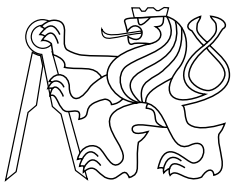
CENTER FOR
MACHINE PERCEPTION



CZECH TECHNICAL
UNIVERSITY IN PRAGUE

RESEARCH REPORT

ISSN 1213-2365

Nao Robot Applications Developed In Choregraphe Environment

Štěpán Křivanec, Alena Petráčková, Tran Thi Phuong Linh, Daniel Průša

{krivaste, petraal1, tranthip, prusapa1}@fel.cvut.cz

CTU–CMP–2010–21

December 9, 2010

Available at
ftp://cmp.felk.cvut.cz/pub/cmp/articles/prusa/Krivanec-TR-2010-21.pdf

The work was supported by EC project FP7-ICT-247525 HUMAVIPS. Any opinions expressed in this paper do not necessarily reflect the views of the European Community. The Community is not liable for any use that may be made of the information contained herein.

Research Reports of CMP, Czech Technical University in Prague, No. 21, 2010

Published by

Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University
Technická 2, 166 27 Prague 6, Czech Republic
fax +420 2 2435 7385, phone +420 2 2435 7637, www: http://cmp.felk.cvut.cz

Nao Robot Applications Developed In Choregraphe Environment

Štěpán Křivanec, Alena Petráčková, Tran Thi Phuong Linh, Daniel Průša

Abstract

We report results achieved during a student project that took place in July and August 2010. The goal of the project was to propose and implement applications to demonstrate abilities of a humanoid Nao robot. We chose Choregraphe as the main development tool. It is a programming software that allows the user to create and edit robot movements and interactive behaviors. We give a description of all implemented modules. We also summarize some technical issues that we had to overcome.

Introduction

The student project, which of results we describe here, was initiated thanks to the EC project HUMAVIPS. It stands for *Humanoids with auditory and visual abilities in populated spaces* and its ambitious goal is to teach humanoid robots how to focus attention on just one person in the midst of other people, voices and background noise. Nao robot, manufactured by Aldebaran company, is used as the main testing platform.

A team of three students was establish to work with the robot using Choregraphe programming environment. The implementations were done by customizing predefined Choregraphe modular units (so called “boxes“), as well as by creating own boxes containing a non-trivial code in Python.

The project had a positive impact on our Nao programming knowledge. We also became more familiar with the hardware.

Choregraphe libraries containing the applications are available in a SVN repository, which is placed on `ptak.felk.cvut.cz:/datagrid/humavips/repo/`
The libraries reside in folder *student_project_2010*.

The following table lists names of our main applications, together with their authors.

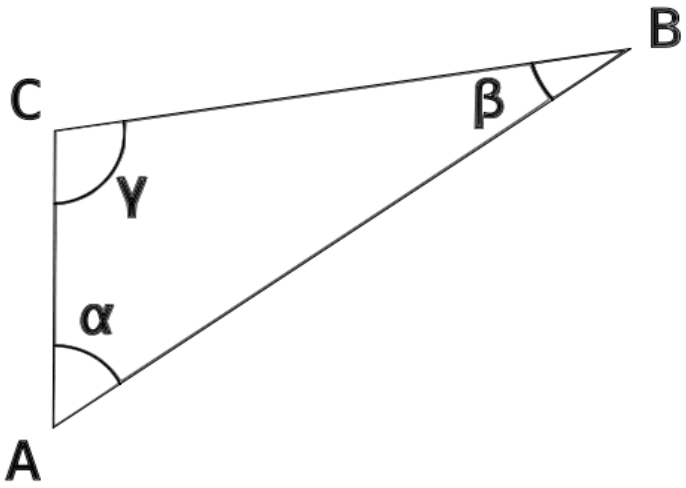
Author: Štěpán Křivanec

Naomark Distance

The goal of this task is to measure the distance from the camera to Naomark / human face using a face / mark detection implemented by API.

Problem description

I decided to use a triangulation method to get the distance. However, the robot does not provide stereo vision, so it is necessary to move the robot between two angle measurements.



By a simple calculation

$$c = b * \sin\gamma / \sin\beta$$

we get the distance from the Nao's bottom head position to the target. From the measurement, we know apha and gamma angles.

$$\beta = 2\pi - (\alpha + \gamma)$$

Naomark Distance box returnins the distance between the robot and object by foot.

Thats done by calculation

$$n = c * \sin\alpha,$$

where *n* is the distance by foot.

Implementation

The whole procedure is implemented in one 'box' called “Naomark Distance” using Choreographe environment. It is written in Python.

Description of functions used in Naomark Distance follows:

debugPrint – detailed debuging function used in in all functions. Used in: ctPoloha, headPoloha, zPoloha, zmer, mark, onImput_onStart.

dlog – second universal, less detailed, debuging function. Used in: ctPoloha, headPoloha, zPoloha,zmer, onInput_onStop.

moveLog – detailed debuging function used in functions that perform robot's motions: drep, stuj.

markLog – detailed degubing function used in detection functions: mark

ctPoloha – returns Top camera coordinates in SPACE_NAO

headPoloha – returs Head coordinates in SPACE_NAO

zPoloha – returns Top cameras 'z' coordinate in SPACE_NAO

zmer – returns distance between detected naomark and the robot

mark – returns mark detection result

Experimental practice

Positives:

Choreographe

- It is a powerfull tool for creating movement routines.

