CENG300 Summer Practice Report
Department of Computer Engineering
METU

Summer 2014

Student Name : Ilker Bozcan
Student ID : 1881093
SP Location : Kovan Research Laboratory
SP Date : 30/06/2014 - 25/08/2014

Table of Contents

1 Introduction

I have performed my summer practice in Kovan Research Laboratory. The internship lasted 50 days. There were several reasons in choosing Kovan Research Lab as my summer practice location. Firstly, since I have wanted to work in academic life, I have wondered that what laboratory working environment have looked like. Secondly, I have been interested in humanoid robots and artificial intelligence and Kovan Research Lab have given me a chance of working with humanoid robots.

During my summer practice, I have built a scenario with NAO Humanoid Robot[1]. In scenerio, NAO has cleared an area from some objects by carring them to out of area. Border of area has determined by black-yellow security line tape that sticked to ground. Objects were seperated into two groups: small objects and big objects. NAO was able to carry small objects by own. However, big objects could be carried only by assistance of human.

In my summer practice, I have focused mainly building of this scenerio. At the end of my internship, I have been familiar with basic concepts of humanoid robots and computer vision.

This report includes detailed and comprehensive explanation of works that I have conducted during the summer practice, after having introduced Kovan Research Laboratory and NAO Humonaid Robot.

2 Kovan Research Laboratory

Kovan Research Lab is located at MODSIMMER(Modelleme ve Simulasyon Merkezi) building in Middle East Technical University. It was founded to study the synthesis of intelligent systems inspired from nature[2]. Ongoing projects are based on artificial intelligence and image processing.

2.1 People

Assistant Professor Erol Sahin is head of KOVAN. Associate Professor Gokturk Ucoluk and Associate Professor Sinan Kalkan are members of KOVAN Lab. There are also two PhD. students and three Msc. Students in Kovan.

2.2 Robots

There have been three robots which have been used for ongoing projects including NAO which I have used during my internship. ICUB humonid robot is the most advanced robot in KOVAN. It has had more sensors and joints which have made ICUB more humanoid than NAO. The other one is an quadrotor that is lifted and propelled by four rotors. The last one is NAO. It will be described thoroughly.

3 NAO Humanoid Robot

NAO is programmable humanoid robot produced by Aldebaran. It can walk, talk, listen and interact with environment around it. It is used for education for children or academic research[3]. Also, It has been using for RoboCup since 2007.

There are several versions of NAO. NAO V4 is used in KOVAN. Its spasifications are here:

- ATOM Z530 1.6 GHz CPU
- 1 GB RAM
- 2 GB Flash memory
- 8 GB Micro SDHC

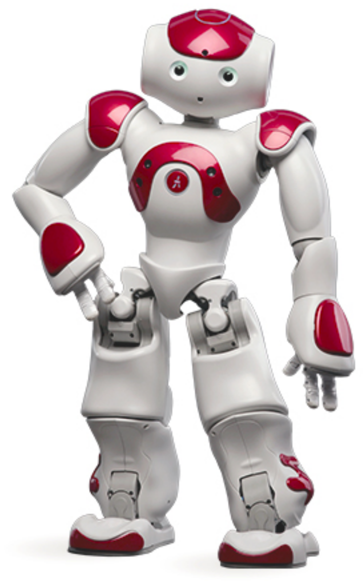


Figure 1: NAO Humanoid Robot

3.1 Constractions

3.1.1 Dimension and Masses

NAO v4 has 573 mm height, 311 mm depth and 275 mm width.Its mass is 5.4 kg.[4]

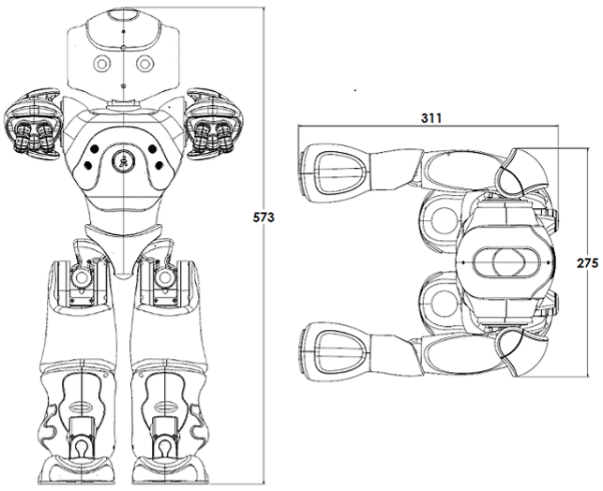


Figure 2: Dimensions of NAO

3.1.2 Kinematics Data

Three axis are defined for NAO. The X axis is positive toward NAOs front, the Y from right to left and the Z is vertical.

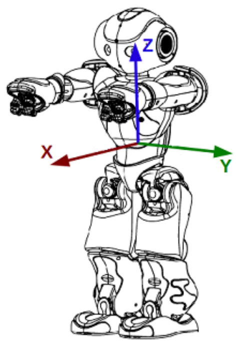


Figure 3: Axis of NAO

3.1.3 Joints

There are several joints in NAO's body which perform rotation in three axis. Roll rotations take place around the X axis, pitch rotations take place around the Y axis and yaw rotations take place around the Z axis.

There are two joints in NAO's head which named **HeadYaw** and **HeadPitch**. HeadYaw is the joint that twist head(motion around Z axis) in range -119.5 to 119.5 degrees. HeadPitch is the joint that move head front and back(motion around Y axis) in range -38.5 to 29.5[5].

