Written by Chenhan Zhang, Jaspor Jiang, and Heyun Wang. Apr.04

1. How to setup dVRK environments on Robotorium computer
2. (Optional) On your PC
3. How to run and call GUI
4. How to physically control dVRK arm

1. Section1： How to setup dVRK ROS at Robatarium

First, click this link and follow the instructions.

https://github.com/jhu-cisst/cisst/wiki/Compiling-cisst-and-SAW-with-CMake#setting-up-your-environment-variables-for-ros

The computer at Robatarium has already installed the ROS 20.04 Noetic, so we should skip ROS install start with installing CISST-SAW library:

1). Find 1**. Building using catkin build tools for ROS, open a terminal via"ctrl+alt+t" and copy following codes from the webpage.**

Remember to replace the "kinetic" by "noetic" which is our ROS version.

2). Then create catkin workspace :

3).

4). Use code:" gedit ~/.bashrc" to open that file and copy these codes to the end of that file.

Save that file and back to our terminal.

5). this part is optional, it's not important for our project.

6).skip 2.1, run two examples from 2.2:

Now, you are all set!, try our codes on section 3 to go further!

2. Section2: (Optional) On PC

3. Section3： How to run and call GUI on ROS:

1). First, open the terminal on desktop via ctrl+alt+t (or mouse right click, but not recommended).

2). Right click on the terminal and "Split Vertically".

3). Type in following command below, so that you should run and call the GUI of PSM3 for physical control.

```
roscore # on one side teminal. And copy following code on another terminal. Please use "crtl+c" to turn # off the roscore before logging out!!!!!
cd ~/ # return to root.
cd catkin_ws/src/cisst-saw/sawIntuitiveResearchKit/share/jhu-daVinci/ # Jump to the target folder.
rosrun dvrk_robot dvrk_console_json -j console-PSM3.json # Run and call the GUI
```

4). Then take following operation on GUI:

a. Click power on to power on the dVRM arm. **Please Remember click Power off or turn off GUI before you log out the system!!!**

b. Then click “Home”, “Disabled”, and “PSM3”, you will see:

c. Then click”Select tool type” choose 400049 as instance, for other tools, you can find their code on its shell. You will see a Warning, just click ”Yes”.

d. Click ”Direct control”, you will see:

E. Now it’s all set, you can do physical control on PSM3 now! Try to change the values of joints and read their values.

F. To quit the GUI, you can use “ctrl+c ” on terminal and remember quit roscore on terminal via”ctrl+c” as well.

4. Section4: How to physically control dVRK arm joint

1) First follow the steps in section3, then create a new python file in the dvrk python folder, paste the following code into the file, and finally use the command in terminal