You can simply move real NAO like a puppet and capture the movement.

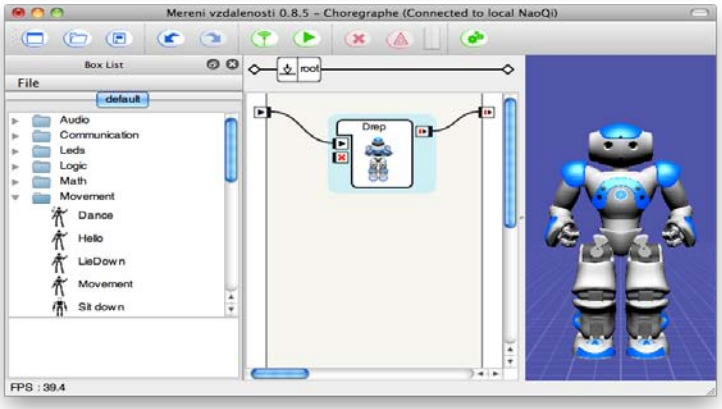
Search for „*Creating a movement with a real Nao robot to define the joints values*"

article in the documentation.

- It is easy to export movements to a Python or C++ code.

Procedure :

1) Open motion box



2) Right-click the timeline, > Export motion to clipboard

3) Select your desired output format

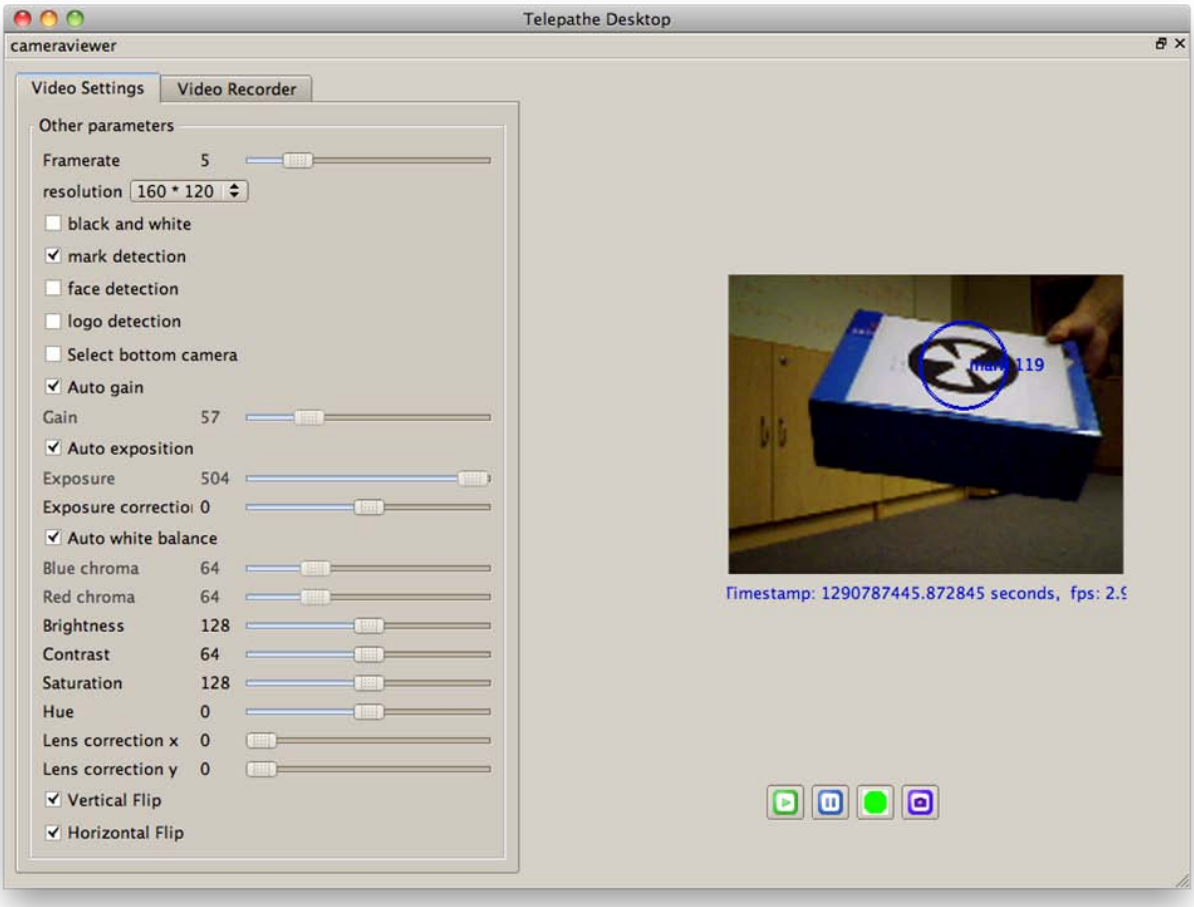
Negatives:

ALLandMarkDetection

- According to the documentation, list fields called SizeX and SizeY should contain size of the detected naomark in X and Y coordinates. But these values are equal, independetly on the detected shape of naomark.

- The documentation does not describe sufficiently the meaning of “shape” and “heading” values returned by naomark detection. Their meaning is unknown to our team.

Aldebaran was informed about this, but we have not got any response about this topic from them.



ALFaceDetection

- Same as ALLandMarkDetection. SizeX and SizeY do not return any usable values.

