Grabbing Objects with the NAO Robot Using Multiple Behaviors and Interval Estimation for their Selection

Christof Oost, University of Groningen, Bachelor Artificial Intelligence

Abstract. When using a robot in the real world the ability to manipulate objects is important, but the question is how to grab the objects. In this research it is shown that it is possible to do this with multiple behaviors. Each grab attempt leads to a *success* or *failure*. The best behavior is selected with interval estimation. The interval is calculated with the Adjusted Wald formula. Six behaviors are created. Also nine objects are collected. Each behavior-object combination is tested 6 times, collecting data and bootstrapping the robot on these objects. The interval was firmly reduced after 6 trials and the confidence interval moved towards the expected average. Seven of the nine objects had a behavior that could grab them in all the trials. While there is still a lot to research in successful grabbing, this research is a step in that direction.

Keywords: Binomial, Interval Estimation, behaviors, grabbing multiple objects, Adjusted Wald, NAO robot.

1 Introduction

Whether people are getting old, are disabled, heavy or repetitive work needs to be done or there is simply no time for the (regular) tasks in and around the house a personal attendant can be a great help. The ideal attendant for the job would be a robot. It never gets bored, is always there and will always deliver the same quality of work or better.

To be useful, most robots need to manipulate the world around them to perform their tasks. Simple manipulation can sometimes be done by driving against the object to move it. However most manipulations need a more complex operation. A good manner to do this, is using an arm and gripper combination. Humans are the perfect example with their arms and hands that can grab and manipulate various objects. This combination is especially ideal for objects and spaces that are designed for human manipulation.

Robocop@home[1] [1] competition is meant to stimulate the development of service robots. For this reason the environments are created to replicate a human home as good as possible. Also the tasks are similar to what a human servant should be able to do, like getting an object. Since the world is dynamic, always changing, this is also embedded in the competition. The objects are never placed in exactly the same place and there are also humans in the environment.

[1] www.robocupathome.org/

The University of Groningen competes with the BORG team[2] [2] in this Robocop@home competition. The basis of their robot is a Pioneer 2-DX Robot[3]. On top of the Pioneer there is a frame that contains laptops for processing power, sensors and a NAO robot[4] [3]. The NAO is for the interaction part and object grabbing. The rest of the robot ensures fast movement, processing power and a frame to mount additional sensors.

Central in this paper is the question of how to grab different kinds of objects that are positioned in front of the NAO. Since there is not yet a vision module that returns useful information, manual feedback is used.

1.1 Grippers and Hands

The goal of this research is to grab multiple objects with the NAO robot, but how can this be done? The manner in which a robot can grab depends on the shape, size and power of the gripper. It also depends on the limitations of the arm. The shape and power of the gripper dictate how the interaction can be between gripper and the object. The arm has to move the gripper into the required position. Without the movement of the rest of the robot not all gripper positions are possible from the robots location.

A fist impression of the NAO hands gives the idea that they are very similar to human hands. Unfortunately there are some differences. The first is the control of the fingers. The three fingers in one hand move synchronous. Each finger is operated by one wire, which implies that the individual parts of the finger cannot move separately from the other parts. There are only three of the five fingers (including a thumb) on each hand. Also the location of the thumb is different and it has less flexibility.

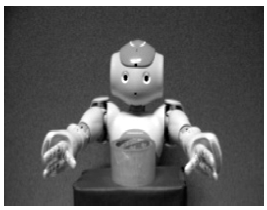

Fig. 1. The NAO robot and an object (tape) that it might encounter as service robot

The size of the NAO, 58 cm high, makes it a miniature human. This has implications of how objects can be grabbed, since objects that are normal for humans can be huge and heavy for the NAO. For instance, scotch box sealing tape is quite big if compared to the NAO, fig.1. Therefore it will be impossible for the NAO to mimic all human grabbing behavior and another way of grabbing / learning how to grab is needed.

The hands of a NAO can be seen as two fingered grabbers, since the fingers on each side move as one, and as multiple fingered grippers since the fingers are flexible. Some papers on grabbing objects focus on grabbing one specific object or use very complex methods. One of these complex methods is shown in [4]. They use touch sensors, image processing and a data base. In this data base characteristics of each object have to be added by a human before the robot can grab. But the learning of new objects cannot be done autonomous and the NAO is lacking these sensors.

