

CS365 Project Report

# Grasping Known Objects with Aldebaran Nao

By:

Ashu Gupta([ashug@iitk.ac.in](mailto:ashug@iitk.ac.in))

Mohd. Dawood([mdawood@iitk.ac.in](mailto:mdawood@iitk.ac.in))

Department of Computer Science and Engineering

IIT Kanpur

Mentor:

Prof. Amitabha Mukherjee

April,2012

# Abstract

*Our project aims to identify known objects through ALDEBARAN NAO and then to grasp that object. This grasping problem is of great importance for humanoids for doing even trivial things. In our project we first detect the known object using **Image Processing** techniques, then we compute the position of that object using **Machine Learning** and finally we reach the object using **Inverse Kinematics**.*

# 1 Introduction

The research interest in humanoid robots has increased tremendously since the past few years. The attention is now drifting towards creating intelligent and autonomous humanoids. The most remarkable milestone achieved in this field is *ROBOCUP*. It is a Soccer cup between autonomous robots, particularly *ALDEBARAN NAO*, which is one of the best in its class. The research for robocup is mainly related to Object Recognition and Depth Estimation through Image Processing and also the balancing of the robot.

These autonomous robots can also be used for carrying out day-to day trivial activities and might be the replacement for human beings for a lot of trivial work, in the near future. For e.g. the task of identifying an object lying on the floor, classifying it to be garbage or not garbage and throwing it in dustbin if it is garbage can be done by them. But tasks like these and many others, require it to grasp an object.

Our project focuses on identifying known objects using the camera feed of Aldebaran Nao, reaching the object by its arm and then grasping the object using its 3 fingered palm [which is of 1 DOF]. We have used Image Processing techniques for Object Detection, Machine Learning Techniques for locating the object and finally Inverse Kinematics to reach the object.

## 2 Related Work

A lot of work has been done with Aldebaran Nao, but most of it has been restricted to the needs of ROBOCUP. The work for robocup mainly involves detection of the various elements of the field including ball, other Nao's, different lines on the floor, goalpost, etc. and responding differently to the different things detected.

In a related work, Tomás González Sánchez [1], laid out the different techniques for identifying the object in a field [Robocup] and the depth estimation. In his work he has mentioned of different techniques that can be used for depth estimation including heights and distances, Stereoscopic vision and creating a table of distance with area occupied on the image of object.

In our approach, we have used machine learning technique for depth estimation which is somewhat similar to the third method suggested in his paper. It involves linear curve fitting or regression analysis.

After locating the object we have used the inverse kinematics equation to reach the object.

### 3 Overview of the project

Grasping Known Objects with  
Aldebaran Nao

```
graph TD; A[Grasping Known Objects with Aldebaran Nao] --- B[Image Processing]; A --- C[Machine Learning]; A --- D[Kinematics]
```

Image Processing

Machine  
Learning

Kinematics

## 4 Image Processing

We have used image processing techniques for detection of the object. We have used OpenCV libraries in python to implement our Image Processing codes. The following steps were followed for detecting the object--

**1. Getting the BGR image from Nao---**

First of all we get the camera feed from Nao, preferably the lower camera.

**2. BGR to HSV**

The camera feed is then converted to HSV image format from its present BGR format. The reason for converting to HSV format is that it produces better results for creating binary image and also it makes the computing faster. However, the HSV format may respond differently to change in lightening conditions and may produce remarkably different results in different light conditions.

**3. HSV to Binary Image**

The HSV image is then converted to binary image for proper recognition of the object shape. The binary image consists of only 2 colours. One colour is for the object and the other is for everything else. We preferably use black and white for that purpose. Due to presence of only two colours it makes object detection much easier as we have to only match which patch is of the required shape.

**4. Smoothing Binary Image**

The next step is the smoothing of the binary image to avoid rough boundary and removing discontinuous changes in pixel values. This is achieved by applying Gaussian Filter to the binary image. This smoothing improves our results substantially.

**5. Detection of circle, its centre and radius**

Next, we detect the circle in our smoothed binary image. We also mark the centre of circle and get its radius. All this is done using functions of OpenCV libraries.

A diagram containing all the images in order.