- “shape” and “heading” have unknown meaning.

Robot overheating

- The head overheats even in an air-conditioned room with a running fan pointing to NAOs head.

A new head was promissed by Aldebaran, bud has not been delivered yet.

- Joints are heating by putting a force on them for a extended period of time.

Joints overheating changes joints parameters, mainly stiffness.

For example: Overheating of the knees causes a robot‘s unstability while walking and, in general, it is followed by a robot‘s fall.

- It is possible to reduce the heating in joints by reducing time spent in critical positions

or by setting stiffness to 0 after each task.

An example of a critical position.

An example of a position in which the knee-motors will not overheat.

Knee stiffness set to 0.

Knee stiffness set to 1.

In the first image, knee motors are in tension and they are heating.

The second image shows no heating knee-motors.

Experimental results

- the ideal measured distance is in interval <0.8 ; 2.5> meters with a symetrical error of 1%

- it is technically imposible to measure smaller distances without tilting the head, witch this solution does not implent

- results of measurements for distances bigger than 2.5 meters are strongly influent by lighting conditions

- we require that there have to be only one naomark in the view, we do not handle a naomark absence neither a presence of multiple naomarks

Choregraphe

- It lacks features that programing environments should have and thus it is a bit inconvinient for coding. If this does not improve in a near future, I would suggest to find another IDE for developing code in Python.

- There is no way to close loaded boxlist. Aldebaran does not respond on this topic.

So just forget it. You cannot do it. At least in version 0.8.

Telepathe

- For Mac OSX version program, the streamed video window does not refresh itself propertly untill you move the mouse cursor.

This bug is known and Aldebaran promissed to fix it in the next release.

Documentation

- Old and outdated

- No one bothers tell you there which data type is actually returned by API functions.

As far as I know, every function returns a list even when it is returning just one value and it is obvious that it will never return more.

- Information about one topic is spread all around the whole documentation. Usually on 2 – 4 places.

- Use the coordinate system Naospace = 2, the others are inaccurate

WiFi connection

It was unstable for us, we decided to use the ethernet connection only.

Author: Alena Petráčková

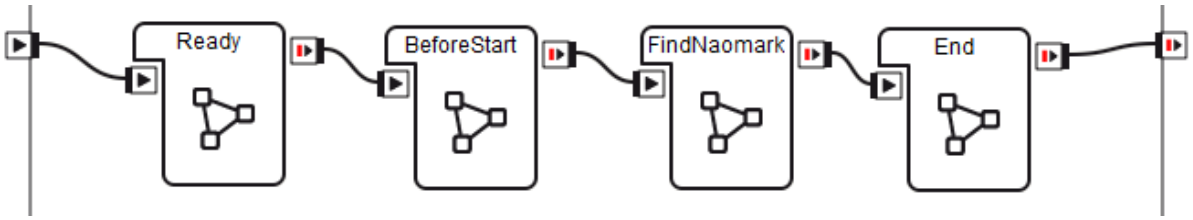
Finding Naomark

Target: NAO finds NaoMark to which it has to come, approaches it, stops and points its hand to it.

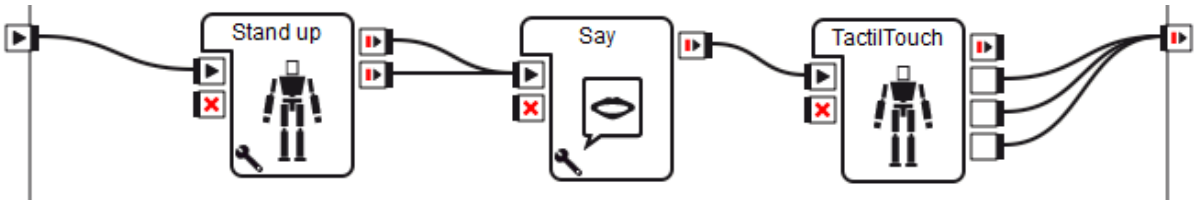
NAO starts and keeps turning on the spot. When it sees NaoMark it stops turning. It starts to go forward and every 20cm it slightly rotates the whole body to align its face to NaoMark. NAO stops when the sonar detects an obstacle 0.5m before it.

Implementation and program functions

The implementation consists of four Choreographe boxes. The boxes are performed in the sequential order as follows:

