COMS W4901 Projects in Computer Science
Professor Peter Allen
Fall 2012

Project Report

Columbia Hand

Hand Team

Ji Wang

Columbia University
Department of Computer Science
12/31/12

ABSTRACT

For Columbia Hand, the structure has been improved and the measurement system has been tested. The hand has been rewired and the sensors are fixed. The position of the hand has been rearranged for grasping a smaller cylinder. Fingerpads have been put on the hand to protect the sensor and increase the friction.

For Harvard Hand, measurement system has been tested. The hand mount has been designed and manufactured. The wires have been fixed and organized. The sensor signals of the other three fingers have been sampled and processed. The circuits of sensor signal sampling and filtering have been created. An average software filter has been designed. Two dynamic force sensors have been fixed. A reactive control algorithm has been designed and modules like close_hand.py, arm_move.py and listener_talker.py have been developed.

TABLE OF CONTENTS

1. INTRODUCTION

The old Columbia Hand could open and close freely and some of the sensors worked. The hand could hold some big bottles but failed to grasp smaller bottles due to the unappropriate position of the fingers. Some piezo-resistive films and wires were also broken. To improve the hand, the ideas of rearranging the position of the fingers and putting fingerpads on the sensors was provided. Also, wires and sensors were fixed or replaced.

Before I worked on Harvard hand, it was able to open and close and the signal of sensors on one finger was sampled. However, the range of hand closing and opening were limited due to the motor control program. Also, the signals of the other three fingers were not sampled and the circuits of signal sampling were not working perfectly because one amplifier was bad. The filters of the signals were not designed so that the signals were not decent for grasping experiment. Two of the sensors on the fingers were not as sensitive as others. A mount to Staubli Arm was also necessary for graping experiment. A feed-forward control algorithm and reactive control algorithm were needed. The program for running ROS across two computers were also a must.

2. APPARATUS

ROS was used to realized communication between two computers. C/C++ and Python were used in programming. Staubli Arm was used for grasping experiment. Two OP485 chips were used for signal sampling and filtering. PRO-ENGINEERING/Creo 5.0 was used to design the mount of Harvard Hand. Modules such as HaoUtil.py and Move_Hand were also used to develop the reactive control algorithm. Multimeter was used to test the sensor and wires. Oscillograph was used to test the signal of sensors.

3. APPROACH

To acquire decent sensor signal from Harvard Hand, circuits with amplifiers and first order RC filter were created. Also, an average software filter were designed to reduce the noise. To realize the reactive control algorithm, firstly, communication between two computers were needed. Secondly, Hao's program was necessary. Thirdly, knowledge of manipulation of the Staubli Arm was also needed. Finally, modules like close_hand.py, listener and talker were also a must.

