Robot Grasping with Optical Proximity Sensors

Min Young Kim, Dookun Park, and Jaewon Yang
Stanford University
{minykim, dkpark, crucis}@stanford.edu

1. Introduction

Grasping is a basic and important problem in robotic manipulation, and sensing the object geometry is important for reliable grasping action. In this project, we implement an object grasping system. Basically, we use WAM robot arm and Barrett robot hand for objects grasping. There are optical proximity sensors on each fingertips of the robot hand and we use the optical sensors for pre-touch pose estimation. There is an existing algorithm, but we use different learning methods to improve the object geometry recognition and finally enhance the performance of grasping action.

2. Data

We receive input data from 12 sensors attached on the fingertips. The robot has 3 fingers and each finger has 4 sensors. Input data consists of 20 sub values. The meaning of these values will be described in the following section. Outputs which we should estimate are the geometrical configuration of the surface of the object from each finger, distance and two angles which depict the tilt of the surface. Output consists of 3 values.

3. Regression Models

We measure $x \in R^{20}$ from four optical sensors mounted on the finger tips of STAIR2. First four readings are made without emission of light and then sensors emit by turns. When a sensor radiates, not only the emitting sensor but all the others measure light; this is called "crosstalk".

Given the sensor reading x, our job is to estimate $y \in R^3$ that consists of the distance from the object to the finger and two angles specifying the orientation of the object. We use a Bayesian polynomial regression model. High order polynomials is necessary in order to address nonlinearity between x and y, and a Bayesian approach would keep our model from over-fitting.

4. Robot Control

WAMArm(STAIR2) and BarrettHand

The robot we use is WAMArm (STAIR2) in A.I. lab. We also have Barrett hand for grasping simulation and the hand is attached on the end-effecter of STAIR2. Controllers for manipulating these robots are already implemented by Stanford A.I. lab. However, a force sensor has recently been added to STAIR2 and we needed to change configuration by figuring out the mass, the position of the center of mass and value of d parameter for DH table using 'btdiag' controller which is the old version. The values we figured out are as follows:

.

BarrettHand_*mass* = 1.4 (kg)
BarrettHand_*centerMass* = (0.0, 0.0, -0.04)
DH_PARAM_*d* = 0.12032
(Values are obtained by trial and error)

To get started with the project we needed to download and install ROS for manipulating STAIR2. There should be roughly 3GB spaces for installation both ROS and STAIR project and STAIR project consists of at least four programs-'newController', 'Relay Server Hack', 'XWam', and 'IRstreamer'.

After all installation, it is necessary to run both of WAMArm (STAIR2) and BarrettHand, and test the basic movement of them.

STAIR2 Manipulation

The operation system that we used is called ROS(Robot Operating System)[2]. Since the existing controls and related programs run on ROS, we needed to make sure if ROS and all ROS-related packages had been accurately configured. Though, in the middle of the project, ROS did not work so we had to completely re-install ROS.

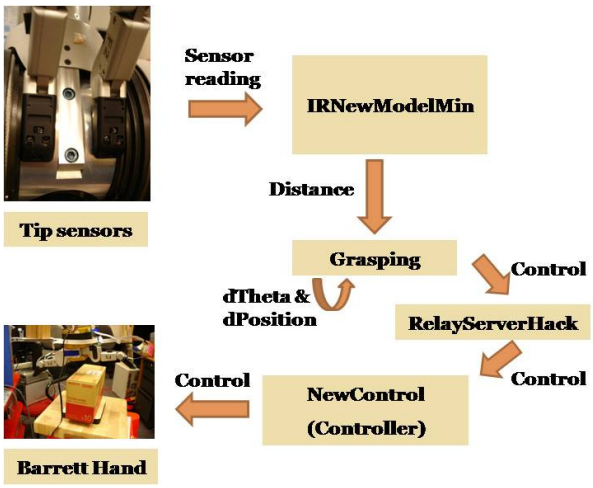

Figure 1. Process of STAIR2 manipulation

We manipulated BarrettHand with a controller called 'newControl' in WAM1 machine. There is another program called 'IRstreamer'. It takes serial input and sends the data to the controller. We used the 'IRstreamer' to collect sensors' reading, apply our model, and finally map the result data from the regression into certain variable which would be used for actual grasping. At last but not least, we used 'Grasping' program to run STAIR2.

However, since 'newControl' runs on WAM1 machine and the others run on 'smart-wing' machine, there's a need to use socket for communicating within machines. 'Relay-ServerHack' is used on WAM1 machine for these communications. All programs need to be executed in a right order.

Because there was no document for instructions to run these programs, we spent quite a big amount of time to make things right. We'd finally succeeded to run all programs and started to create a new version of IRstreamer for applying our regression model. The new program is called 'IRNewModelMin' and we did not use calibration parameters calculated by the previous researcher. After getting distances from the model, we mapped the distances to the desired value which would then be used for grasping. The desired values were ranged from 1 to 100. If the desired value is 1, it means the finger is located far from the object. On the other hand, '100' means that the finger is very close to the object. We used a linear mapping which maps 0.5 centimeters to value 100 and 3.5 centimeters to value 1. Desired values worked as a measurement while deciding future movement of each finger.

We basically used 'Grasping' program to simulate STAIR2, and tuned the program to

.

properly fit our model into STAIR2.

5. Results

Preliminary result with previous data

During the first a few weeks, we performed regression based on the dataset that Hsiao collected about a grey box during this summer. We could not gather a new training set because STAIR2 has been under reconfiguration.

We first verified the relevance of input features because in Hsia et al [1], the authors argued that all but three attributes of x are meaningless. As a quick measurement, we calculated the correlation coefficient of each x_i and y_j, and the probability that x_i and y_j have no correlation. Surprisingly, any x and y showed a correlation coefficient about 0.4, and the probability of no correlation was 0 in any case. Therefore, we decided to include all the features. We found suitable model and estimate a generalization error of our model by a cross-validation (CV) test.