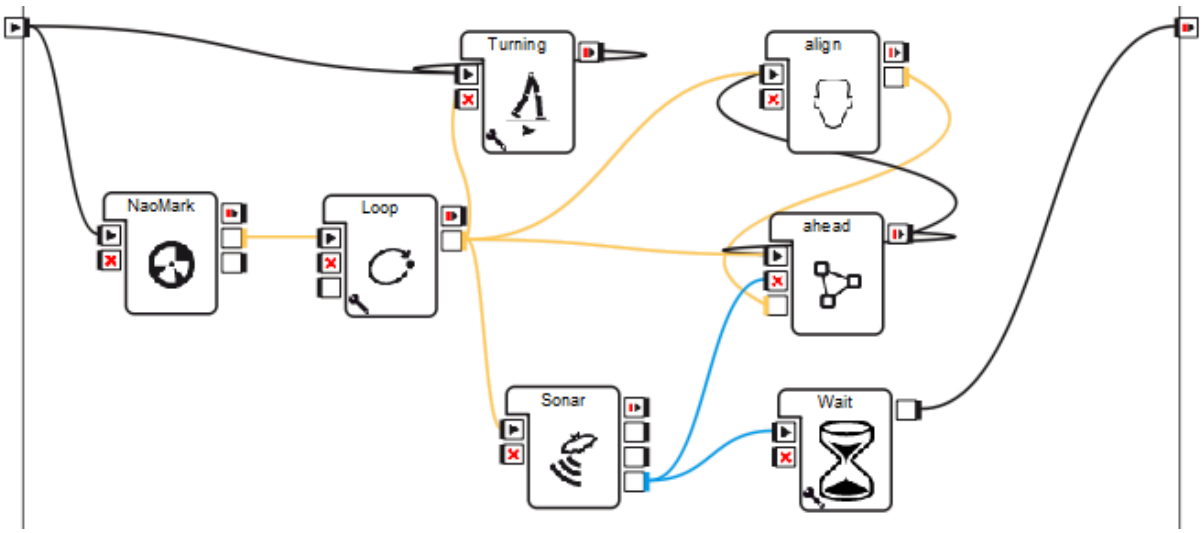


Ready: NAO stands up, it says “Hello, I am ready” and waits until the tactile sensor on the head is touched.



BeforeStart: At the same time, NAO starts discolorating eyes, lighting ears, scratching its head and it says “Where is NaoMark?”.

FindNaomark: At the same time, the robot starts rotating and finding NaoMark. The box “Loop” monitors that the signal from “NaoMark” comes only once. If NAO finds NaoMark, it stops turning, switches the sonar on and walks forward. Every 20 cm it aligns its face to NaoMark. It walks forward seeking an obstacle. When the obstacle is detected by the sonar then the robot stops in the distance approximately 0.5m before it.



Remark: The code is set to search for NaoMark number 170.

End: Finally, at the same time NAO starts discoloration of eyes and points by hand to NaoMark and says “Here is NaoMark”.

Known issues and limitations

Rotating of the robot is not very accurate. NaoMark could be lost from the sight sometimes.

Sometimes, the program does not finish, but the robot does nothing. In this case, it is necessary to stop it using Finish button in Choreographe. The reason is unknown to us.

Used functions from API

setWalkTargetVelocity()

walkTo()

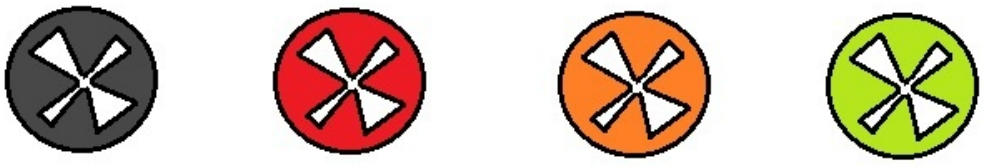
waitUntilWalksFinished()

Author: Alena Petráčková

Traffic Light

Target: NAO works as a traffic light. It performs commands “Stop”, “Wait” and “Start” in the dependence on NaoMark number that it sees. The three states are represented by eye’s colours, specific hand movements and spoken words.

Usage: Run the demo in Choreographe and wait until NAO is ready. Show NAO special NaoMarks to change its state. NaoMarks meaning is explained in the following Figure:

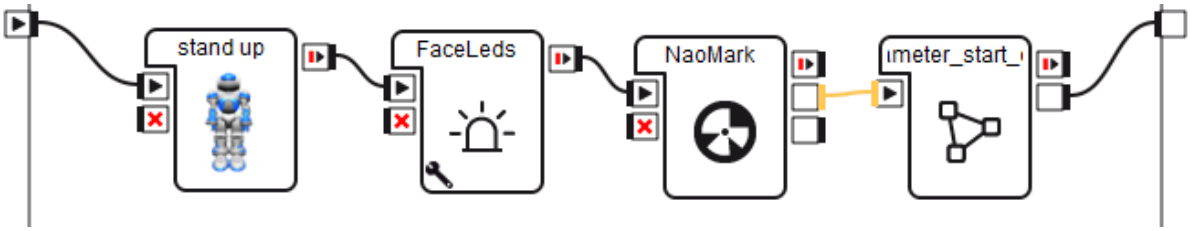


	Stop, wait, start	Stop	Wait	Start
Mark ID:	187	170	68	112

Remark: The NaoMark colour is not important for the detection, only NaoMark number (Mark ID) matters.

Implementation and program functions

The first layer of the implementation consists of the following Choreographe boxes that are performed in the sequential order:

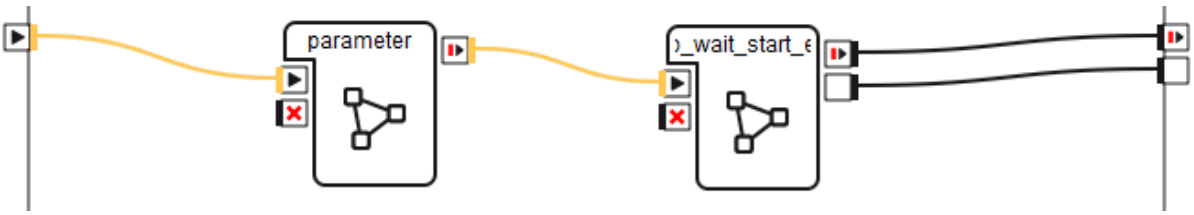


Stand up: NAO stands up. This is present, because the next motions involve the hands only.