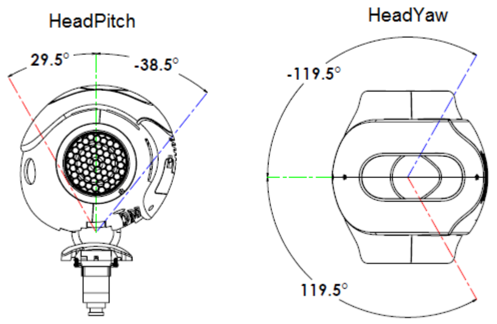


Figure 4: Range of Head Joints

NAO's arms have twelve joints(6 per each arm). Left arm joints and right arm joints are identical. These joints are **ShoulderPitch**, **ShoulderRoll**, **ElbowYaw**, **ElbowRoll**, **WristYaw** and **Hand**. ShoulderPitch is the joint that moves shoulder front and back(Y) in range -119.5 to 119.5 degrees.

ShoulderRoll is the joint that moves shoulder right and left(Z) in range -18 to 76 degrees. ElbowYaw is the joint that twists shoulder(X) in range -119.5 to 119.5 degrees. ElbowRoll is the joint that moves elbow in (Z) axis in range -88.5 to -2 degrees. WristYaw is the joint that moves wrist in (X) axis in range -104.5 to 104.5. Hand joint is the joint that opens or closes fingers.

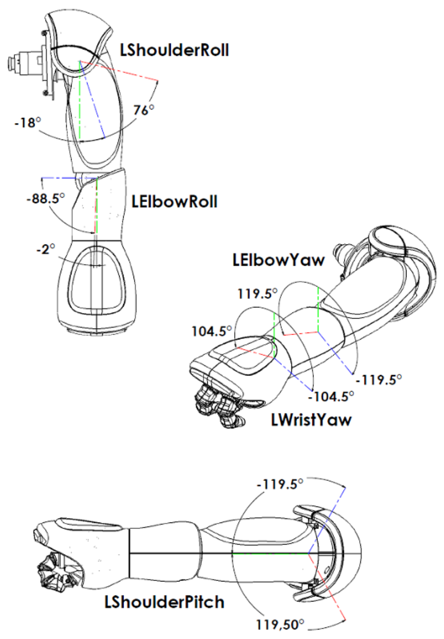


Figure 5: Range of Arm Joints(left arm shown)

There are also ten joints in NAO's legs(five per each leg). These are **hipRoll**, **hipPitch**,**kneePitch**,**anklePitch**, **ankleRoll**.

HipRoll is the joint that moves hip right and left(X) in range -21.74 to 45.29 degrees. HipPitch is the joint that moves hip front and back(Y) in range -88.0 to 27.73 degrees. KneePitch is the joint that bends knee(Y) in range -5.29 to 121.04 degrees. AnklePitch is the joint that moves ankle front and back(Y) in range -68.15 to 52.86 degrees. Ankle Roll is the joint that moves ankle right and left(X) in range -22.79 to 44.06 degrees.

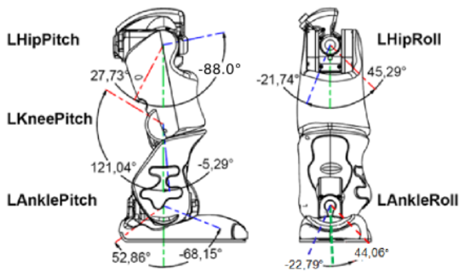


Figure 6: Range of Leg Joints(left arm shown)

3.2 Interaction

3.2.1 Loudspeakers

NAO is equipped with a stereo broadcast system made up of two loudspeakers in its ears.One of them is on the left ear and the other is on the right ear.

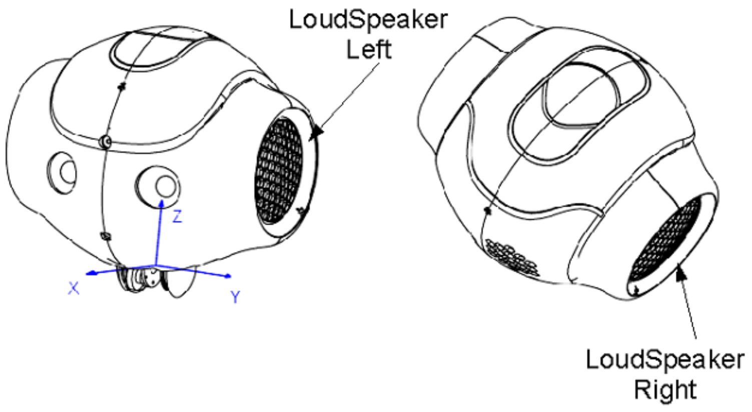


Figure 7: Location Of Loudspeakers

3.2.2 Microphones

There are four identical microphones on NAO's head. The electrical bandpass of the microphones ways is 300Hz - 8kHz[6]. Thanks to four microphones which has different position, NAO can recognize location of sound source respect to its own.

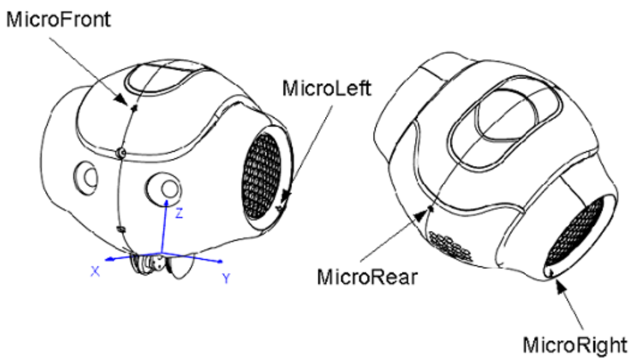


Figure 8: Location Of Microphones

3.2.3 Video Cameras

Two identical video cameras are located in the forehead.

They provide a up to 1280x960 resolution at 30 frames per second[7]. They can be used to identify objects in the visual field such as goals and balls, and bottom camera can ease NAOs dribbles.

Thanks to NAO Vision API, some camera parameters (resolution, colorspace, frame rate etc.) can be changed.

Unfortunately, since fields of view of two cameras do not overlap, stereo vision techniques cannot be used except objects that placed so far.