INPUT IMAGE

HSV IMAGE

BINARY IMAGE

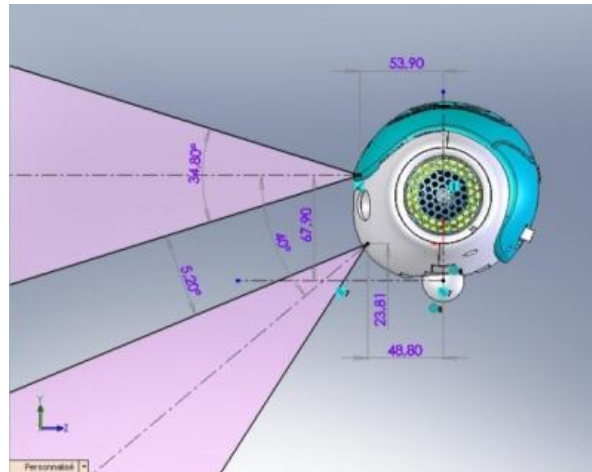
SMOOTHEN BINARY IMAGE

FINAL IMAGE

## 5 Depth Estimation and Locating the Object

Locating the object accurately is one of the most challenging part of our project. For this we have used the machine learning technique of Regression Analysis [4] which is explained later. In this technique we use a training set [1] to get results for our test set. Before that, we need to know some basic things about Nao's camera.

### 5.1 Perception in Aldebaran nao

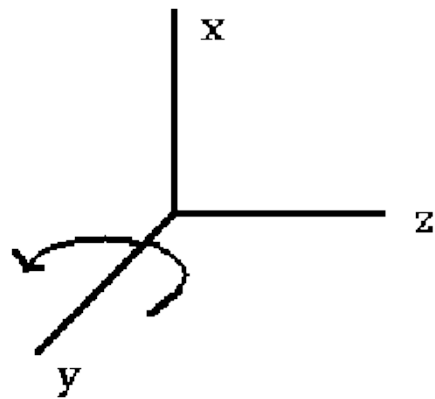


Field of view of Nao [2]

The Robot has 2 CMOS cameras each one provides 640x480 and on a frame rate of 30 fps [2]. They are located with offset of 40 degrees in the vertical axis.

### 5.2 Co-ordinate Transformation

As the optical axis of bottom camera is not fixed and depends on the head pitch angle we need to transform coordinates before training nao.



$Q = 40 \text{ degrees (offset of bottom camera) + Head pitch angle}$

$$z' = z \cos \theta - x \sin \theta$$

$$x' = z \sin \theta + x \cos \theta$$

$$y' = y$$

$x, y, z$  are co-ordinates in nao's COM frame

$x', y', z'$  are co-ordinates in nao's bottom camera

The model that we have used for depth calculation is ---

$$X = a + b/r$$

$X$  = depth of ball from camera.

$r$  = radius of ball in image.

Here  $a$  and  $b$  are constants.

For calculating  $y$  we have used----

$$y = \frac{m * (Y + k)}{X + A}$$

$Y$  = object  $y$  distance.

$y$  = image  $y$  distance.

Here  $m$  and  $k$  and  $A$  are constant.

The model for  $z$  is similar to that of  $y$ .

We first collect our training data set. For each  $r$  there is a corresponding value of  $P$ .

Then we have

$$X = a + b/r$$

Let's call  $\beta$  to be the matrix containing the constants  $a$  and  $b$ . The  $n$  on solving the linear least squares problem we would get----

$$\beta = (A^T A)^{-1} A^T X$$

where  $A_{ij}$  is the partial differential of our proposed function at  $i^{\text{th}}$   $x$  with respect to the  $j^{\text{th}}$  element of  $\beta$  matrix or the  $j^{\text{th}}$  constant in our model or the proposed function.

We put our training set in the above equation to generate the  $\beta$  matrix containing the constants.

Thus in this way we can solve for  $y$  and  $z$  dimensions also.