FaceLeds: Eyes are coloured in pink. This informs the user that NAO is ready to work.

NaoMark: Finds NaoMark and returns NaoMark number.

Parameter_start_eyes:



Parameter: The input to this box is NaoMark number, which has been detected by NAO. The box maps NaoMark number to an integer from 0 to 3 and passes it next. The mapping is as follows.

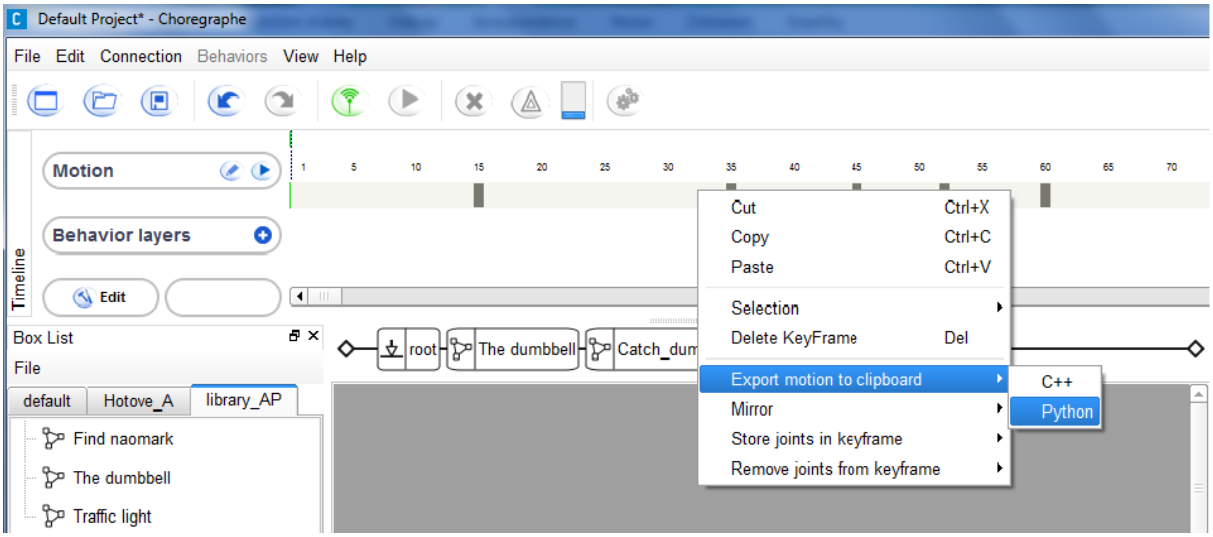
0 ... mark ID 187
1 ... mark ID 170
2 ... mark ID 68
3 ... mark ID 112

Stop_wait_start_eyes: Based on the input integral parameter, functions for a movement, change in eye colour and speech are called with appropriate parameters.

• Function sayPosition takes a word or phrase as the input parameter and passes it to AlTextToSpeech module, so that NAO pronounce it.

• Several functions are used to change the eyes colour. Their inputs are numeric and determine the desired colour.

• Functions Stop, Wait, Start serve to perform the hand motions generated by Choreographe. The following Figure demonstrates how to export an edited movement to a piece of Python code.



• Input of function onInput_onStrart is a parameter derived from the NaoMark number. Based on it, the function delegates to other internal functions what they should perform.

Everything is repeated because new signals still come from NaoMark detector.

Known issues and limitations

Program will not end because NAO still searches for NaoMark. It is necessary to stop it using Finish button in Choreographe.

Used functions from API

tts.postsay()

fadeRGB()

Authors: Alena Petráčková, Tran Thi Phuong Linh

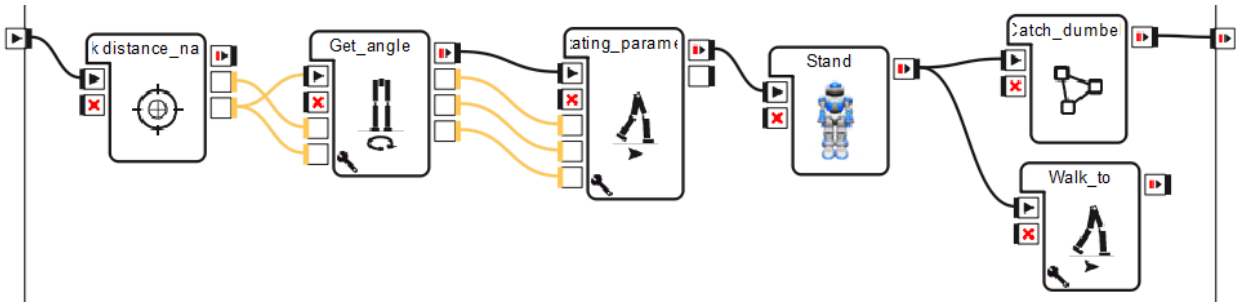
Dumbbell

Target: NAO has to lift a special dumbbell which of weight bars are covered by NaoMarks. In the beginning, it stands in some distance from the dumbbell and looks at it.

Task simplification: The scenario works well only if the robot sees both, the outer as well as the inner, NaoMarks on the dumbbell. The dumbbell must be at a place where the robot can see it when it stands. It is impossible to perform the task when the dumbbell is on the floor because of the fixed position of the head, which is required by the implementation of box “NaoMark distance_naomarkID”.

