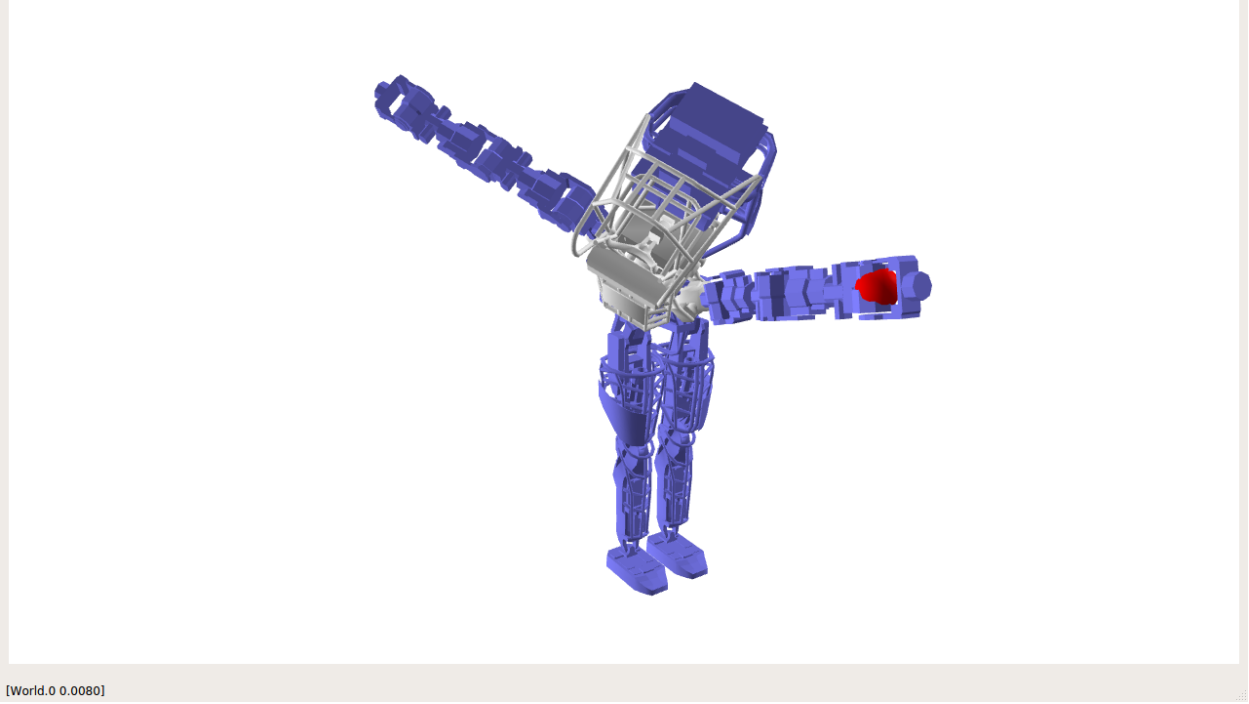
(continued from previous page)

```
class MyWorld(pydart.World):

    def __init__(self, ):
        pydart.World.__init__(self, 0.001)
        self.set_gravity([0.0, 0.0, -9.81])
        print('pydart create_world OK')

        self.robot = self.add_skeleton("./data/urdf/PR2/pr2.urdf")
        print('pydart add_skeleton OK')

        # Lock the first joint
        self.robot.joints[0].set_actuator_type(pydart.joint.Joint.LOCKED)

        # Move bit lower (for camera)
        positions = self.robot.positions()
        positions['rootJoint_pos_z'] = -0.6
        self.robot.set_positions(positions)

        # Initialize the controller
        self.controller = GravityCompensationController(self.robot)
        self.robot.set_controller(self.controller)
        print('create controller OK')

    def on_key_press(self, key):
        if key == 'G':
            self.controller.enabled = not self.controller.enabled

    def draw_with_ri(self, ri):
        ri.set_color(0, 0, 0)
        ri.draw_text([20, 40], "time = %.4fs" % self.t)
        ri.draw_text([20, 70], "Gravity Compensation = %s" %
                     ("ON" if self.controller.enabled else "OFF"))


if __name__ == '__main__':
    print('Example: gravity compensation')

    pydart.init()
    print('pydart initialization OK')

    world = MyWorld()

    win = pydart.gui.viewer.PydartWindow(world)
    win.camera_event(1)
    win.set_capture_rate(10)
    win.run_application()
```