---

[2] http://www.ai.rug.nl/crl/

[3] http://www.mobilerobots.com/ResearchRobots/ResearchRobots/PioneerP3DX.aspx

[4] http://www.aldebaran-robotics.com/

1.2 General Task Description

The initial research question is: *how to grab multiple objects with the NAO*. The mentioned objects are objects that can be found in a house. This means that they differ in shape, size, material and texture. The amount of objects that fit this description is huge. Not all of these objects can be known in advance. Since the manner of how to grab an object differs, it is essential that the robot is able to learn how to grab new objects. This learning must be fast since the robot operates in the real world and a user does not want to wait for a long time.

Therefore the research question is: *how to grab different objects with the NAO robot while learning time is short and new objects can be encountered*. One technique is searching for the technique with the highest overall performance. While this leads to a certain overall optimum, this doesn't need to be the best individual result. When new situations (objects) are encountered, the technique has to be reevaluated. This process gets more and more expensive in time and resources when the data set grows. Better results should be possible.

An alternative is suggested in [5]. The idea is to create multiple techniques (called behaviors in this case) and selecting the best behavior with interval estimation (IE). If it is needed a new behavior can be added. The IE is done based on the success ratio of a behavior on the object that needs to be grabbed.

There are a number of advantages. There is no need for calculating the best overall grab approach. That saves calculation time, saves processing power and the performance of known objects never drops. The performance can only improve, over time or when new behaviors are added. This adding of new behaviors is easy. Because of these advantages this idea of multiple behaviors and IE is used.

2 Methods

This chapter starts with the selection of objects and the description of a number of possible grab behaviors. Thereafter it is described how the best behavior can be selected with interval estimation, how the algorithm and the program work and the test setup.

2.1 Objects and Grabbing

The selected objects are inspired by the robocup@home rulebook from 2010 and are objects that are found in a home. While the ultimate goal is to grab objects that a human can grab or more, the robot has some limitations. Since the NAO is much smaller, the maximum weight that it can lift is 300 grams and the maximum size of objects is far smaller than what a human can grab.

The chosen objects represent a number of different shapes, materials and sizes as presented in Fig 1. They are described in table 1.

Fig. 1. The 9 objects, from left to right, up to down: bottleP, cup, dessertCup, Pringles, can, tape, ball, sponge, box.

Table 1. Describing the objects

Name	Description	Length &width (cm)	Height (cm)	Weight (grams)
bottleP	Empty, 0,5 liter Pepsi bottle	Ø 6	23	28
Cup	2 stacked coffee cups, 175ml	Ø 4,5-7[5]	8,5	8
dessertCup	Cup for chocolate mousse, empty	Ø 6,5– 9,5[5]	5	8
Pringles	Can for Pringles, size: 40 grams, empty	Ø 8	8,5	23
Can	Can for coca cola, 330 ml, empty	Ø 6,5	11,5	25
Tape	Roll of tape, not empty	Ø 9.5	5	109
Ball	Tennis ball	Ø 6,5	6,5	57
Sponge	Cleaning sponge, dry	6,5 X 9	3	10
Box	Top part of a game box	18 * 11	7	46

2.2 Grab Methods

In chapter 1 there is concluded that the NAO cannot directly copy human grabbing behavior. Reasons are weight of objects, limitations in movement and the far smaller size of the NAO. It also concluded that the best way to proceed is to create multiple behaviors for grabbing, learn them and select the best option.

Six grabbing behaviors are created and a start position. More behaviors can be added at any time. This is done with a program that comes with the robot (Choregraphe). The designer can select the parts necessary for the action and then record their movement. Subsequently it is possible to make adjustments to improve the

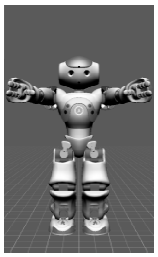

Fig.2. Start position

[5] The two sizes are from the smallest and biggest diameter from the object

behavior, like *smoothen* the movement and change final motor values. The data is stored in a xml file that the robot can execute.

Description of Behaviors and Object Positions.

The start position (fig. 2.) is created to prevent the arms of hitting the table or object if the robot approaches the table or when it has to move to the first position of the behavior. The arms are stretched and the hands opened. The hands stay open during all grabbing behaviors. The behaviors are called grab0 to grab5. The end positions can be seen in fig 2 to 6

Grab0 moves the hands together trying to hold the objects between the fingers, the movement of the arms is symmetrical. The object should be located directly in front of the robot (in the center).