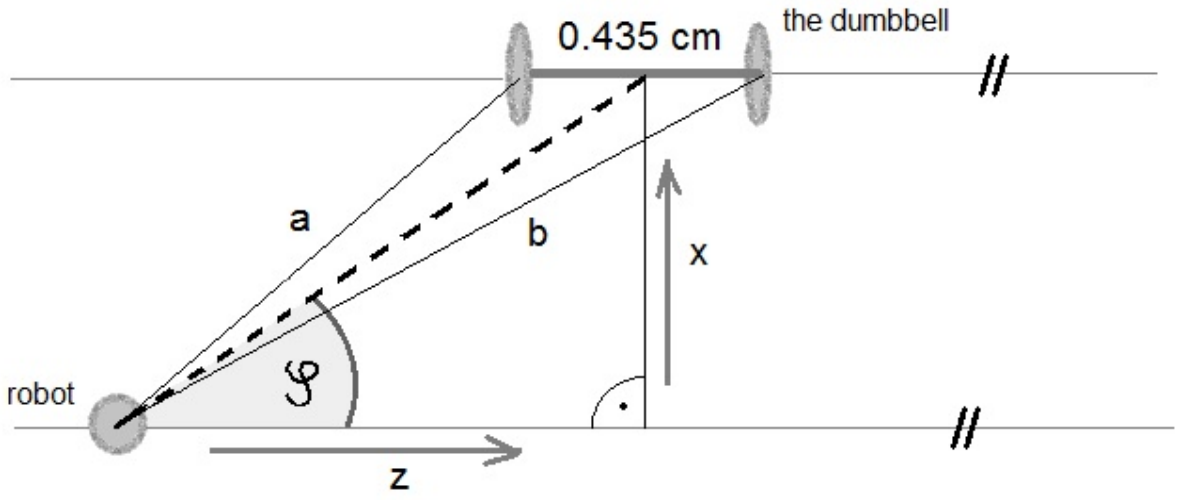
Implementation and program functions

The implementation consists of several Choreographe boxes. The boxes are performed in the order given by the scheme.



NaoMark distance_naomarkID: **s.** Documentation by Štěpán Křivanec for details on the implementation. Moreover, the original box has been modified as follows: NaoMark number is added as a parameter and the distance is measured from the mark with this number. There are two outputs, the distance from the dumbbell’s outer NaoMark (ID 170) and the distance from the inner NaoMark (ID 187) (**s**. distances a and b in the Figure bellow the next paragraph).

Get_angle: The box has two inputs - distances from NaoMarks number 170 and 187. The body of this box performs a mathematical calculation of the distances and the robot rotation angle. There are two distances and one angle on the output (**s**. angle φ and distances Z and X in the following Figure).



Rotating_parametr: Inputs are two distances and one angle taken from Get_angle box. The robot turns its body so that its forward motion is parallel to the axis of the dumbbell. Then, NAO walks the first distance, rotates its body by 90 degrees and it walks the second distance.

Stand: The robot stands up.

Boxes Catch_dumbbell and Walk_to start at the same time. The robot holds up the hands to take the dumbbell and walks 10cm straight. It lifts up the dumbbell above the head then.

Known issues and limitations

Currently, the program works only for the left side. (**i.e.**, in the robot's view, the outer part of the dumbbell (ID 170) is on the left with respect to the inner part (ID 187).)

The measurement of the distance is not exact. It causes that calculations cannot be done accurately. (**s.** Documentation by Štěpán Křivanec)

NAO unexpectedly kneels and stands up during its walk. This problem is somehow related to NaoMark_distance measurement box.

Used functions from API

walkTo(), waitUntilWalksFinished()

Author: Tran Thi Phuong Linh

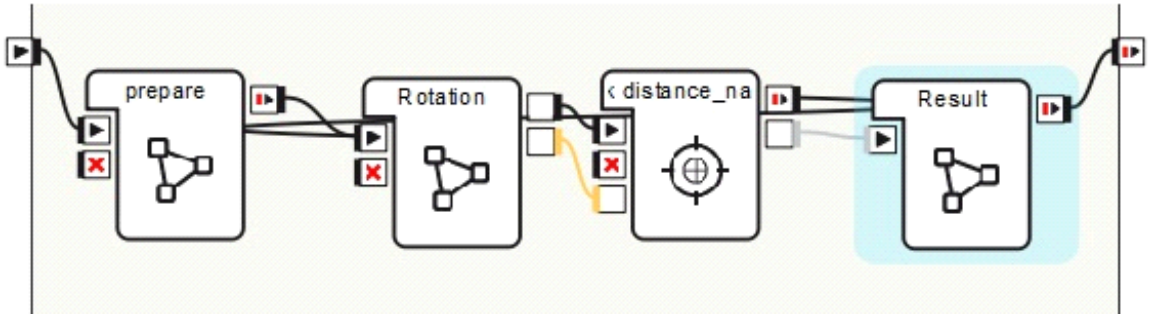
Count Naomarks

Target: The robot looks around (turning its head and body) and detects all NaoMarks which are visible from its position.

At the end, it says how many NaoMarks it saw and shows that one that is closest. Ie., it turns towards this NaoMark and points to it by the hand.