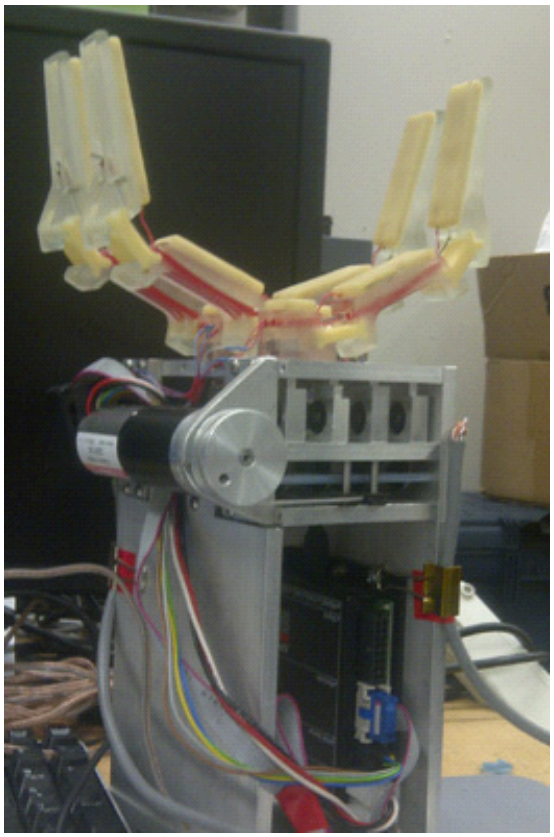
4. RESULTS

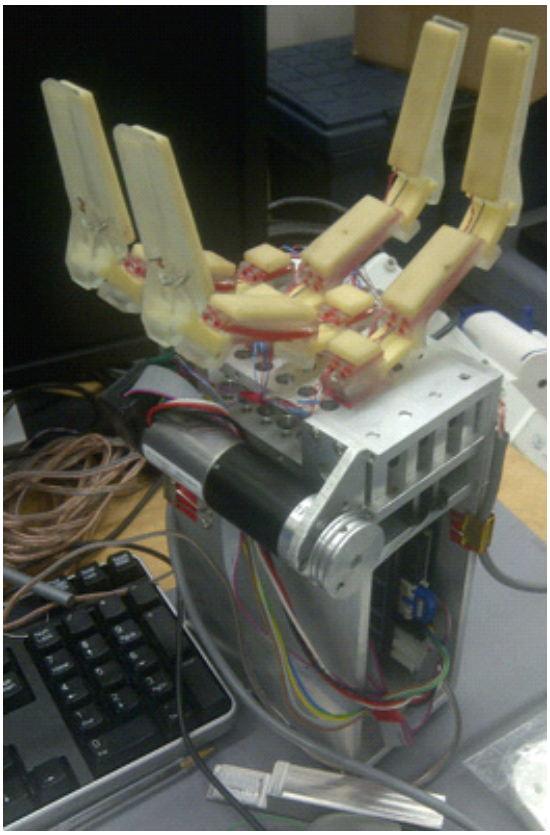
4.1 Columbia Hand's Wires and Sensors

I wired the old Columbia Hand and fixed the piezo-resistive sensors. Also, I checked the signal sampling device(NI DAQ board) and guaranteed all the sensors and potentiometers worked well. I also assisted Kyle with the mechanism design and provided the ideas of finger position rearrangement and fingerpads.

4.2 Harvard Hand's Mount

To put Harvard Hand onto Staubli Arm, I designed a mount. The mount is a holder for motor controller and wires as well. The material of the mount is aluminum and the shape is like the letter 'U'. The mount can easily afford the hand weight, which is 5.5 pounds. The process of manufacture includes cutting, drilling and tapping.



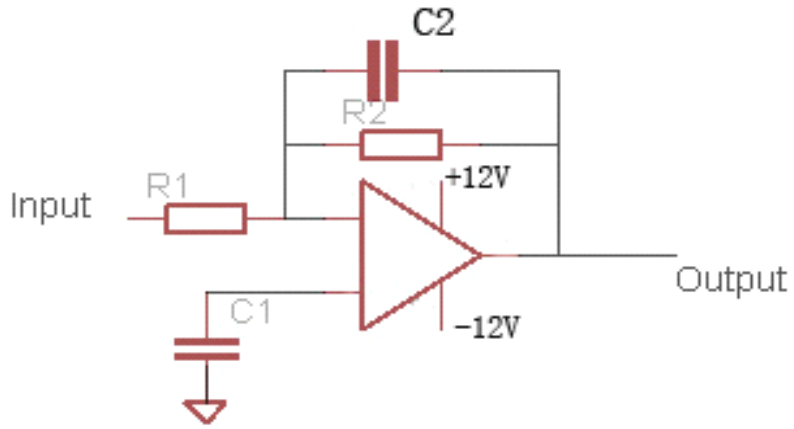


4.3 Harvard Hand Signal Sampling and Processing

NI DAQ board is used for the signal sampling. I added 6 more ports for the other 3 fingers. The relationship between the ports and sensors are listed in the following table. Circuits were created to realize input and output impedance matching to finally acquire stable signals.

Sensors	Ports	Sensors	Ports
Sensor 1A	66	Sensor 1B	31
Sensor 2A	63	Sensor 2B	61
Sensor 3A	26	Sensor 3B	58
Sensor 4A	66 (on DAQ 2)	Sensor 4B	23

To acquire decent signals, both hardware and software filters are designed and applied. The hardware filter is a RC circuits filter attached on the amplifier. It is an active first-order low-pass RC filter, whose cutoff frequency is around 80 Hz. This filter helps reduce a lot of noise and greatly improve the signal to noise ratio.