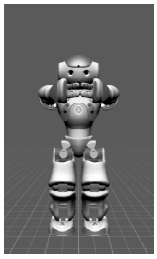

Fig. 3. Grab0

Grab1 & grab2. One of the arms stays in the starting position while the other arm sweeps the object, in the horizontal plane, towards the chest. When it goes well the object is pinned between the sweeping arm and the chest. In grab1 the left arm moves, in grab2 the right. The object should be located directly in front of the robot or in the sweep path.

Grab3 & grab4. The idea was to keep the arms horizontal and grab the object between the two lower arms. Unfortunately this was not possible so a variation was made. The arms both move towards the chest, but one of the arms is above the other. The object get pinned between the chest and the arms. In grab3 the left arm is on top and in grab4 the right. The object should be located directly in front of the robot.

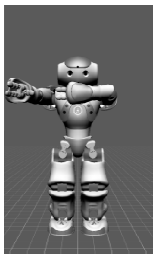

Fig. 4. grab1 & mirrored grab2

Grab5 gets the arms under an object and lifts it up from below. This could work for objects like bowls. The hands move symmetrical. The object should be located directly in front of the robot.

The two variations Grab2 and grab4, are not an exact mirrored copies from grab1 respectively grab3. The small differences will probably also lead to different performances in grabbing.

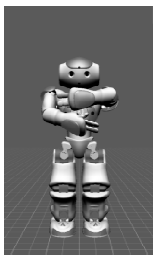

Fig. 5. grab3 &mirrored grab4

2.3 Learning How To Grab

With these 6 behaviors it is the question how to optimize the chance of a successful grab. The robot needs to select the best possible behavior to grab the specified object, while having only limited or no information about the success of the behaviors. Also the learning time is short since the grabbing is needed to fulfill other tasks. This means that the robot cannot try every behavior 30 times or more per object.

To solve this problem of learning and selecting the expected best behavior, interval estimation is used. The upper bound is calculated per behavior, based on the grab experience that the behavior had on the object that needs to be grabbed. The behavior with the highest

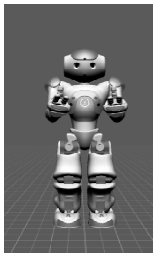

Fig. 6. grab5

upper bound is possibly the best behavior available: it is the most optimistic value. The feedback of the grab will reinforce the confidence interval: it gets narrower and the calculated midpoint shifts to accommodate the data. When the chosen behavior fails, the upper bound shifts and another behavior might become the best option available. The time that it takes to switch to another behavior depends on the previous experiences.

Which formula can be used for the calculation of the confidence interval (or upper bound), depends on the data. In this case the data is binomial (success (1) or failure (0)). The formula must be able to handle sample sizes starting from 0 and be accurate in all cases.

Article [6] shows that most statistics textbooks use the *Wald confidence interval* (introductory) or *exact confidence interval* (advanced). Better alternatives are the *Wilson score* and the *adjusted Wald* intervals for *p*. They provide shorter intervals with actual coverage probability usually nearer the nominal confidence level, even for small sample sizes. It is to be expected that some object-behavior combinations will have near 100% success rate, while others have a near 0% success rate. In this case the *adjusted Wald* interval is the best option, since it is also accurate with these scores. Similar data as [6] can be found in [7].

The *Adjusted Wald interval* is similar to the original. It differs from the normal *Wald interval* by changing the value of sample size (n) and number of successes (X). The squared z-score ($z^2_{1-\alpha/2}$) is used which is dependent on the probability that the data is not representative for the actual population (α). The sample size is adjusted by adding the squared z-score, resulting in the adjusted sample size (ñ), equation 1. The number of successes is adjusted by 0,5 times the squared z-score (which makes $\tilde{X}$). This is used for the calculation of the adjusted midpoint of the interval ($\tilde{p}$),equation 2. With these values the upper and lower bounds can be calculated, equation 3.

$$\tilde{n} = n + z^2_{1-\alpha/2} \tag{1}$$

$$\tilde{p} = \frac{X + (z^2_{1-\alpha/2} * \frac{1}{2})}{\tilde{n}} \tag{2}$$

$$\tilde{p} \pm z_{1-\alpha/2} \sqrt{\frac{\tilde{p}(1-\tilde{p})}{\tilde{n}}} \tag{3}$$

The data from 3 object-behavior combinations is used to study the confidence interval. This is done by plotting the confidence interval with n ranging from 0 to 6.