Implementation and program functions

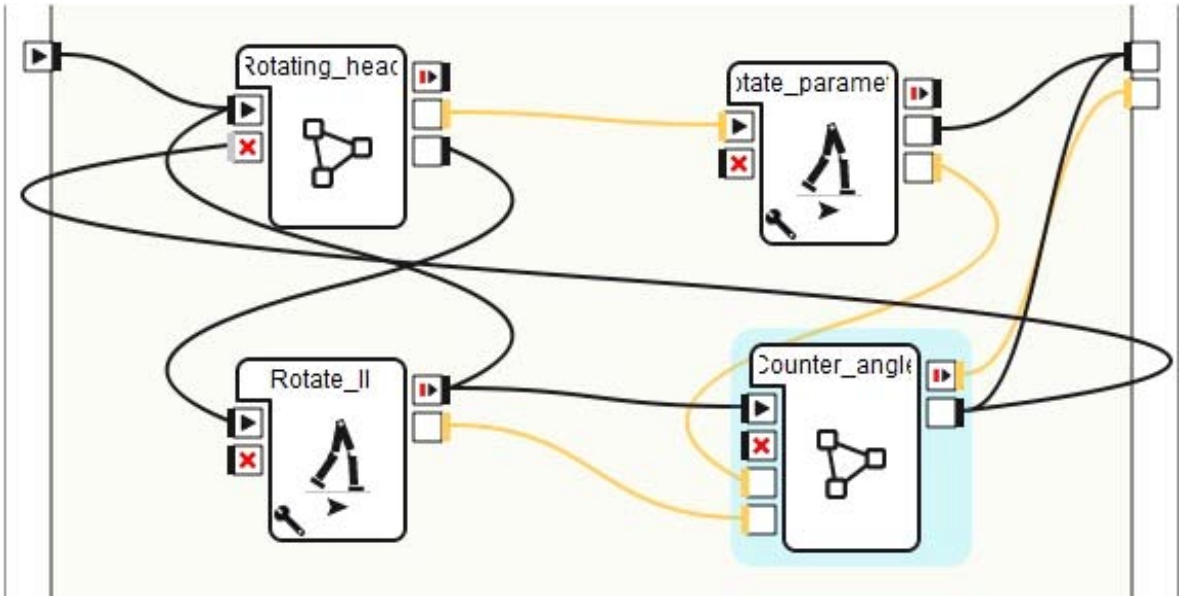
The implementation consists of four Choreographe boxes. The boxes are performed in the following order:



Prepare**:**

- Inserts to AlMemory list ID_number. This list records NaoMarks that the robot saw and it is an empty list at the beginning.

Rotation**:** The robot rotates to find NaoMark.



Rotating_head: The robot's head rotates to find NaoMark.

- If it finds some NaoMarks, its head will stop rotating and turns back to the beginning position.Then the output onOutput is activated with a parameter which is the rotation angle.

- If it does not find any NaoMark, the output noMark is activated. Then, box Rotate_II is started.

Rotate_parameter: When the robot finds some NaoMark, its body rotates towards to this detected NaoMark by the rotation angle.

- Input is the rotation angle.

- Output starts box Distance_NaoMark.

Rotate_II: When the robot does not find any NaoMark, its body rotates by angle 1.9 radians.

- Output activates box Rotating_head to find NaoMark again.

Counter_angle: Calculates the sum of the angles by which the robot rotated. When the sum of the angles is bigger than 6.28 radians, the robot stops rotating and box Result is started.

- Intput: the rotation angles from box Rotate_parameter and Rotate_II.

- Output: the sum of the angeles.

Distance_NaoMark**: s.** The documentation by Štěpán Křivanec for the basic variant of the box.

The box has been modified as follows:

- Results consisting of NaoMark numbers and their distances are saved to a list.

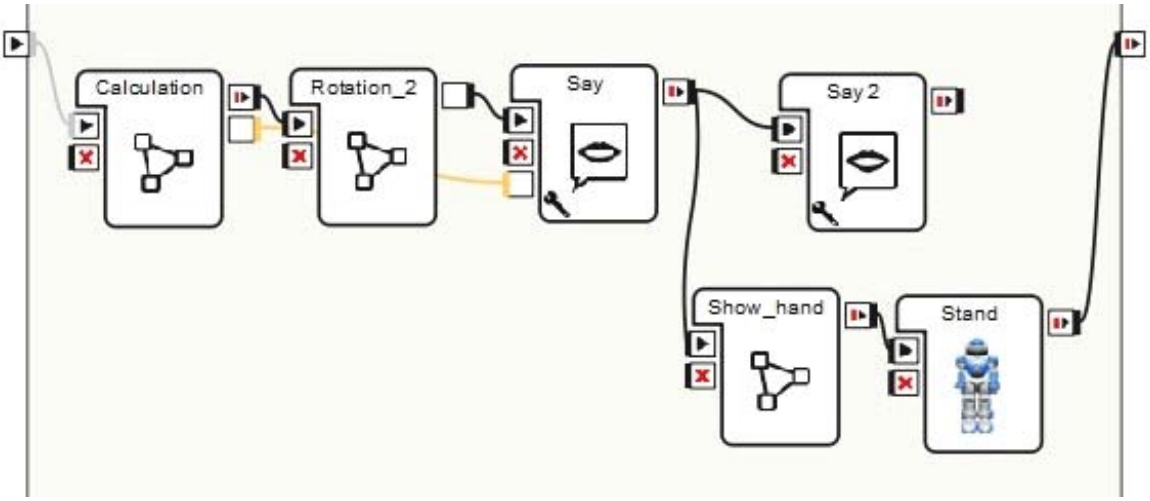
- Sometimes the distance measurement is done badly and it has to be repeated.