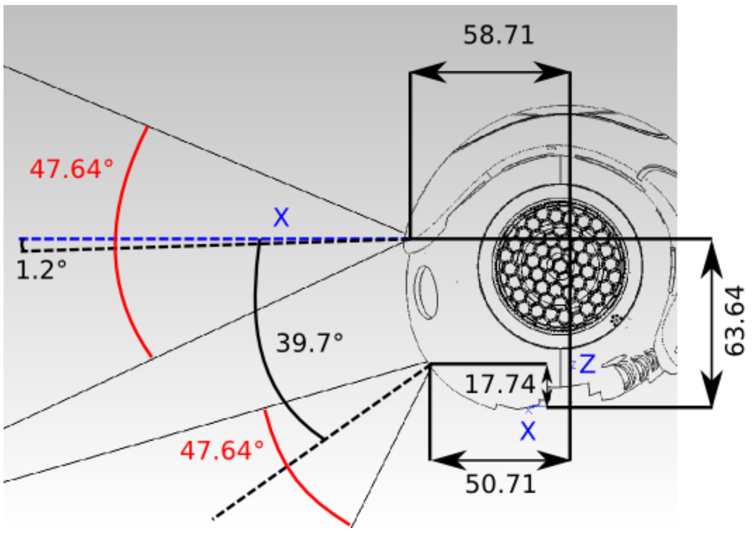


Figure 9: Location Of Cameras and Field of Views (Lateral View)

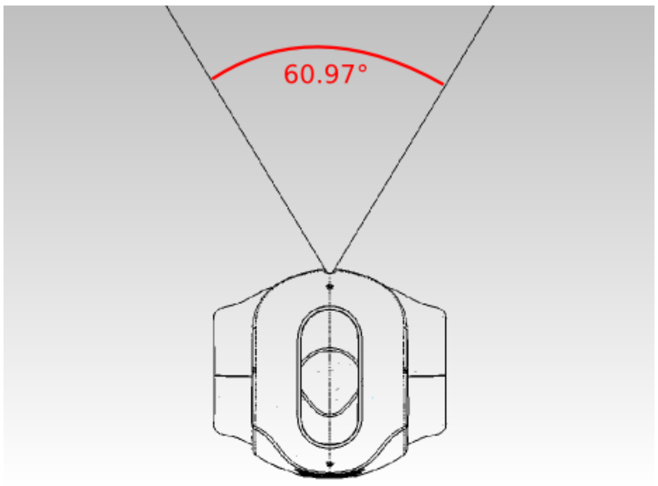


Figure 10: Field of Views (Top View)

3.2.4 LEDs

There are various leds in NAO's whole body. These leds may be used for feedback or making NAO more humanoid by blinking. All leds can be controlled by NAO LED api. One led can have much more than hundred color.

LEDs that NAO has are here:

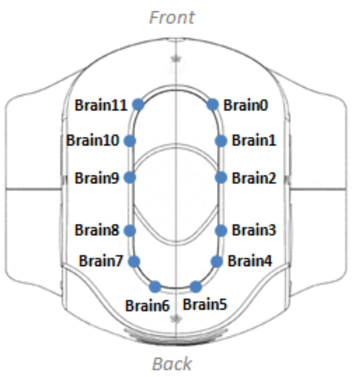


Figure 11: Head tactile sensor LED locations)

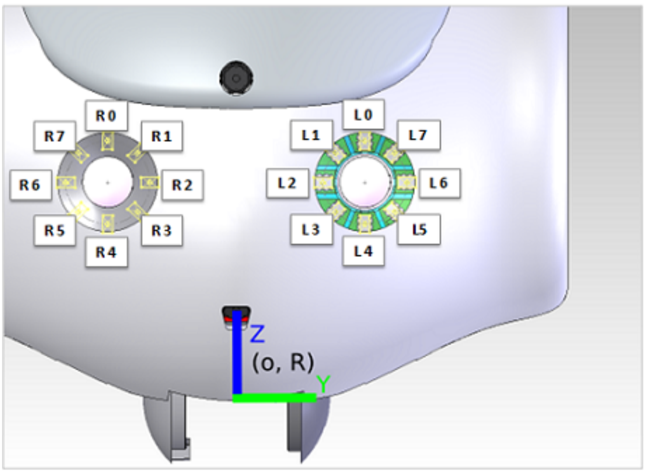


Figure 12: Eye LED locations

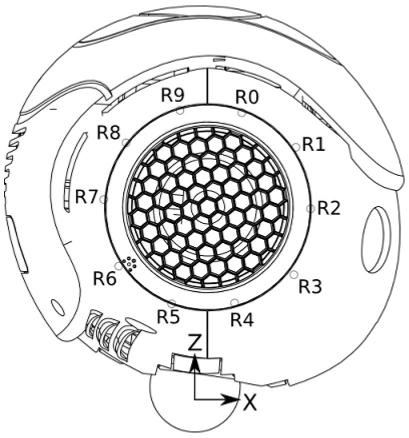


Figure 13: Right ear LED locations(Left and right ear leds are identical)

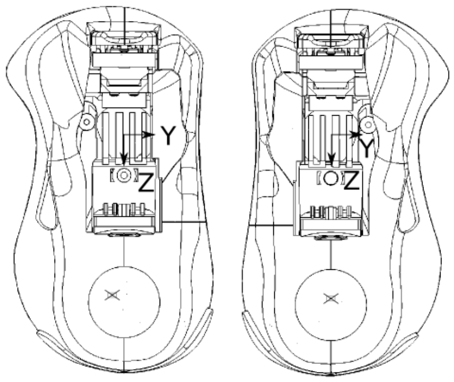


Figure 14: Foot LED locations

3.3 Sensors

There are four main sensor on NAO Robot which I have dealt with during my summer practice. These are **Force Sensitive Resistors** , **Sonars**, **Joint Position Sensors** and **Contact and tactile sensors**.