2.4 The Algorithm

With the behaviors created and the formula's for the upper bound known, the algorithm can be made. The program consists of two parts, one for *learning which behavior is best for grabbing an object* and one for *executing the best behavior*.

When the program is started the user (human or robot) can chose which of the two options is to be executed.

The next segment of pseudo code shows how the best behavior for grabbing is chosen. There are two variables given to the method. The first is OBJECT_NAME which is a string with the object name. The second is BEHAVIOR_NAMES which is a list of strings, each filled with a name of a behavior. The GRAB_DATA variable stores the results from previous grab experiences.

```
program grabObject(OBJECT_NAME, BEHAVIOR_NAMES):
  HIGHEST_UPPER_BOUND = 0
  for NAME in BEHAVIOR_NAMES:
    GRAB_DATA = getGrabData(NAME,OBJECT_NAME)
    NEW_UPPER_BOUND = calcUpperbound(GRAB_DATA)
    if NEW_UPPER_BOUND > HIGHEST_UPPER_BOUND:
      HIGHEST_UPPER_BOUND = NEW_UPPER_BOUND
      NAME_OF_BEST_BEHAVIOR = NAME
  Execute NAME_OF_BEST_BEHAVIOR on the nao
  storeData(getResult(), OBJECT_NAME, NAME_OF_BEST_BEHAVIOR)
```

There are 4 other methods called in this code. The first is getGrabData(NAME, OBJECT_NAME) which retrieves the data from previous grab experiences. The second is calcUpperbound(GRAB_DATA), which is described in the next segment of pseudo code. The third is storeData(getResult(), OBJECT_NAME, BEHAVIOR_NAME). This stores the data to a variable in the program or an external location. The first of the three inputs is getResult(), which retrieves the result of the grab. The variables names are in all cases, in the pseudo code and the text, identical to the variables presented in the program grabObject().

Variable name	**Description**
GRAB_DATA	List with grab data
Z	$z^2_{1-\alpha/2}$
DATAPOINTS	No of data points
SUCCESSES	No of ones
N_ADJ	$\tilde{n}$
X_ADJ	$\tilde{X}$
P_ADJ	$\tilde{p}$
UPPER_BOUND	The upper bound

```
program calcUpperbound(GRAB_DATA):
  Z = retreaveZScore()
  DATAPOINTS = GRABDATA.length()
  SUCCESSES = no_of_ones(GRABDATA)
  N_ADJ = DATAPOINTS + Z^2
  X_ADJ = SUCCESES + [(Z^2)/ 2]
  P_ADJ = XADJ / NADJ
  UPPER_BOUND = PADJ + Z * squareRoot(
       [P_ADJ * (1 - P_ADJ)] / N_ADJ)
return UPPER_BOUND
```

The method *no_of_ones(GRAB_DATA)* returns the number of values that are 1 in the list *GRAB_DATA*. The function squareRoot(x) returns the square root of x.

The next segment of pseudo code describes how to grab an object. This is used to bootstrap the robot. It requires two inputs. The first is the OBJECT_NAME which is a sting with the name of the object that needs to be trained. The second is

NO_OF_GRABS, the number of grab trials per object-behavior combination. With this information each behavior is tested that number of times, and after each execution the result is requested and stored.

```
program trainObject(OBJECT_NAME, NO_OF_GRABS):
     for BEHAVIOR_NAME in BEHAVIOR_NAMES:
       for I in range(NO_OF_GRABS):
         executeBehavior(BEHAVIOR_NAME)
         storeData(getResult(), OBJECT_NAME, BEHAVIOR_NAME)
```

2.5 Implementation

Now the algorithm is ready, the program can be made. The code is written in programming language Python. To get decent code, the test driven programming technique is used. In the program the data is saved as a list of dictionaries. Offline the data is saved to text files.

Test Driven Programming and Data Storage.
Test driven programming means that the programmer creates a set of tests for a certain method, before writing the code. These tests can *pass* or *fail.* Since the code that needs testing is not yet written, all new tests should return *"fail"*.

Then the code for this method is written, which is finished when all the tests return *"pass"*. The tests are run by a program called the testrunner which calls each separate test. The set of tests has to be as extensive as possible. When this is done and all the tests *pass*, the claim can be made that code is functioning proper. This reduces debugging time, possibly to zero. Also when parts of the code or its dependencies change, it is easy to check if all the parts of the program still work.