### 5.3 Radius and Distance Values



$r=23$   
Depth=8.3 cm  
Error= $\pm 5$  mm



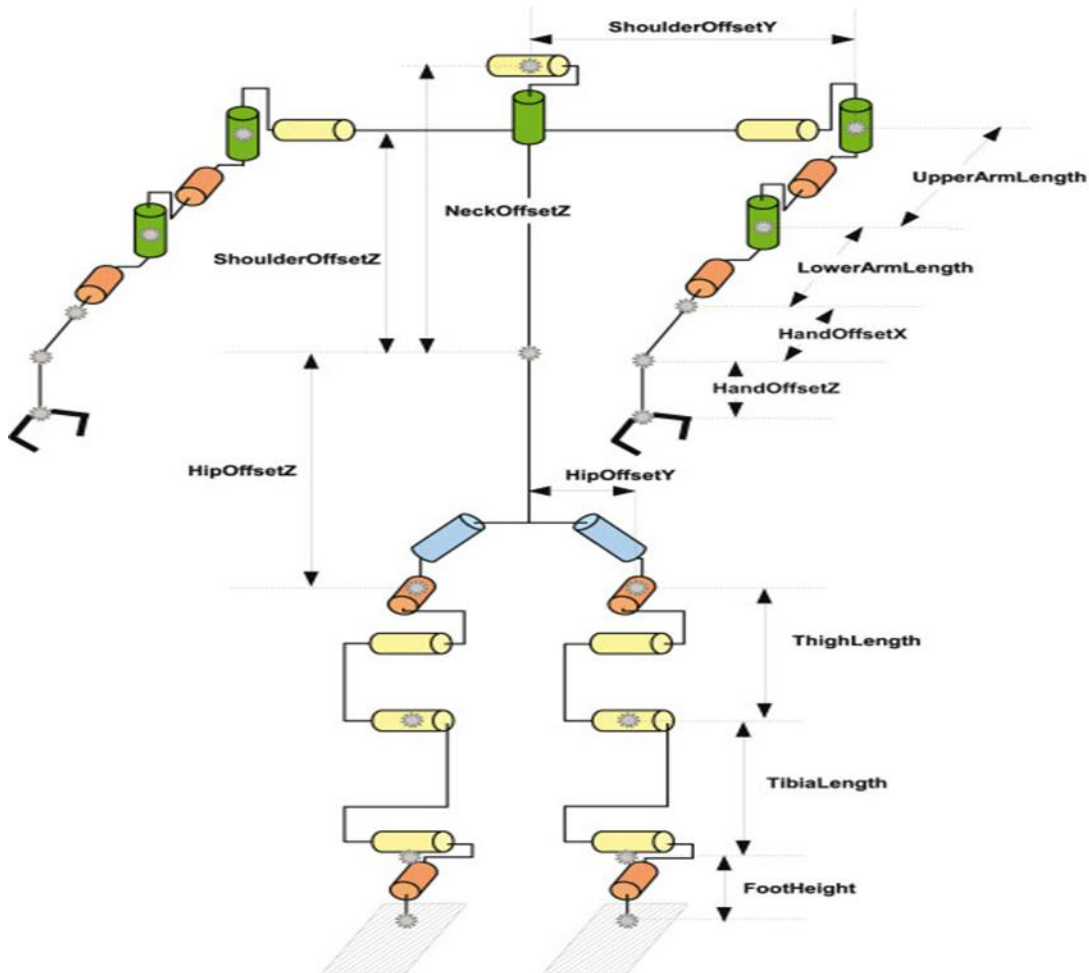
$r=28$   
Depth = 4.2 cm  
Error =  $\pm 3$  mm



$r=33$   
Depth = 2.1  
Error =  $\pm 3$  mm



## 6 Kinematics of Nao

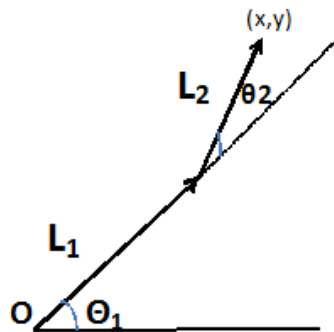


Nao Architecture

When we get the 3D space information from the image, we move the end-effector of its left hand to the calculated co-ordinates and grasp the object

Solving the inverse kinematics equation for the arm we get  $\theta_1$  and  $\theta_2$  and then we move arm such that the end effector reaches  $P_2$  and the arm angles are  $\theta_1$  and  $\theta_2$ .

We consider the inverse kinematics equation in the x-y plane at shoulder height. We find the link lengths of the arm by experimentally changing  $\theta_1$  and  $\theta_2$  and observing corresponding change in co-ordinates of end-effector.



### Finding $L_2$ -----

We changed  $\theta_2$  to  $\theta_2 + \delta$

$$\Delta x = L_2 \cos(\theta_1 + \theta_2 + \delta) - L_2 \cos(\theta_1 + \theta_2) \quad (1)$$

$$\Delta y = L_2 \sin(\theta_1 + \theta_2 + \delta) - L_2 \sin(\theta_1 + \theta_2) \quad (2)$$

Using (1) and (2) we find  $L_2$ .

### Finding $L_1$ -----

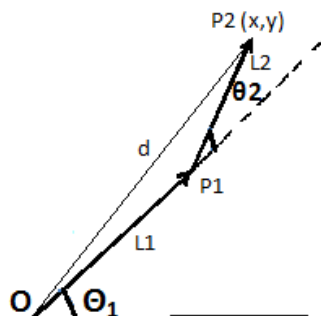
Changing  $\theta_2 = 0$  and  $\theta_1 = \theta_1 + \delta$ ,

$$\Delta x = (L_1 + L_2) \cos(\theta_1 + \delta) - (L_1 + L_2) \cos(\theta_1) \quad (3)$$

$$\Delta y = (L_1 + L_2) \sin(\theta_1 + \delta) - (L_1 + L_2) \sin(\theta_1) \quad (4)$$

Using (3) and (4) we find  $L_1$ .