3.3.1 Force Sensitive Resistors(FSRs)

Eight FSRs are located under NAO's foots(four per each foot). These sensors measure a resistance change according to the pressure applied. The FSR located on the feet have a working range from 0 N to 25 N.[8]

They are usually used for checking if NAO stands on both foots or not.

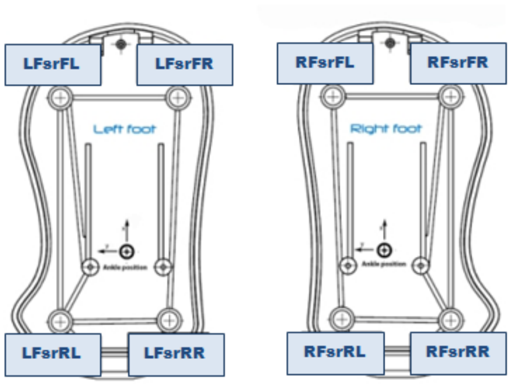


Figure 15: Location of FSRs

3.3.2 Sonars

NAO is equipped with two ultrasonic sensors (or sonars) which allow it to estimate the distance to obstacles in its environment. Official website says that "The detection range goes from 25 cm to 255 cm, but under 25 cm there is no distance information, the robot only knows that an object is present."[9].

Sonars may be used for detection of obstacles. However, the distance between object and NAO cannot be calculated accuretaly. Therefore, in my project, I have not preferred to use sonars.

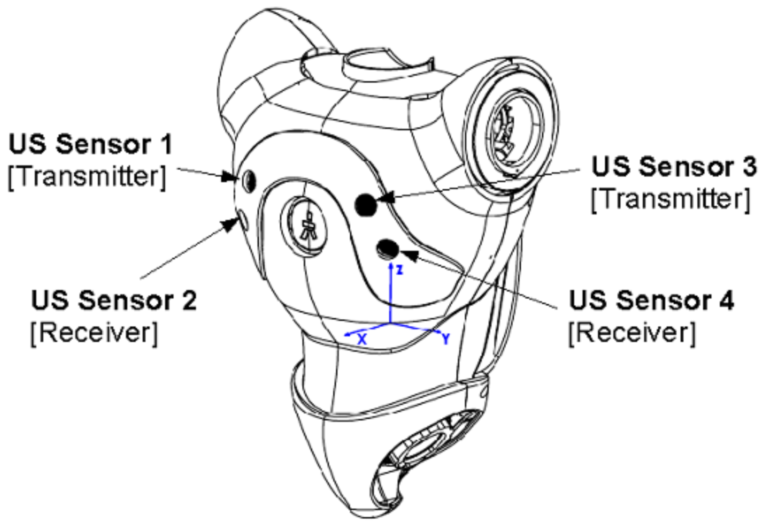


Figure 16: Location of Sonars

3.3.3 Joint Position Sensors

NAO can get data of joints' positions and angles. Even if these data are not used directly, most of NAO's APIs use JPSs data.

Thanks to that, pose of NAO is always known. Benefits of JPSs will be explanied and understood in "Scenerio" section.

3.3.4 Contact and Tactile Sensors

There are two tactile sensors on NAO which placed head and out of hand. Furthermore, there are one chees button and two feet bumbers(one per each foot).

Contact and tactile sensors are the most clear sensors on NAO's body. There is actually no noise for these sensors. However, in my project I have tried to avoid to use these sensors in order to make NAO more humanoid.

Tactile head is seperated in three part. In figure 17, A,B,C are the front,middle and back parts of the touch sensitive tactile head respectively.

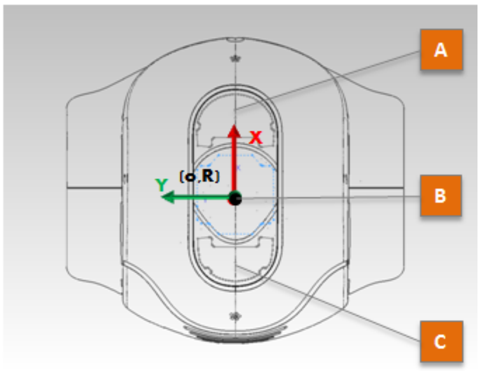


Figure 17: Parts of Tactile Head

Chest button is a contact sensor of NAO. It is also used for open/close operations. In figure 18, A shows chest button.

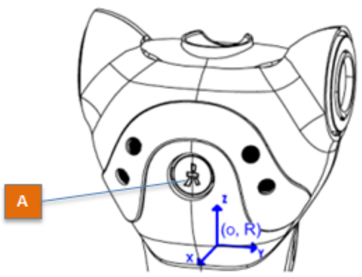


Figure 18: Chest Button

Hand Tactile Sensors consist of three parts for each hand. In figure 19, A and C shows sensors that placed left and right sides of hand. B shows middle sensor that is over the hand.

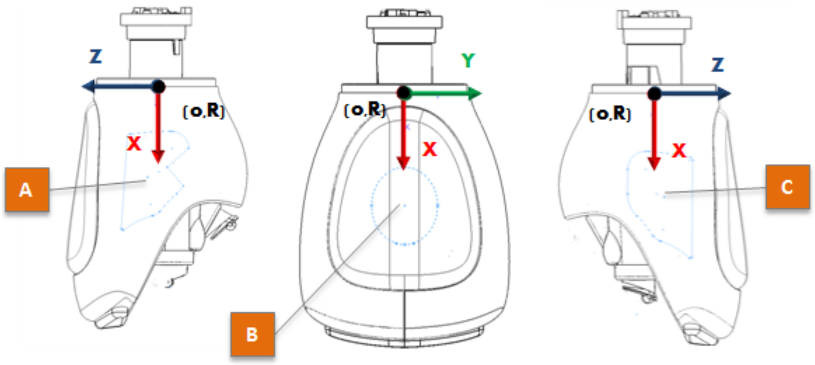


Figure 19: Tactile Hands

Lastly, there is the bumper which consists of two parts for each foot. These are simple ON/OFF switches, thus 4 in total. These bumper are usually used for stopping NAO when his foot contacts with an object while it is walking. Figure 20 shows these bumpers.