2.2.10 Inverse Kinematics

This example demonstrates how to solve inverse kinematics of Atlas. It solves the problem using Sequential Quadratic Programming from the scipy.optimize package.

Screenshot



Code

```
class MyWorld(pydart.World):

    def __init__(self, ):
        pydart.World.__init__(self, 0.001)
        self.set_gravity([0.0, 0.0, -9.81])
        print('pydart create_world OK')

        filename = "./data/sdf/atlas/atlas_v3_no_head.sdf"
        self.robot = self.add_skeleton(filename)
        self.robot.set_root_joint_to_trans_and_euler()
        print('pydart add_skeleton OK')

        self.theta = 0.0 * np.pi
        self.update_target()
        self.solve()

        print("click step to rotate the target")

    def update_target(self, ):
        th, r = self.theta - 0.5 * np.pi, 0.6
        x, y = r * np.cos(th) + 0.4, r * np.sin(th)
        self.target = np.array([x, y, 0.3])

    def set_params(self, x):
        q = self.robot.positions()
        q[6:] = x
        self.robot.set_positions(q)
```

(continues on next page)

(continued from previous page)

```
28
29      def f(self, x):
30          self.set_params(x)
31
32          lhs = self.robot.body("l_hand").to_world([0.0, 0.0, 0.0])
33          rhs = self.target
34          return 0.5 * np.linalg.norm(lhs - rhs) ** 2
35
36      def g(self, x):
37          self.set_params(x)
38
39          lhs = self.robot.body("l_hand").to_world([0.0, 0.0, 0.0])
40          rhs = self.target
41          J = self.robot.body("l_hand").linear_jacobian()
42          g = (lhs - rhs).dot(J)[6:]
43
44          DEBUG = True
45          if DEBUG:  # Debug by comparing with the numerical computation
46              from pydart2.utils.misc import grad
47              lhs = g
48              rhs = grad(self.f, x, 1e-5)
49              print(lhs)
50              print(rhs)
51              print("OK" if np.allclose(lhs, rhs) else "NG!!!!")
52
53          return g
54
55      def solve(self, ):
56          res = minimize(self.f,
57                         x0=self.robot.positions()[6:],
58                         jac=self.g,
59                         method="SLSQP")
60          print(">>> theta = %.4f" % self.theta)
61          print(res)
62
63      def step(self, ):
64          super(MyWorld, self).step()
65          self.theta = (self.theta + pydart.utils.misc.deg2rad(10)) % np.pi
66          self.update_target()
67          self.solve()
68
69      def render_with_ri(self, ri):
70          ri.set_color(1.0, 0.0, 0.0)
71          ri.render_sphere(self.target, 0.05)
72
73
74  if __name__ == '__main__':
75      print('Example: inverse kinematics')
76
77      pydart.init()
78      print('pydart initialization OK')
79
80      world = MyWorld()
81
82      win = pydart.gui.pyqt5.window.PyQt5Window(world)
83      win.scene.set_camera(1)  # Z-up Camera
84      win.run()
```

CHAPTER 3

Environment

- Ubuntu 16.04
- Python2/Python3
- DART 6.0.1 (or higher): https://github.com/dartsim/dart/

CHAPTER 4

Contact

- GitHub Repository: https://github.com/sehoonha/pydart2

Please contact me when you have questions or suggestions: sehoon.ha@gmail.com

CHAPTER 5

Indices and tables

- genindex
- modindex
- search