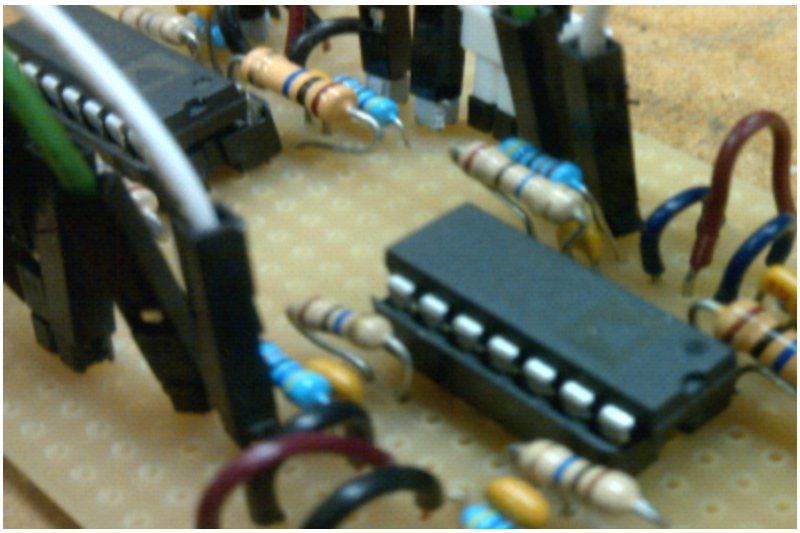


Fig.1 First-order RC filter circuits

For the software part, I tried both average filter and median filter and finally found average filter produced a more stable signal. Finally, the signal to noise ratio is about 10:1, which is high enough to tell the behaviour of the sensors. Below is the filtered signal of sensors from a random grasp.

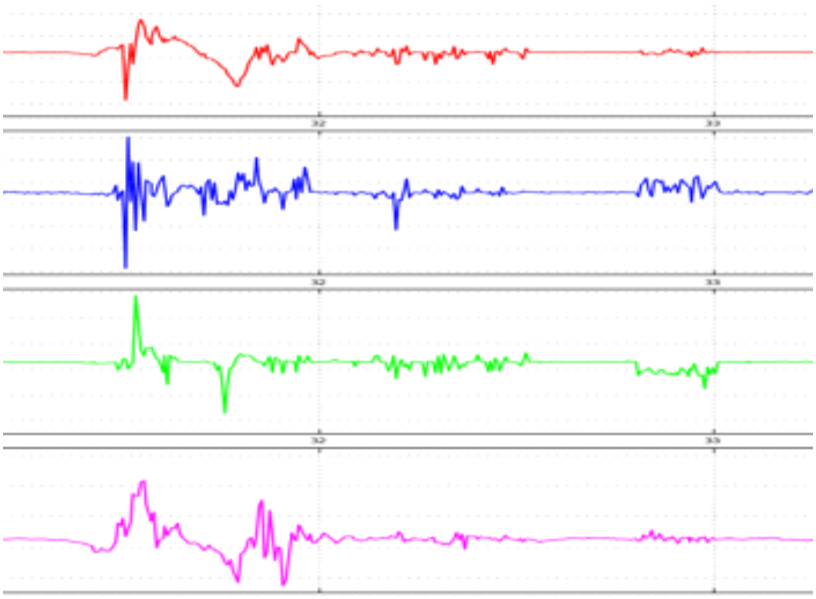
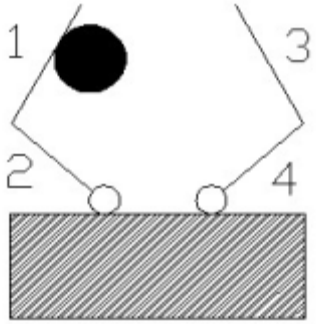
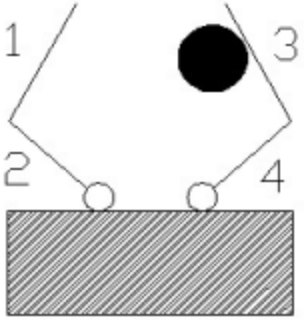