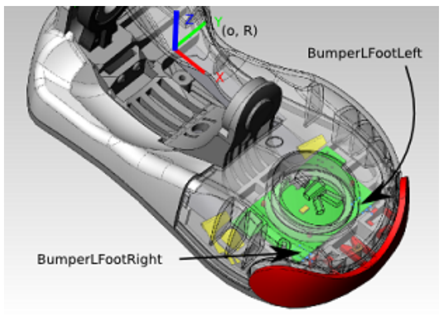


Figure 20: Feet Bumpers

3.4 Software

3.4.1 NAOqi Framework

NAOqi Framework is a linux based operation system that runs in NAO. This framework allows communication between different modules(motion, vision, sensors, audio etc.).

Modules are basically a class within a library and each module have variaity of functions that can be worked on NAO. For instance, **ALMotion** module has **walk()** method which starts NAO to walk or **ALVision** module has **getFrameFromLocal()** which capture frame from selected video camera. All modules and their functions can be seen in Aldebaran Website.Furthermore, NAOqi Framework gives to users knowledge of what module is running on NAO. In addition, it knows if a function of any module is being executed.

All recorded dump files, photos, videos, text files etc. are saved in NAO's memory. Users can connect NAO via SSH.

3.4.2 Software Development Kit(SDK)

NAO applications and new modules can be developed by Python or C++. Related SDK have to be used for programming NAO robots. These SDKs allow programmers to use NAO APIs in their project. Using Python is one of the easiest ways to program with NAO robots. I have preferred Python in my project.

3.4.3 Choregraphe

Choregraphe is a multi-platform desktop application, allowing users to:

- Create animations, behaviors and dialogs,
- Test them on a simulated robot, or directly on a real one,
- Monitor and control you robot,
- Enrich Choregraphe behaviors with your own Python code.
- Boot or upgrade NAOqi.

Choregraphe allows users to create applications containing interaction with people, dance, e-mails sending, without writing a single line of code.

During my summer practice, I have used Choregraphe in order to develop my project.

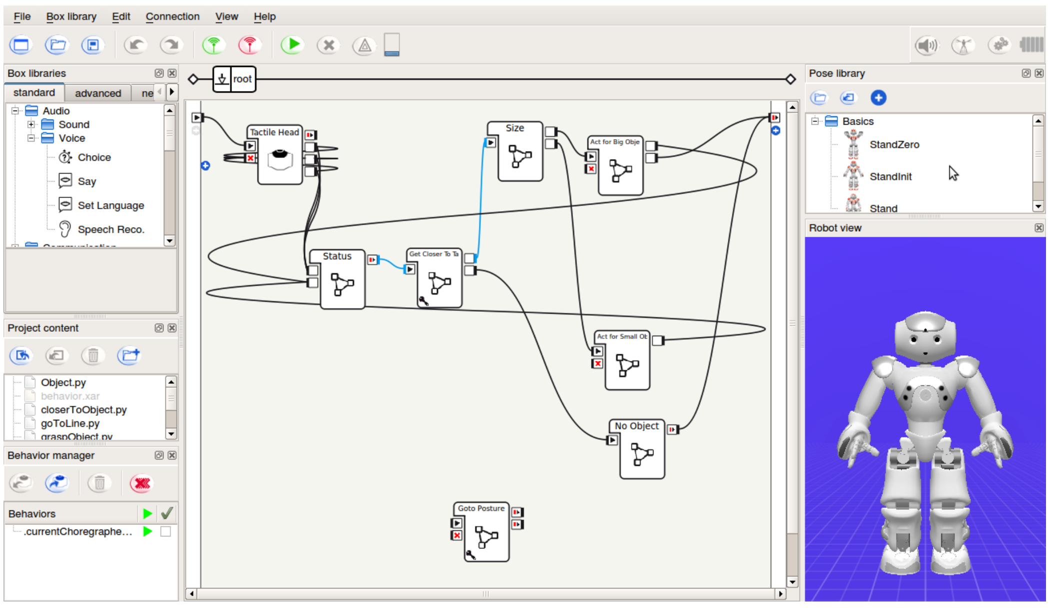
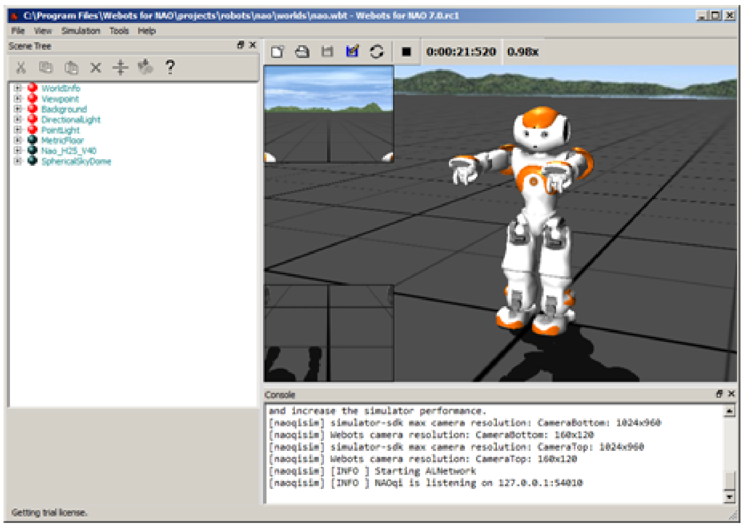


Figure 21: Screenshot from Choregraphe

3.4.4 Webots for NAO

Webots for NAO is the simulator for NAO. It offers safe place to try behaviours before playing them in real robot. There are many environment and object options in Webots.