rosrun dvrk_python <python-name>.py <arm-name>

2)

```
#!/usr/bin/env python
import dvrk
import math
import sys
import time
import rospy
import numpy
import PyKDL
import argparse

# print with node id
def print_id(message):
    print('%s -> %s' % (rospy.get_caller_id(), message))

# example of application using arm.py
class example_application:

    # configuration
    def configure(self, robot_name, expected_interval):
        print_id('configuring dvrk_arm_test for %s' % robot_name)
        self.expected_interval = expected_interval
        self.arm = dvrk.arm(arm_name = robot_name,
                            expected_interval = expected_interval)

    # homing example
    def home(self):
```

```
        print_id('starting enable')
        if not self.arm.enable(10):
            sys.exit('failed to enable within 10 seconds')
        print_id('starting home')
        if not self.arm.home(10):
            sys.exit('failed to home within 10 seconds')
        # get current joints just to set size
        print_id('move to starting position')
        goal = numpy.copy(self.arm.setpoint_jp())
        # go to zero position, for PSM and ECM make sure 3rd joint is past cannula
        goal.fill(0)
        if ((self.arm.name() == 'PSM1') or (self.arm.name() == 'PSM2')
            or (self.arm.name() == 'PSM3') or (self.arm.name() == 'ECM')):
            goal[2] = 0.189597
            goal[0] = math.radians(-14.335)
            goal[1] = math.radians(-1.478)
            goal[3] = math.radians(-19)
            goal[4] = 0
            goal[5] = 0
        # move and wait
        print_id('moving to starting position')
        self.arm.move_jp(goal).wait()
        # try to move again to make sure waiting is working fine, i.e. not blocking
        print_id('testing move to current position')
        move_handle = self.arm.move_jp(goal)
        time.sleep(1.0) # add some artificial latency on this side
        move_handle.wait()
        print_id('home complete')

    # get methods
    def run_get(self):
        [p, v, e, t] = self.arm.measured_js()
        d = self.arm.measured_jp()
        [d, t] = self.arm.measured_jp(extra = True)
        d = self.arm.measured_jv()
        [d, t] = self.arm.measured_jv(extra = True)
        d = self.arm.measured_jf()
        [d, t] = self.arm.measured_jf(extra = True)
        d = self.arm.measured_cp()
        [d, t] = self.arm.measured_cp(extra = True)
        d = self.arm.local.measured_cp()
        [d, t] = self.arm.local.measured_cp(extra = True)
        d = self.arm.measured_cv()
        [d, t] = self.arm.measured_cv(extra = True)
```

```
        d = self.arm.body.measured_cf()
        [d, t] = self.arm.body.measured_cf(extra = True)
        d = self.arm.spatial.measured_cf()
        [d, t] = self.arm.spatial.measured_cf(extra = True)

        [p, v, e, t] = self.arm.setpoint_js()
        d = self.arm.setpoint_jp()
        [d, t] = self.arm.setpoint_jp(extra = True)
        d = self.arm.setpoint_jv()
        [d, t] = self.arm.setpoint_jv(extra = True)
        d = self.arm.setpoint_jf()
        [d, t] = self.arm.setpoint_jf(extra = True)
        d = self.arm.setpoint_cp()
        [d, t] = self.arm.setpoint_cp(extra = True)
        d = self.arm.local.setpoint_cp()
        [d, t] = self.arm.local.setpoint_cp(extra = True)

# direct joint control example
def run_servo_jp(self):
        L = 0.0595
        J2 = []
        print_id('starting servo_jp')
        # get current position
        initial_joint_position = numpy.copy(self.arm.setpoint_jp())
        print_id('testing direct joint position for 2 joints out of %i' % initial_joint_position.size)
        def equation (a, l,x):
            return l*(1- np.cos(a))/a-x
        initial_guess = 1
        x = 0.03
        a_solution   = fsolve(equation, initial_guess, args = (L,x))
        print("Solution for a", a_solution)
        # amplitude = math.radians(40) # +/- 5 degrees
        amplitude = a_solution/2
        duration = 5   # seconds
        samples = duration / self.expected_interval
        # create a new goal starting with current position
        goal = numpy.copy(initial_joint_position)
        start = rospy.Time.now()
        for i in range(int(samples*0.5)):
            J4 = amplitude *   (1.0 - math.cos((i+1) * math.radians(360.0) / samples))
            # print(J4)
            goal[4] = (initial_joint_position[4] + J4)
            J2 = L- L * np.sin(J4) / J4
            # print(J2)
```

```
        goal[2] = (initial_joint_position[2] + J2 )
        # goal[2] = 0.1600000000165212
        # D = initial_joint_position[2] + J2
        # x = L * math.cos(0)*(1-math.cos(J4))/J4
        # print(x)
        self.arm.servo_jp(goal)
        rospy.sleep(self.expected_interval)
    actual_duration = rospy.Time.now() - start
    # self.arm.servo_jp(initial_joint_position)
    rospy.sleep(self.expected_interval)
    print_id('servo_jp complete in %2.2f seconds (expected %2.2f)' % (actual_duration.to_sec(), duration))

    initial_joint_position = numpy.copy(self.arm.setpoint_jp())
    print_id('testing direct joint position for 2 joints out of %i' % initial_joint_position.size)
    def equation (a, l,x):
        return l*(1- np.cos(a))/a-x
    initial_guess = 1
    x = 0.02
    a_solution  = fsolve(equation, initial_guess, args = (L,x))
    print("Solution for a", a_solution)
    # amplitude = math.radians(40) # +/- 5 degrees
    amplitude = a_solution/2
    duration = 5   # seconds
    rotation = math.radians(90)
    samples = duration / self.expected_interval