- List ID_number of NaoMarks which the robot saw is inserted to AlMemory.

- If the sum of the angles is at least 6.28 radians, the box Result is activated with the result parameter.

- If the sum of Angeles is less than 6.28 radians, the box Rotation is activates and the robot keeps on seaching NaoMarks.

Result**:** The robot counts the number of NaoMarks and detects the nearest one.



Calculation**:**

- Input: list containing NaoMark IDs and their distances.
- Compares distances of NaoMarks and finds which NaoMark is the nearest.
- outputs: the number of NaoMarks, Id of the nearest NaoMark.

Rotation_2**:** The robot rotates to find the nearest NaoMark.

- This box is similar to box Rotation.
- Input: Id of the nearest NaoMark.
- The robot rotates towards the nearest NaoMark.

Say**:** The robot says the number of NaoMarks it has seen.

Say_2 and Show_hand: The robot says "This is the nearest mark" and points to the nearest NaoMark by the hand at the same time.

Known issues and limitations

When the robot's head rotates, the robot often sees NaoMarks badly with wrong IDs.

The measurement of distance is not exact and sometimes it produces mistakes.

The program will run faster if robot does not have a cable in its head because of the rotating angle limit. However, it runs well even with the cable.

Author: Tran Thi Phuong Linh

Walk with Sonar

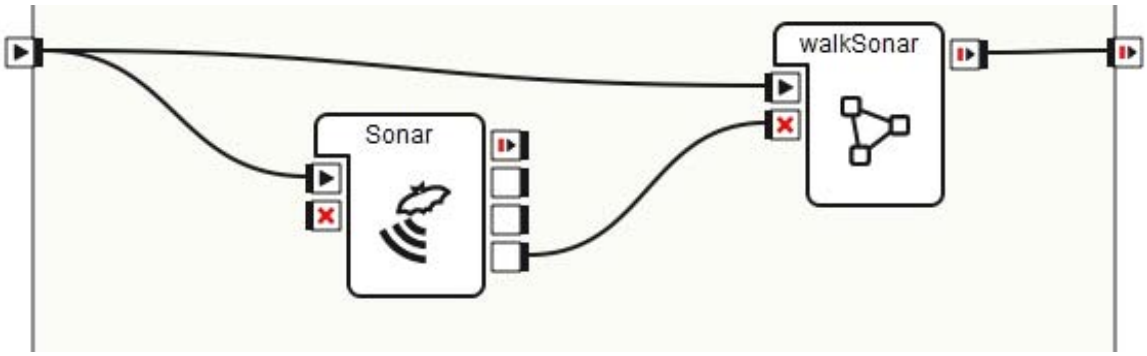
Target: Program in Choreographe the following robot behavior:

1. The robot starts to move in a straight line (use the setWalkTargetVelocity (...)).

2. If the robot detects an obstacle by the sonar closer than 40 cm, the robot stops walking.

Implementation and program functions

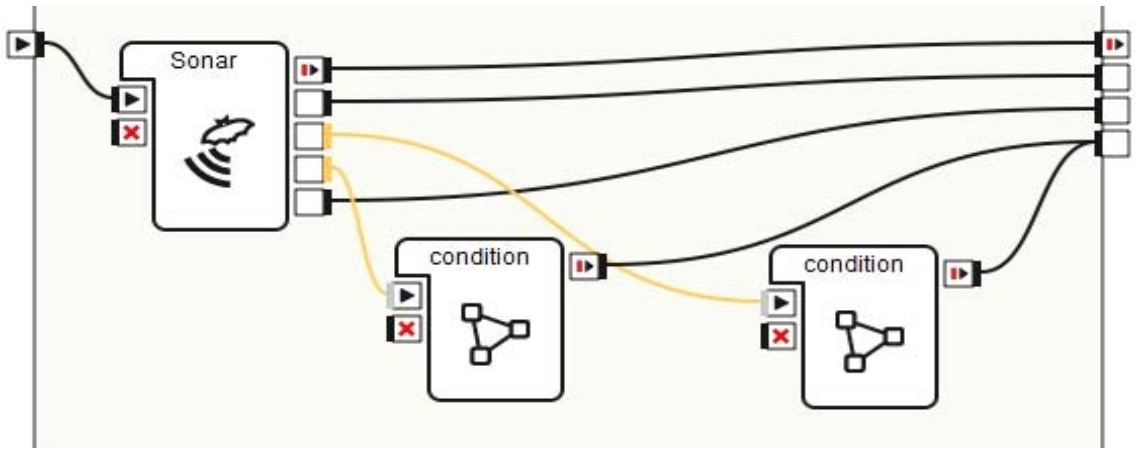
The implementation consists of the following Choreographe boxes. The boxes are performed in the following order:



walkSonar: The robot walks straight until it meets an obstacle.

The robot starts to move in a straight line (function setWalkTargetVelocity(...) is used, it is provided by ALMotion module). When input onStop is actived, the robot stops walking.

Sonar:



Sonar: This is a box in the default Choreographe library - directory Sensors.

condition: A Condition on the distance to the detected obstacle. If it is less than 40 cm, output onObstacle is activated and box walkSonar stops. Then, the robot will stop walking.

Known issues and limitations

Errors from ALMemory module are unexpectedly reported in the Choreographe log, but the program still runs well.