Some useful NAO modules(including vision and sensing modules) are missing in trial version of Webots for NAO. Therefore, I have used it for only test complex movements like carry object or sit down.



```
and increase the simulator performance.
[naoqisim] simulator-sdk max camera resolution: CameraBottom: 1024x960
[naoqisim] Webots camera resolution: CameraBottom: 160x120
[naoqisim] simulator-sdk max camera resolution: CameraTop: 1024x960
[naoqisim] Webots camera resolution: CameraTop: 160x120
[naoqisim] [INFO ] Starting ALNetwork
[naoqisim] [INFO ] NAOqi is listening on 127.0.0.1:54010
```

Figure 22: Screenshot from Webots for NAO

4 Project

4.1 Introduction

I have started my project at the third week of my internship, after have learned usage of NAO and its softwares. All codes have written in Python and OpenCv for Python have been used for image processing. Bottom camera of NAO has been used default camera to capture frame. All frames have had BGR colorspace and frame rate has been 5 fps.

First two week, I have browsed sample projects on the internet. Moreover, aldebaran website has many projects that be benefical for NAO programming beginners.Taking a glance at Aldebaran documentation which released on website is good starting point for NAO programming.

4.2 Scenario

4.2.1 Overview

In my project, NAO has worked as a "garbage collector". It has tried to remove garbages (cylindrical shaped red objects) that have been "small" or "big" from an "area" that its sides have been fixed by yellow-black colored security line.

Main algorithm which I have used is here:

- 1 - Search an object in area for max searching time. If the object has been found, continue step 2; else goto step 7
- 2 - Get closer to object.
- 3 - Determine size of object (small or big).
- 4 - If it is big then goto step 5, else goto step 6,
- 5 - Act for big objects: Search human in order to carry object out of area. If human is not found then goto step 7 ;else give directions to human in order to carry object then goto 1.
- 6 - Act for small objects: Carry object by NAO's itself out of area. Goto step 1
- 7 - Finish

In implementation part, All steps will be explained detalied.

4.2.2 Working area

Working area background color is gray. NAO can distinguish easily it from objects and security line.

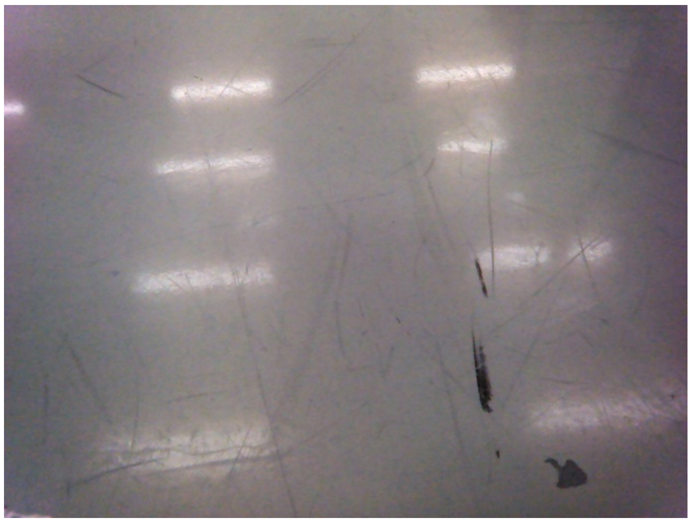


Figure 23: Empty Working Area

4.2.3 Objects

There have been two types of cylindrical red objects as garbages that have been in different sizes. Gradation between big and small objects has been not important and NAO have recognized them as same colored objects .

Big objects have had 5 cm radius and 50 cm height. They have not been able to carried by NAO itself due to their sizes. I have used rolled red funds carton to create big object.

Small objects have had 2.5 cm radius and 21 cm height. NAO have been able to take and carry them by itself. Small objects must have been solid since NAO should have applied pressure in order to hold firmly. Therefore, I have used cutten red pole noodles to create small objects.

4.3 Implementation

4.3.1 Searching an Object

Searching an object process is made by using image process via OpenCV for Python. NAO searchs red colored blob in captured frame. These red blobs mean they are objects. If there have been more than one objects. The closest one has been chosen as the target.

Since searching an object is only color based object detection, trouble might be happen in choosing target object if two object have stand same direction. Nao might be recognize both of them as one object.

In my scenerio, I have avoid such cases.

Steps which I have used are here and detailed function and library descripton are in Appendix part:

- Capture frame from bottom camera.
- Get contours list C of red blobs using by processImage(..) function from imageProcess.py
- Filter the C. Drop contours if their contour areas are smaller than 6000 in order to eliminate red blobs that occured because of bad light conditions.
- Get target object using by getTargetObject(C) function from closerToObject.py
- If no target is found, go forward and search object until reaching line. If line is detected, turn right.

4.3.2 Line Detection

Line detection is necessary for NAO to stay in working area and this process is purely colored based object detection. NAO can recognize line if it is 30 cm away from line. If it is closer than 30 cm, it cannot see line due to angle of view of bottom camera.

NAO should check if it reached line when searching an object and carring an object out of line. This process is done by capturing a *frame* from bottom camera and using lineDetector(*frame*) from goToLine.py. Detalied explanations of this this function will be found in Appendix part.

4.3.3 Getting Closer To Object

After NAO detects an object, it has to get closer to its target by walking and stops when it is nearly 30 cm far from object. NAO calibrates its direction by turning while walking.

Walking is made by setWalkTargetVelocity(*x*,*y*,*theta*,*frequency*) function in Motion API that provided by Aldebaran. This function sets NAOs instantaneous (normalized) step length and frequency, and thus control its velocity (direction and intensity) indirectly.