Fig.2 The signal of a grasping process of Harvard Hand

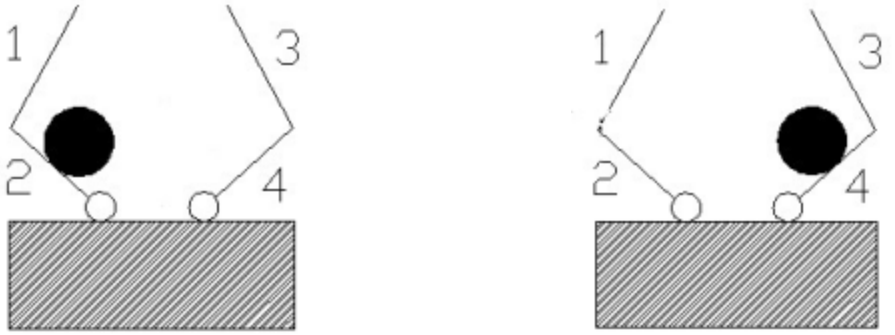
4.4 Control Algorithm Development

The control algorithm I am developing for Harvard Hand is reactive control. The idea of the control method basically came from Professor Howe's paper. Reactive control allows the hand to behave very quickly according to the change of the environment. The hand will react differently according to the signals of sensors on each finger during grasping. The sensitivity of the dynamic piezoelectric films is extremely important. Below are the possibilities of the contact between hand and the object.

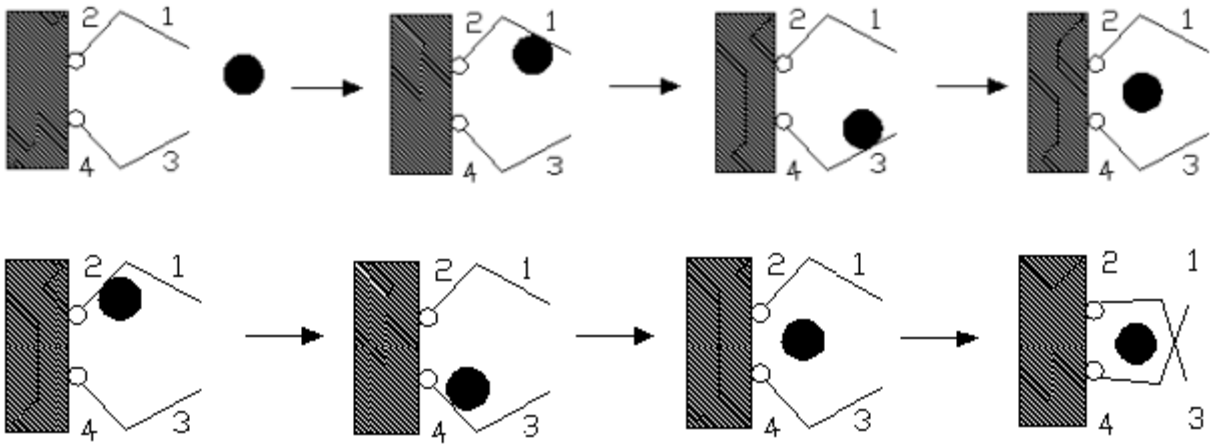




The situation above is named Outer Contact by Professor Howe. In this situation, the object is off-center; For a good grasp, the hand should move left or right according to Senor 1 and Sensor 2.



This is called Inner Contact in Professor Howe's paper. For this situation, the hand should move a little bit back-forward to make the object stay at the center of the palm. The picture below shows how the hand is going to grasp the object.