Furthermore, we concluded that the previous datasets are not trustworthy. After we enable STAIR2 move again, we tested our model on a real gray box to verify the integrity of training data. However, this test showed more than 40% errors, which was abnormally higher than CV errors.

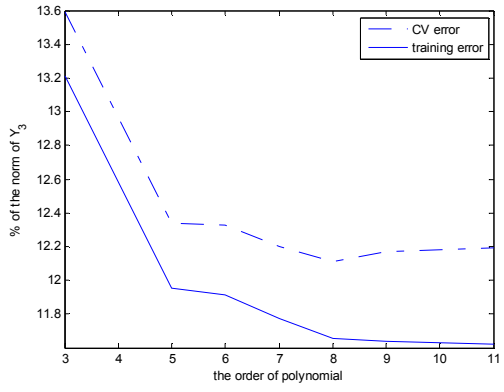

Figure 2. Errors vs. order of polynomial

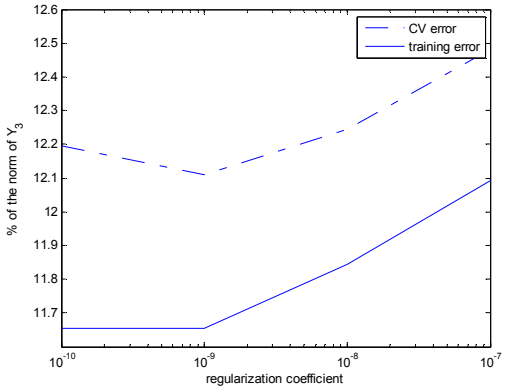

Figure 3. Errors vs. regularization parameter

Result with new training sets

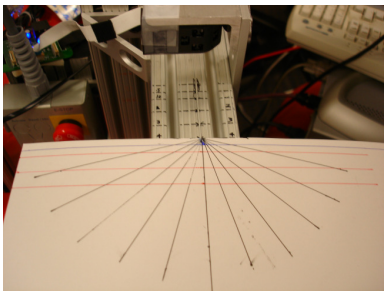

Figure 4. Measurement tool for data collection

We collected training data with a black box and a wooden box. We did not choose brighter objects because the sensors are saturated too often with bright objects. We varied the distance from 5mm to 40mm, and the angles from 30 to 120. Later, we included in the training set the case that the finger is normal to the surface because we observed that this case happens often in reality.

After gathering a training set, we performed regression, choosing model parameters by a CV test. We tested our model with various object. Our model estimates the distance within 20% errors when the surface has similar color and reflectivity compared to the objects over which we trained.

The shortcoming of current configuration is that it cannot detect the reflectivity of the surface, which is crucial in accurate estimation. Indeed, our model estimated distance shorter than the real value when we tested over bright or reflective objects. It turned out, however, that the relative, not absolute, distances between each finger and the surface are used in the control program of STAIR2. Therefore, we could handle this shortcoming by choosing 5th polynomial model that showed monotone decrease between sensor readings and the estimates.

Grasping test for several objects

(a) Wood Cube

(b) Adapter

(c) Paper box

(d) Green bowl

Figure5. Tested objects

All Objects used for data collection and some other objects are used in simulation. A black box, laptop adapter, and wood cube are the objects used for data and BarrettHand resulted in grasping these objects with accurate probing. A green bowl, blue-stripped white bowl, and brown paper box were the new objects and BarrettHand grasped green bowl and paper box with high accuracy but BarrettHand failed to grasp blue-stripped with bowl. It is convincing that the reason would be at reflection rate on the surface of the object. When the reflection rate is too high, such as white color, sensor values are too high to result in feasible data. Overall grasping rate of our model for those different objects was 82.35% (28 succeeded out of 34 trials).

6. Discussion

Having succeeded to grasp objects is noticeable result, though there's some shortcoming with BarrettHand itself and objects to be grasped. One thing is that due to limited freedom in finger movements, some objects are very hard to grasp.

With the current configuration of sensors, a regression model cannot estimate the distance with high accuracy for different kinds of surfaces. Given sensor readings at one point, it is impossible to determine the reflectivity of the surface. We will be addressed this shortcoming by combining sensor readings from various points. For example, we can watch how much sensor readings increase as fingers approaches to the object, and determine the reflectivity based on the observations.

One of the alternative solutions is to use different type of sensors whose signal relatively independent from reflection rates of the surface, then we can get much better results when testing with unknown objects.

Issues occurred during the project and suggestions

i. Network in Gates building sometimes failed. Since the overall system is based on many programs running on different computers, network failure definitely had slowed down the project.

.

ii. Finger tip parts can be easily detached from the hand with a small screw driver and it was convenient to do that when collecting data.

iii. When proximity sensor values seem to be weird, it is highly recommended to reset the sensor power.

iv. The 20th values of 20 inputs of the finer two(F2) seems not reliable.

v. When the robot arm of STAIR2 seems to be weird and the joint movement is not smooth, it is probably the problem of WAM1, not of the arm. It is recommended to restart WAM1 and try your job again.

vi. Since 'btdiag' program was initially designed for the basic diagnosis of the robot arm, it is one of the most stable program for the robot arm. Running 'btdiag' program can be a good start point if the robot have stopped urgently.

vii. It is highly recommended not to assume anything would be naturally working. What you really assumed fine can be a key point of your problem.

7. References

[1] K. Hsiao, P. Nangeroni, M. Huber, A. Saxena, and A. Y. Ng. Grasping Using Optical Proximity Sensors.

[2] Robot Operating System Wiki
http://pr.willowgarage.com/wiki/ROS

.