- *x* is the fraction of maximum step length in X axis direction. Range is [-1,1]. -1 means that going back with maximum step length, +1 means that go forward with maxiumum length step, 0 means that no movement in X axis direction.
- *y* is the fraction of maximum step length in Y axis direction. Range is [-1,1]. -1 means that go right with maximum step length, +1 means that go left with maxiumum length step, 0 means that no movement in Y axis direction.
- *theta* is the fraction of maximum step length in Z axis direction. Range is [-1,1]. -1 means that turn in clockwise direction with maximum step length, +1 means that turn in counter clockwise direction with maxiumum length step, 0 means that no turning movement.
- *frequency* sets step frequency of NAO. Its range is [0,1]. 0 means no movement even x,y,theta are not 0. If theta is 1, NAO has the fastest walk speed.

More detailed information about this function is can be found in Aldebaran Website.
Turning angle that used for calibrate NAO's direction is calculated by getTargetAngle(..) from closerToObject.py. This angle is used in setWalkTargetVelocity(*x*, *y*, *theta*, *frequency*) function as a *theta* to calibrate NAO.

4.3.4 Object Recognation

In this part, type of object is determined by image processing using by typeOfObject(..) function from Object.py. This function returns string "small" or "big" by checking area of the contour area of object. If the contour area is larger than 6000 then it is "big" else "small".

4.3.5 Action for Small Objects

After object recognation part, if the target object is "small". NAO should carry it by itself to out of area. There are several steps to do this job.

Firstly, NAO has to move toward to object and stop in front of object(nearly 10 cm far from object) since NAO can calculate distance from object and itself, it knows distance that it should move by using getXYPointsOfObject(..) function from graspObject.py.

Secondly, NAO crouches in order to take object. Then, it gets position to pick up object from ground. Taking position have been done by Animation Mode from Choragraphe.

Afterwards, NAO should grasp the object with its hands using by inverse kinematic solver. Firstly, it detects grasping point of object using by getXYPointsOfObject(..) again. NAO finds x and y distances of object respect X and Y coordinates by accepting origin itself. Radius of the small object is 0.035 m. As a result, grasping point for left hand and right hand is (x+0.035,y+0.035) and (x-0.035,y+0.035) respectively.

Next, NAO stands up and check whether grasping is successful or not using by **isGraspingSuccessful(..)** from graspObject.py. If it could not grasp object, goes back 30 cm and tries all steps in this part(Action for Small Object) starting from first one. Failure about grasping object may occur because of trouble about inverse kinematic solver of NAO provided by Aldebaran. Object must not be held less than 10 cm far from NAO. If it does, object cannot be token.

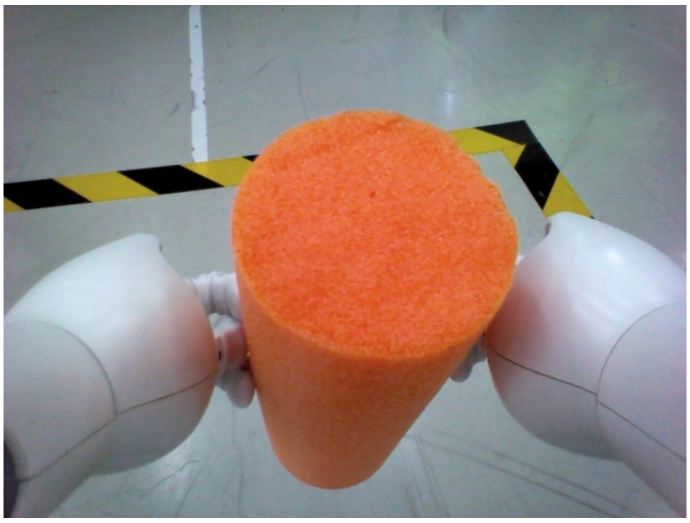


Figure 24: Successful grasping(from NAO's view)

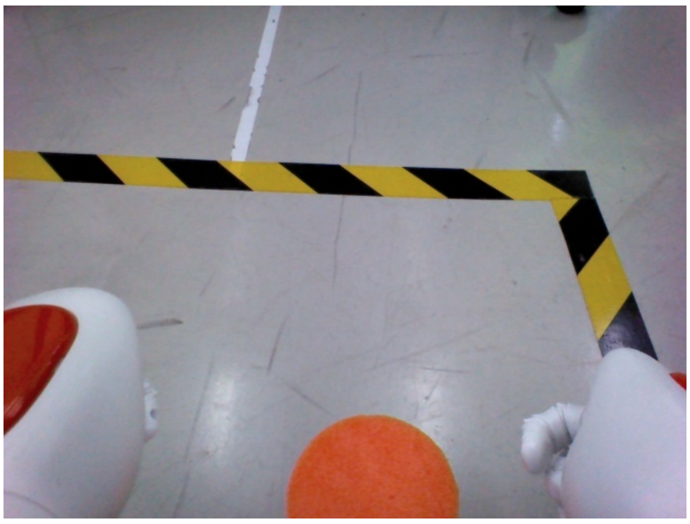


Figure 25: Unsuccessful grasping(from NAO's view)

Lastly, it tries to find a line in order to put object. This task is done by walking again. If there is an obstacle(another object) in front of NAO in carring process, it turns its direction to avoid from the obstacle. Turning angle is obtained by **getAngleToAvoidObstacle(..)** from goToLine.py. When it reaches line, it drops object and begins to search new target object.

4.3.6 Acting for Big Objects

If the object is ”big” type, NAO must search a human for his/her asistance in order to carry object. There are also several steps for this task.

Firstly, NAO turns one full rotation around itself and during rotation it searchs human using by its own HumanDetection module provided by Aldebaran. If NAO finds human, it interacts with human and says ”Help Me! Take This Box!”. Else, he says ”I cannot do this job.” and finishs task.

If human does not take object then NAO insists people. Checking whether the object is taken or not is done by **youReachTarget(..)** from closerToObject.py.