In the program each behavior stores its data in a dictionary. A dictionary is a data structure in the programming language Python, that has key: value pairs. These pairs are unordered and each key is unique within one dictionary. The used key is the name of the object, and the value is a list containing the successes and failures that this behavior had while trying to grab the object. In this structure it is easy to add new objects. All the dictionary's, one for each behavior, are combined in a list.

When the program goes off-line, the data is stored in text files. There is one file per dictionary. To do this, a simple program is created that should work on any system. When the program is activated, the data is retrieved again.

Program Description.
When the program is started the user (person or robot) starts by choosing the action. This can be grabbing an object (grabObject) or training it (trainObject). The data from previous experiences is collected out of data files. Then the program asks for the object name and executes the chosen action. After this execution the data is stored into data files.

grabObject uses the object name to retrieve the corresponding data and calculate the upper bounds for each object-behavior combination. The behavior with the

highest upper bound is then selected and executed. Finally the feedback of the grab is collected and stored into the corresponding dictionary.

trainObject asks for the number of times that each behavior is to be tested. After that, each behavior is tested that number of times before continuing with the next behavior. After each test the feedback of the grab is collected and stored into the corresponding dictionary

In both cases the execution of the behavior has the following sequence. The robot gets into the start position (Fig 2). The program then waits for confirmation of the user that it is ready to grab. The confirmation in this case was a certain keyboard input. When the confirmation is entered the behavior is executed.

The feedback for a success is '1' and '0' for a failure. This is entered manually since there is, at this moment, not yet an automated module that supplies this.

2.6 Test Setup

The robot is placed in a squatting position in front of a box /table on which the object is located. This squatting position prevents the overheating of the leg engines and improves stability of the robot. This doesn't have implications for the data of this experiment.

The object is placed on such a place in front of the robot that the robot can grab the object by only moving the arms. The behaviors grab3, grab4 & grab5 need a different table height. For these behaviors the table height is 14 cm instead of 26 cm. The legs of the NAO are placed beneath the table. This enables the torso to be within a few centimeters from the table edge. A close proximity to the table is important for the behaviors in which the robot clamps the object between the arms and the body. If the distance is too big, the object will fall between the robot and the table.

Each behavior is tested six times on each object. This is limited to six trials since this should be enough to get a reasonable indication about how successful the behaviors are and it takes a long time to collect this data**.**

In each of the 324 trials (= 6 times x 6 behaviors x 9 objects), the experimenters will place the object in front of the NAO, notify the NAO that the object is ready, let the NAO grab the object and report back the result of the action (success or failure). Per object the experimenter will be asked how many times the object must be grabbed. All the data will be written to 6 text files, one for each behavior. They contain the grab data from that behavior on each object.

3 Results

After collecting and analyzing the data there are three items to report. The first is a table containing an overview of success of each behavior per object. The second is the development of a few of the corresponding confidence intervals over six trials. The third is feedback about how the robot behaved.

3.1 Test Data

Table 2. Percentages of the grab results, 6 trials per object-behavior combination were made.

Object:	Grab0	Grab1	Grab2	Grab3	Grab4	Grab5
box	100%	100%	100%	0%	0%	0%
dessertCup	83%	17%	17%	0%	0%	0%
ball	67%	50%	67%	0%	0%	0%
bottleP	100%	33%	50%	100%	17%	0%
cup	100%	83%	83%	17%	0%	0%
sponge	100%	17%	50%	17%	33%	0%
tape	17%	83%	100%	33%	0%	17%
can	100%	100%	100%	33%	0%	0%
pringles	100%	100%	100%	0%	0%	0%
AVERAGE	**85%**	**65%**	**74%**	**22%**	**6%**	**2%**

Every trial was recorded as a success or failure. The success rate of the behavior-object combinations are reported here in table 2. To give a clearer overview, there is chosen to report percentages of the success in grab attempts instead of numbers. Each combination has been tested six times.

Here is an overview of the data. The grabbing of an object between two hands (grab0) is a very successful option for most of the tested objects. For eight of the nine objects, it was one of the behaviors with the highest success rate. The biggest problems for grab0 were the weight and texture of the tape. This made it slip between the fingers.

The expected differences between the variations “grab1-grab2” and “grab3-grab4” can be observed. The least successful behavior with these objects was grab5. It performed only one successful grab, but this was due to an different position of the object. The tape was not placed flat on the table, so the robot was able to put a hand in the center of the tape and lift it.