    # create a new goal starting with current position
    goal = numpy.copy(initial_joint_position)
    start = rospy.Time.now()
    for i in range(int(samples*0.5)):
        J4 = amplitude *   (1.0 - math.cos((i+1) * math.radians(360.0) / samples))
        goal[3] = (initial_joint_position[3] - rotation *   (1.0 - math.cos(i * math.radians(360.0) / samples)))
        self.arm.servo_jp(goal)
        rospy.sleep(self.expected_interval)
    actual_duration = rospy.Time.now() - start
    # self.arm.servo_jp(initial_joint_position)
    rospy.sleep(self.expected_interval)
    print_id('servo_jp complete in %2.2f seconds (expected %2.2f)' % (actual_duration.to_sec(), duration))

    initial_joint_position = numpy.copy(self.arm.setpoint_jp())
    print_id('testing direct joint position for 2 joints out of %i' % initial_joint_position.size)
    def equation (a, l,x):
        return l*(1- np.cos(a))/a-x
    initial_guess = 1
    x = 0.015
```

```
a_solution   = fsolve(equation, initial_guess, args = (L,x))
print("Solution for a", a_solution)
# amplitude = math.radians(40) # +/- 5 degrees
amplitude = a_solution/2
duration = 5   # seconds
rotation = -math.radians(45)
samples = duration / self.expected_interval
# create a new goal starting with current position
goal = numpy.copy(initial_joint_position)
start = rospy.Time.now()
for i in range(int(samples*0.5)):
    J4 = amplitude *   (1.0 - math.cos((i+1) * math.radians(360.0)  / samples))
    # print(J4)
    goal[4] = (initial_joint_position[4]  - J4)
    J2 = L- L * np.sin(J4) / J4
    # print(J2)
    goal[2] = (initial_joint_position[2]  - 1.1*J2 )
    goal[3] = (initial_joint_position[3]  - rotation *   (1.0 - math.cos(i * math.radians(360.0)  / samples)))
    # goal[2] = 0.1600000000165212
    # D = initial_joint_position[2] + J2
    # x = L * math.cos(0)*(1-math.cos(J4))/J4
    # print(x)
    self.arm.servo_jp(goal)
    rospy.sleep(self.expected_interval)
actual_duration = rospy.Time.now() - start
# self.arm.servo_jp(initial_joint_position)
rospy.sleep(self.expected_interval)
print_id('servo_jp complete  in %2.2f seconds (expected  %2.2f)' % (actual_duration.to_sec(),  duration))

initial_joint_position  = numpy.copy(self.arm.setpoint_jp())
print_id('testing direct joint position for 2 joints out of %i' % initial_joint_position.size)
def equation (a, l,x):
    return l*(1- np.cos(a))/a-x
initial_guess = 1
x = 0.02
a_solution   = fsolve(equation, initial_guess, args = (L,x))
print("Solution for a", a_solution)
# amplitude = math.radians(40) # +/- 5 degrees
amplitude = a_solution/2
duration = 5   # seconds
rotation = math.radians(90)
samples = duration / self.expected_interval
# create a new goal starting with current position
goal = numpy.copy(initial_joint_position)
```

```
            start = rospy.Time.now()
            for i in range(int(samples*0.5)):
                J4 = amplitude *   (1.0 - math.cos((i+1) * math.radians(360.0) / samples))
                goal[3] = (initial_joint_position[3] - rotation *   (1.0 - math.cos(i * math.radians(360.0) / samples)))
                self.arm.servo_jp(goal)
                rospy.sleep(self.expected_interval)
            actual_duration = rospy.Time.now() - start
            # self.arm.servo_jp(initial_joint_position)
            rospy.sleep(self.expected_interval)
            print_id('servo_jp complete in %2.2f seconds (expected %2.2f)' % (actual_duration.to_sec(), duration))


        # goal joint control example
        def run_move_jp(self):
            print_id('starting move_jp')
            # get current position
            initial_joint_position = numpy.copy(self.arm.setpoint_jp())
            print_id('testing goal joint position for 2 joints out of %i' % initial_joint_position.size)
            amplitude = math.radians(10.0)
            # create a new goal starting with current position
            goal = numpy.copy(initial_joint_position)
            # first motion
            goal[0] = initial_joint_position[0] + amplitude
            goal[1] = initial_joint_position[1] - amplitude
            self.arm.move_jp(goal).wait()
            # second motion
            goal[0] = initial_joint_position[0] - amplitude
            goal[1] = initial_joint_position[1] + amplitude
            self.arm.move_jp(goal).wait()
            # back to initial position
            self.arm.move_jp(initial_joint_position).wait()
            print_id('move_jp complete')
        # main method
        def run(self):
            self.home()
            self.run_get()
            self.run_servo_jp()
             self.run_move_jp()
    self.home()
    if __name__ == '__main__':
        # ros init node so we can use default ros arguments (e.g. __ns:= for namespace)
        rospy.init_node('dvrk_arm_test', anonymous=True)
        # strip ros arguments
        argv = rospy.myargv(argv=sys.argv)
```

```
    # parse arguments
    parser = argparse.ArgumentParser()
    parser.add_argument('-a', '--arm', type=str, required=True,
                        choices=['ECM', 'MTML', 'MTMR', 'PSM1', 'PSM2', 'PSM3'],
                        help = 'arm name corresponding to ROS topics without namespace.  Use __ns:= to specify the
namespace')
    parser.add_argument('-i', '--interval', type=float, default=0.01,
                        help = 'expected interval in seconds between messages sent by the device')
    args = parser.parse_args(argv[1:]) # skip argv[0], script name

    application = example_application()
    application.configure(args.arm, args.interval)
    application.run()
```