In a word, the reactive control algorithm is based on the contact situation, which are outer contact and inner contact. Below is the basic idea of the program I am writing.

begin

while no traveling limitations

```
do
{
Hand advances
}

if sensor3 is touched
{
 Hand moves down 6 cm
   if sensor1 is touched
     {
      Hand moves up 3cm
     }
   else motion_error
   Hand moves forward 3cm
   Hand moves up 3cm
   if sensor4 is touched
     {
      Hand moves down 3cm
        if sensor2 is touched
           {
            Hand moves up 3cm
            Hand moves backward 2cm
            Close_Hand
           }
     }
   else motion_error
}

else if sensor1 is touched
{
 Hand moves up 6 cm
   if sensor3 is touched
     {
      Hand moves down 3cm
     }
   else motion_error
   Hand moves forward 3cm
   Hand moves up 3cm
   if sensor4 is touched
     {
      Hand moves down 3cm
        if sensor2 is touched
           {
            Hand moves up 3cm
            Hand moves backward 2cm
            Close_Hand
```

```
        }
    }
  else motion_error
}

else if sensor4 is touched
{
     Hand moves down 3cm
      if sensor2 is touched
        {
         Hand moves up 3cm
         Hand moves backward 2cm
         Close_Hand
        }
      else motion_error
}

else if sensor2 is touched
{
     Hand moves up 3cm
      if sensor2 is touched
        {
         Hand moves up 3cm
         Hand moves backward 2cm
         Close_Hand
        }
      else motion_error
}

else motion_error

end
```

The hand and the arm are controlled by two computers, so communication between two computers is necessary. I chose ROS as the stage of control and communication for the arm and hand. I have developed the functions such as close_hand.py, talker.py and listener.py. Now, I am trying to put the functions together to realize the algorithm.

5. CONCLUSIONS

Feed-forward control algorithm has been realized. Modules for hand motion and signal processing were created. ROS services and clients were developed. Communication between two computers based on ROS was also worked out.

6. ACKNOWLEDGMENTS

I would like to acknowledge the assistance and instruction of Hao. He help me with the manipulation of Staubli Arm and showed me how to use the modules he had developed. Also need to thank Kyle, he helped me with some machining stuff.

7. REFERENCES

[1] Ample's document, including Hand_document and daq_labels_2011_11_04

[2] Aaron Dollar et al (2010) *Contact Sensor and Grasping Performance of Compliant Hand,*

8. APPENDICES

8.1 Programs

8.1.1. Average filter

```
for( int i = 0; i < 8; i++ )
{
comediData[i] = 0;
}
 while (ros::ok())
  {
for(int i = 0; i < storageAmount; i++){
for(int j = 0; j < 8;  j++){
   comedi_data_read(it,subdev,daqChannels[j],range,aref,&comediData[j]);
   sensorReading[j] = comediData[j];
   storageArray[i][j] = sensorReading[j];
}
}
for(int i = 0; i < 8; i++){
sensorSum[i] = 0;
for(int j = 0; j < storageAmount; j++){
```

```
sensorSum[i] += storageArray[j][i];
}
sensorAverage[i] = sensorSum[i]/storageAmount;
//printf("%d\n", sensorSum[i]);
  }
```

8.1.2 GuardedCloseHA_Hand

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('HA_HAND_1_2')
import rospy
from HA_HAND_1_2.srv import *
import time


def GuardedCloseHA_Hand(req1, req2=[], req3=[]):
    rospy.wait_for_service('EPOS_control1')
    try:
        EPOS_control1 = rospy.ServiceProxy('EPOS_control1', AD)
        EPOS_control1(req1, req2, req3)
        return 0
    except rospy.ServiceException, e:
        print "Service call failed: %s"%e


if __name__== "__main__":
        GuardedCloseHA_Hand(1,[1,1],[-20000,-20000])
        time.sleep(1)
        GuardedCloseHA_Hand(1,[1,1],[-0,-0])
```

8.1.3 Listener.py

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('HA_HAND_1_2')
import rospy
from numpy import *
import HA_HAND_1_2.msg

def callback(data):
    rospy.loginfo(rospy.get_name() + ": I heard %s", data.data)

def listener():
    rospy.init_node('listener', anonymous=True)
    rospy.Subscriber("chatter", HA_HAND_1_2.msg.SensorIntArray, callback)
    rospy.spin()

if __name__ == '__main__':
    listener()
```