Seven out of nine objects have a behavior that has grabbed them in all 6 trials. Out of the other two objects the maximum scores were 5/6 and 4/6.

3.2 Upper Bounds

To get a better idea about how the confidence intervals and upper bounds behave, 3 cases are shown in fig. 6**.** In each of the 3 cases a behavior-object combination is shown. The horizontal ax shows the number of data points (trials). The interval is plotted between 0 and 1 on the vertical ax. In all cases there is a 95% confidence interval. They start with a confidence interval that has a center of 0.5, a lower bound of 0.25 and an upper bound of 0.75. The width of the first interval is 0.5.

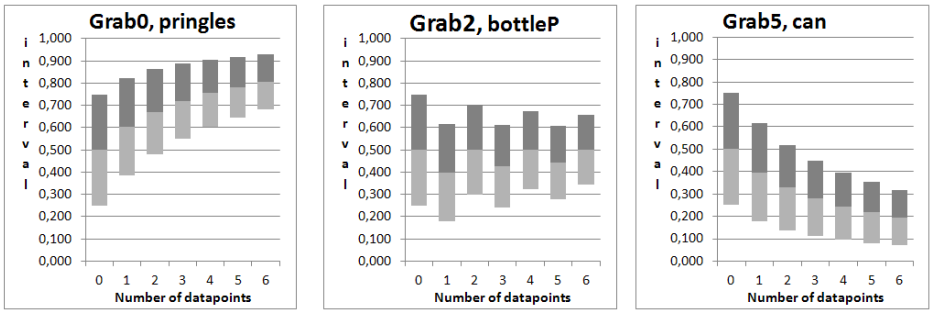

Fig. 7. These images show the change of the interval over 6 trials. The title shows to which object-behavior combination the data belongs. The first plot shows a situation with 6 successes. The second shows a situation that starts with a failure, then a success and repeats this 2 times. The third plot shows a situation with 6 failures. The lowest point is the lower bound, the point where the color of the bar shifts is the median and the highest point is the upper bound.

The confidence interval in Fig. 6. gets smaller as the amount of trials increases. In the first and the last case the width of the confidence interval has decreased from 0.5 to 0.25 after 6 trials, a 50% decrease. In the more complex second situation the width has gone from 0.5 to 0.31 after 6 trials, a 38% decrease.

3.3 Behavior of the Robot

While testing the experimenters noticed two points of attention for further testing. These are reported here.

The first point of attentions is the positions of the fingers. While grabbing the objects the fingers can get in strange positions that might harm the robot in the long run. This notion did not have any effect on the collected data, but it might be good to know for other experiments.

The second point of attention is the overheating of the engines. To lift an object an amount of force is needed. While this force is needed for grabbing (otherwise the object slips away), it is also a big strain on some of the engines. This might cause overheating for some objects or after a short time of use. At this moment without an experimenter keeping the overview, this might lead to unwanted results.

4 Discussion

In the real world it is necessary for most robots to grab objects. But how to grab multiple objects with a robot? A solution is to create multiple grab behaviors and use interval estimation to select the best choice. This has advantages above finding one optimal grabbing technique. The objects can only be grabbed with higher success and there is no need for large time consuming computations when a new object is added.

In the case of the optimum, individual performance might be dropped to retain the best overall score.

For this research six behaviors are created to grab objects and also the nine objects were selected to train on. The selected objects had to be limited in size and weight (maximum of 300 g), and are items that a human can encounter for grabbing in daily life. For the interval estimation it appeared best to use the adjusted Wald method.

Two (parts of) programs were created. The first selects the best behavior to grab an object. The second collects data by testing each behavior a specified number of times on the specified object. Inside the program the data was stored in a dictionary for each grab behavior, when the program goes off-line the data is stored in text files. The data is discussed in the next section.

4.1 Results.

Among the created behaviors there was no perfect behavior that scored 100% success for all objects. The greatest overall success can be achieved by using multiple behaviors. This agrees with our theory that there isn't always 1 best behavior that always succeeds on all objects.

The behaviors grab4 and grab5 were not very successful. There are three options for these behaviors. The first is keeping the behaviors. The second is removing the behaviors. The third is changing the behaviors to increase performance.