After human takes object, NAO gives directives to him/her in order human to follow itself and it tries to reach line as in Action For Small Objects part. When it reaches line, wants human to drop object in front of itself. Checking that human has dropped object in front of NAO is done by **youReachTarget(..)** function again. After human has dropped object, NAO begins to search new objects.

5 Appendix

There will be detailed explanation of python modules and functions which I have used in my project in appendix part. There are five python modules which each of them include several functions.

5.1 imageProcess module

This module is used for processing an image that captured from NAO's bottom camera. Image processing is done via OpenCV Python. Functions in this module are listed below.

5.1.1 processImage(img, lowerHue, upperHue, mode)

This function processes an image **img** and returned processed image.

- **img** is the BGR image captured by NAO.
- **lowerHue** is the lower bound of hue value for HSV image.
- **upperHue** is the upper bound of hue value for HSV image.
- **mode** is type of returned object. There are four types which be returned.

  If mode is HSV then hsv image converted from img is returned.

  If mode is THRESH then binary image that be filtered by boundry which determined by lowerHue and upperHue is returned.

  If mode is CLOSING then THRESH image is closed by using a kernel for better result. This result image is returned.

  If mode is CONTOURS then contours which obtained from CLOSING image are returned.

5.1.2 convertNAOImg2CV2Img(naoImage)

Frames that captured by NAO's camera are not be able used in OpenCV directly. They must be converted in BGR format. This function takes an image captured by NAO's camera (**naoImage**) and returns BGR image which can be used by OPENCV.

5.1.3 getMaxContour(contours)

This function takes a list of contours as the parameter and return the contour that has maximum area.

5.2 Object module

This module contains definition of a class named **Object** and function which determines type of an object.

5.2.1 Object class

This class is used for representing an object which detected by NAO. Classes members are below:

- **contour** is the contour which creates Object.
- **area** is the pixel area of the contour.
- **cx** is the column number of pixel which is center of contour's mass.
- **cy** is the row number of pixel which is center of contour's mass.
- **height** is the difference between row numbers of the topmost and bottommost pixels of contour.
- **height** is the difference between column numbers of the leftmost and rightmost pixels of contour.

5.2.2 typeOfObject(Object)

This function determines type of **Object** instance. If **Object**'s **area** is larger than 6000 than it returns "big" string else returns "small" string.

5.3 CloserToObject module

This module includes several functions that used in order NAO to getting closer to an object.

5.3.1 getDistance(cameraPosition6D, targetObject)

This function calculates real world distances between object and NAO robot. It takes two argument.

cameraPosition6D is 6 dimensional list containing [x, y, z, Wx, Wy, Wz] where Wx is the rotation around the X axis etc. Origin((0,0,0) point) is on the ground and between NAO's foot.

targetObject is the object that NAO tries to reach. It is instance of Object class.

As a result, euclidian distance(in meters) on the ground plane between NAO and object is retured.

5.3.2 getTargetObject(cameraPosition6D, contours)

This function takes two argument: cameraPosition6D and contours and returns target object which is instance of Object class.

5.3.3 getTargetAngle(Object)

This function is used for finding turning angle (in radians) to calibate NAO's direction. It takes an object as a argument and returns turning angle.

5.3.4 youReachTarget(frame)

This function takes captured frame as argument and returns whether NAO reaches target or not. Determining process is done by image processing. It returns true if NAO reaches target, else returns false.

5.4 graspObject module

This module includes several functions that used for grasping small objects.

5.4.1 verticalAngleBetweenRobotAndPixel(yPoint,knownAngle)

This function takes two argument. **yPoint** is the row number of selected pixel and **knownAngle** is the angle between camera axis of bottom camera and z axis of robot. The function returns an angle in radian which calculated using by trigonometry.

5.4.2 horizontalAngleBetweenRobotAndPixel(xPoint)

This function takes only one argument. **xPoint** is the column number of pixel. Returned value is an angle in radian.

5.4.3 coordinatesOfTargetObject(cameraPosition6D, targetObject, targetHeight)

This function takes three argument and calculate coordinates of an object according to Nao. cameraPosition6D, height of object and **targetObject** which is instance of Object class.

5.4.4 isGraspingSuccessful(frame)

This function used for checking whether small object is grasped successfuly or not. **frame** captured by bottom camera is the only argument. True or False boolean values are returned.

5.5 GotoLine module

This module includes several functions that used for line detection.

5.5.1 lineDetector(frame)

This function takes captured frame as argument and determine whether line is detected or not using by image process.

5.5.2 getAngleToAvoidObstacle(frame, securityDistance, cameraPosition6D)

This function takes three argument and return an angle to avoid an obstacle which in security distance.

6 Conclusion

During my summer practice, I have gained experince of studying with humanoid robot NAO and basic concepts of image processing. Before my summer practice, I thouht that work in academic life. Kovan Research Lab gives me chance to observe lab environment and academic life. After my internship, my idea related to academic life is strengthened. Now, I am more interested in robotics and image processing thanks to my internship.

7 References

[1] http://www.aldebaran.com/en/humanoid-robot/nao-robot
[2] http://kovan.ceng.metu.edu.tr/index.php/Main$_{P}age$
[3] http://www.aldebaran.com/en/humanoid-robot/nao-robot/research
[4] http://doc.aldebaran.com/1-14/family/robots/dimensions$_{r}obot.html$
[5] http://doc.aldebaran.com/1-14/family/robots/joints$_{r}obot.html$
[6] http://doc.aldebaran.com/1-14/family/robots/microphone$_{r}obot.html$
[7] http://doc.aldebaran.com/1-14/family/robots/video$_{r}obot.html$
[8] http://doc.aldebaran.com/1-14/family/robots/fsr$_{r}obot.html$
[9] http://doc.aldebaran.com/1-14/family/robots/sonar$_{r}obot.html$