8.1.4 Feed forward control algorithm

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('HA_HAND_1_2')
import rospy
from numpy import *
import HA_HAND_1_2.msg
from HA_HAND_1_2.srv import *
import time
import trajectory_planner as tp
from HaoUtil import *
import tf_conversions.posemath as pm
from std_msgs.msg import String
import HaoUtil as hu
import pdb

def GuardedCloseHA_Hand(req1, req2=[], req3=[]):
    rospy.wait_for_service('EPOS_control1')
    try:
        EPOS_control1 = rospy.ServiceProxy('EPOS_control1', AD)
        EPOS_control1(req1, req2, req3)
        return 0
    except rospy.ServiceException, e:
        print "Service call failed: %s"%e

def callback(data):
    #rospy.loginfo(rospy.get_name() + ": I heard %s", data.data)
    global current_value
    current_value = []
    current_value = data.data
    if current_value[0] > 34640:
        #GuardedCloseHA_Hand(1, [1,1], [-30000, -20000])
        #time.sleep(1.5)
     global_data = tp.SetupStaubliEnv(True)
    #GuardedMoveUntilHit(global_data, array([0,0,1]), 'PALM', 0.05, 20)
    MoveHand(global_data, array([0,0,1]), frame = 'PALM', distance = 0.05)
    time.sleep(2)

def listener():
    rospy.init_node('listener', anonymous=True)
    rospy.Subscriber("chatter", HA_HAND_1_2.msg.SensorIntArray, callback)
    rospy.spin()

if __name__ == '__main__':
    listener()
```

8.2 Guidelines for the next person

I don't know who you will be, but I need to tell you that your task is to finish the reactive control algorithm of Harvard Hand. If you have time, go on working on Columbia Hand. Of course, you can try other control algorithms. Below is the information which I think will be helpful to you, please read.

PART 1

If you have any questions on this part, you can check Ample's Hand_document file.

Steps for closing and opening hand:

1. Turn on the power of the circuits and controllers;
2. Check the input and output ports, make sure they are connected to the right sensors;
3. Check the motor controller, if it is on, the light (green) should flicker;
4. Start ROS, run the command below one by one:

   (1) roscore;

   (2) sudo chmod a+rw /dev/comedi0; //This enables permission of DAQ card

   (3) rosrun HA_HAND_1_2 eposServer1; //Initialize the motor controller, if successful, the green
   //light will stay still rather than flicker

   (4) rosrun HA_HAND_1_2 talker; // talker of the sensor signals

   (5) rxplot /chatter/data[0]; // See the graph of signals of the sensors

   (6) rosrun HA_HAND_1_2 client 1 0 500; // Set the velocity

   rosrun HA_HAND_1_2 client 1 1 -10000; Set the displacement by which the hand will move

PART 2

The steps above are signal sampling and hand motion, below are the steps for the feed-forward control of the hand and arm:

1. Make sure the 1,2,3 steps in PART 1 are done;
2. Turn on the power of the arm;

3. Make sure the two computers are in the same local area network;

4. Run roscore in the ROS of the computer attached to the arm;

5. Run the following commands in the ROS of the computer which is attached to the hand:

(1) export ROS_MASTER_URI=http://192.168.11.200:11311;

rosrun HA_HAND_1_2 eposServer1;

(2) export ROS_MASTER_URI=http://192.168.11.200:11311;

rosrun HA_HAND_1_2 talker;

(3) export ROS_MASTER_URI=http://192.168.11.200:11311;

rxplot /chatter/data[0];

6. Go to the other computer and run the following stuff to initialize the Staubli arm:

(1) roslaunch staubliTX60 staubli.launch

(2) roslaunch calibration launchKinectRobot.launch

(3) rosrun force_sensor_serial_port ForceTorqueServer // Force and Torque Sensors

(4) rosrun trajectory_planner grasp_message_robot_server.py

(5) rxplot /ForceTorque/Readings/force/x:y:z //graph of the force sensor signal

7. Move the hand by the panel:

(1) Turn on the panel and switch the mode to manual; If you can't turn it on, take off the panel from the box and put it back, then you will be OK;

(2) Press the joint button and use the x,y,z to move the hand to where you want;

(3) To move the arm back to initial position(zero position), press 'Menu', then choose 'calibration', then select any nonzero coordinate value, hold 'move/hold' button until it is zero.

8. Run the code of control algorithm, now I have feed-forward control algorithm:

rosrun HA_HAND_1_2 add.py