### Solving Inverse Kinematics for x and y



In triangle  $OP_1P_2$

$$\angle O = \cos^{-1} \left( \frac{(d^2 + L_1^2 - L_2^2)}{2dL_1} \right)$$

$$\angle P_2 = \cos^{-1} \left( \frac{(d^2 + L_2^2 - L_1^2)}{2dL_2} \right)$$

$$\Theta_1 = \tan^{-1}(y/x) - \angle O$$

$$\Theta_2 = \angle O + \angle P_2$$

For moving end-effector to  $(x,y,z)$ , we move the end-effector to  $(x,y)$  in  $x$ - $y$  plane in the shoulder height. Then we move to  $(x,y,z)$  by moving the end-effector along  $z$ -axis.

## 7 RESULTS

Nao now grasps the object with an error upto 4cm. If the height of object is known, then error reduces significantly (upto 5mm).



Nao grasping from Hand



Nao grasping from table

Grasping from table is much hard compared to grasping objects from hand is much easier as human tend to aid in grasping , by accommodating for small errors.

## 8 Conclusion and Future Improvement

The grasping of unattended object like lying on table is very inaccurate as the precision required is very high and very low tolerance.

- Grasping of known objects of different geometry.
- Visual Servoing.
- Implementation of Virtual Thumb Rule for more accuracy.

## 9 References

[1] Master's Thesis of Tomàs González Sánchez, Department of Computer Science and Mathematics, *Universitat Rovira i Virgili*, September 2009, 64-82

[2] Documentation of Aldebaran Nao from Aldebaran Robotics – Documentation Version 1.3.17.

[3] Website of Aldebaran Robotics <http://www.aldebaran-robotics.com/> as observed from February 2012 to April 2012.

[4] Wiki page of Linear Least Squares [http://en.wikipedia.org/wiki/Least\\_squares](http://en.wikipedia.org/wiki/Least_squares) as observed on 27th March 2012.

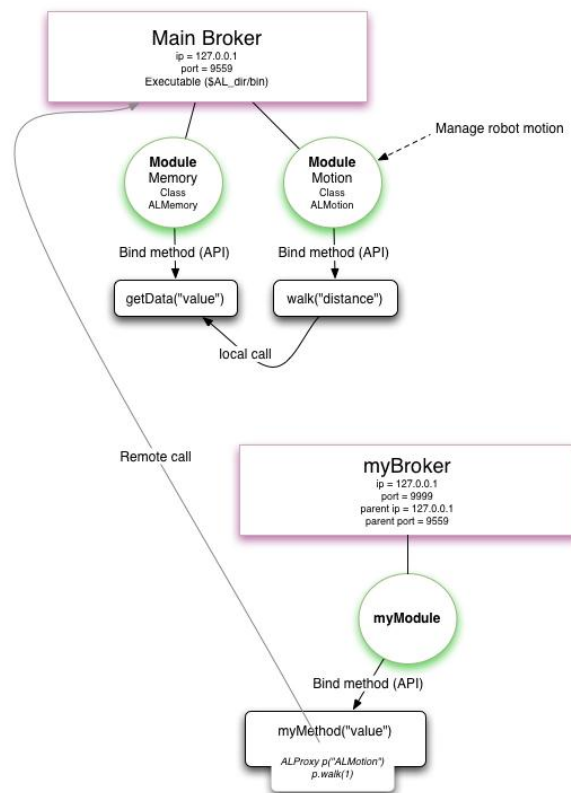
# 10 Appendix

## 10.1 Connecting to Nao

The most common way of connecting to Nao is through WEP or WPA Wi-Fi networks. Use of LAN cables to connect to Nao is strongly discouraged considering the fact that Nao is mobile and may fall due to restriction of LAN cable.

There are a lot of inbuilt modules and libraries in nao which help in connecting and programming aldebaran nao. Naoqi satisfies minimum requirements for quick development .

A broker is an executable that listens to commands on an IP address and port. A default broker is listening to the IP and the Port where nao is connected [2]. A broker is server and an executable. We use ALProxy module to send commands to the broker locally and remotely.



Running a module on Nao [2]

## 10.2 Problems Faced

1. Nao has got problems in connecting to WEP Wi-fi network sometimes.
2. The camera feed of Nao is very poor. The frame rate is extremely poor.
3. The inbuilt motion modules of Nao are often inaccurate.
4. The response to the commanded angles is at times considerably inaccurate.

All codes used in the project were our own and not taken from anywhere.