Behavior grab5 was unsuccessful since there were no objects that were compatible with this behavior. It is meant to get under the sides of an object, like a bowl, and then lift it from below. The dessertCup did have the right shape, but the distance between the two hands was too large and the object to small. The behavior might perform better on bigger objects. For this reason it is not necessary to remove the behavior. In case of other orientations of the objects it might also be interesting since it did once score in that case. Adding a second version with a smaller distance between the hands, might improve the current results. For instance on the dessertCup.

The researcher suggest to change behavior grab4 into a mirrored version of grab3. Both behaviors could be used as one. If the object is in front of the robot or to the left of it, use the first. In the other case use the second. The data from both executions should be stored in one data file

For the objects that are hard to grab with the current behaviors it is suggested to create new behaviors. For instance the performance of grabbing the ball can be improved. New behaviors are easily added to the structure.

The three graphs of the confidence interval's (Fig. 6.) show that the intervals get much smaller after only a few trials. In the most optimistic cases the size of the confidence interval is reduced in half. In the other case the size is decreased with 38%.

After only a few times trying each behavior, the interval will show which behavior to chose. This is the behavior with the lowest number in the behavior name. This leads to a high chance of a successful grab. In the case two scores mach, the fist behavior is chosen. The plotted confidence intervals also show that bootstrapping an object could be necessary to prevent suboptimal choices. Without it, if a behavior

starts with a few unlucky trials it might constantly be beaten by a suboptimal grabbing technique that currently has a high upper bound.

Finally there are two notions of attention. The first is the fingers of the NAO. In some situations the two fingers on one side of the hand are pushed apart. This might lead to damage to the robot, so it is good to keep an eye on this. The second is the overheating of the engines. This is probably going to deliver some problems when the robot needs to hold an object for a longer period in time.

4.2 Future Work

There are four main items that need to be researched. The first is the overheating of the engines, since this limits the advantages of grabbing. The second is a coupling with vision. Currently the robot is unable to detect the success of a grab. This coupling improves the autonomy of the grabbing behavior. The third is hot to get the robot automatically into a grabbing position. This starts with the detection of an object. Then the grabbing behavior is selected. Finally the robot can move into a grabbing position. The last is researching how differences in table height can be handled.

Further the creation of more interesting behaviors, deleting unwanted behaviors and perhaps fine-tuning some of the behaviors, can be researched. Finally research can be done in how to generalize objects. This can help new objects by profiting from the data that is collected on similar objects.

4.3 Final Conclusion

A program is created that is able to select the best possible behavior out of a list of behaviors. After bootstrapping nine objects, testing them six times with each grab behavior, seven of the nine objects had 100% success score and the other two behaviors have scores higher than 50 %.

It is easy to add new behaviors to the program and this research has shown that in this case multiple grabbing behaviors performed better than one single grabbing behavior. The best grab behavior could grab 6 objects with a 100% score. The overall result was 7 objects that had a 100% score.

This rapport shows that there still is a lot of work to be done into the manipulation of objects with the NAO. By using multiple behaviors in combination with the adjusted Wald interval part of this process is done. This idea can also be used for other robots or other situations where multiple behaviors are used and the data is binomial.

References

1. van der Zant, T. and Wisspeintner, T.: RoboCup X: A proposal for a new league where RoboCup goes real world. Lecture Notes in Artificial Intelligence, Springer, (2006)
2. Jansen, S., Lier, C., Neculoiu, P., et al.: Borg - the robocup@ home team of the university of Groningen team description paper. In proceedings of the Robocup@home competition, (2011)
3. Gouaillier, D., Hugel V. et al. "Mechatronic design of NAO humanoid". IEEE Int. Conf. on Robotics and Automation, Kobe, (2009)
4. Xue, Z., Kasper, A., Zöllner, M., Dillmann, R. (2009). An automatic grasp planning system for service robots. In Proceedings of the 14th International Conference on Advanced Robotics (ICAR), 22-26 Juni 2009, München, Deutschland (2009)
5. Zant, T. v. d., Wiering, M., and Eijck, J. v. (2005). On-line robot learning using the interval estimation algorithm. In Leone, D., editor, Proceedings of the 7th European Workshop on Reinforcement Learning, pages 11-12, Napoli, Italy.
6. Agresti, A., Coull, B.A.,: Approximate Is Better than "Exact" for Interval Estimation of Binomial Proportions, The American Statistician Vol. 52, No. 2, 119—126 (1998)
7. Sauro, J., Lewis, R.,: Estimating Completion Rates from Small Samples Using Binomial Confidence Intervals: Comparisons and Recommendations, Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 49 no. 24 2100-2103 (2005)