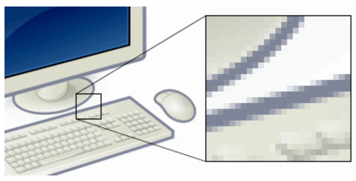
# DIGITAL IMAGES AND PIXELS

You may have used a digital camera and heard of phrases such as "10 megapixels". What is a megapixel? A megapixel is 1,000,000 pixels. What is a pixel then? 'Pixel' stands for picture element. A pixel is the smallest part of an image or screen. In a digital image or photo, a pixel is the smallest area that is a single color. The figure below is taken from the Wikimedia Commons. It shows an image of a computer, with a zoomed-in section of the individual pixels.



Similarly, a digital photo is composed of pixels. Thus, photos taken by a "10 megapixel" camera have 10 million pixels. Every digital camera has a sensor that converts the light coming in through the lens into digital signals. These digital signals are then used to create the pixels of the image.

The NAO has video cameras that do the same thing. However, the NAO's camera doesn't take a single image when a button is pressed. Instead, it takes a video - a continuous stream of images. The NAO's cameras are capable of a *resolution* of 1288 pixels by 968 pixels, at a *frame rate* of 30 frames per second (fps).

The resolution of an image refers to how many pixels there are along the length and breadth of the image. So, for the NAO's camera, the resolution is 1288 by 968. This means that the images are 640 pixels wide, and 968 pixels high. The frame rate refers to how many images are taken per second. So, the frame rate of 30 fps means that the camera takes 30 images every second.

# COMPUTER VISION

From a single digital image, we humans are able to identify the objects within it. For example, when presented with a picture of a beach, we can readily say that it shows a beach. We can also point out the sand, sea, clouds, and trees in the image.

For a computer, this task, known as computer vision, is not easy at all. Firstly, the image is entirely made up of pixels. This means that all the computer has are a series of numbers that represent the colors of each pixel. Given an image of a cup on a table, it is hard for the computer to tell that one pixel is part of the cup while the next pixel is the table. Also, the computer must have some concept of "cup" and "table" in its memory. As humans, we have a large memory, or database, of objects that we know and have interacted with. This database helps us to identify objects just from a quick, small view of it.

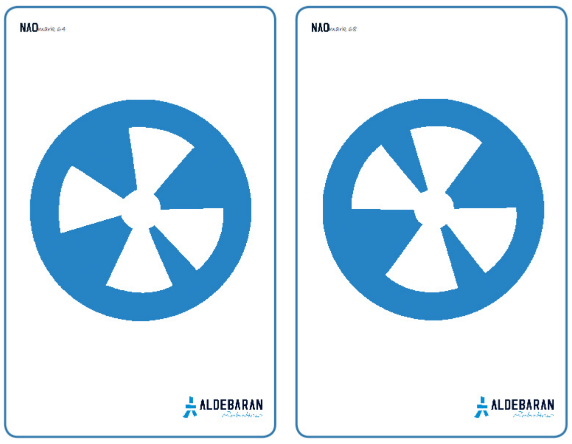
In computer vision, the algorithm first finds *features* in the image. Features are distinguishing parts of an image that aid the computer vision algorithm in deciding what object is present. Typical features include edges and texture. Edges occur because objects tend to look different from the background. Some objects tend to have uniform textures. Another feature that usually comes to mind is color. However, it is difficult for an algorithm to use color as a feature. This is because the same color looks very different under various lighting conditions. In most situations, our eyes are capable of adjusting for the color of the light source so we can tell if an object is red, but computers are unable to do so robustly.

Another aspect of features used in computer vision is that a feature should be *scale-invariant*. Invariant means "never changing". So, a scale-invariant feature means that the same feature is detected regardless of how big or how small the object is in the image. A common feature set used in computer vision is the scale-invariant feature transform (SIFT) feature set. We will not describe the details of what SIFT features are. The main idea is that these features are extracted from an image. They are then used to detect the objects in the image. One method is to have a database of images that are labeled. Given a new image, the features of the new image are compared to the features of images in the database. The object in the database with the closest features is then chosen. For example, suppose there is an image of a cup with SIFT features A, B and C. If the new image also has SIFT features A, B and C, then it is likely that the new image is an image of a cup.

## BASIC TASK
## NAOMARK-CONTROLLED ROBOT

IN THIS EXERCISE, WE WILL BE USING THE NAO TO PERFORM COMPUTER VISION. IN YOUR DVD, YOU WILL FIND A FILE CONTAINING NAOMARKS. NAOMARKS ARE CIRCULAR DESIGNS THAT LOOK LIKE THE FIGURE BELOW:

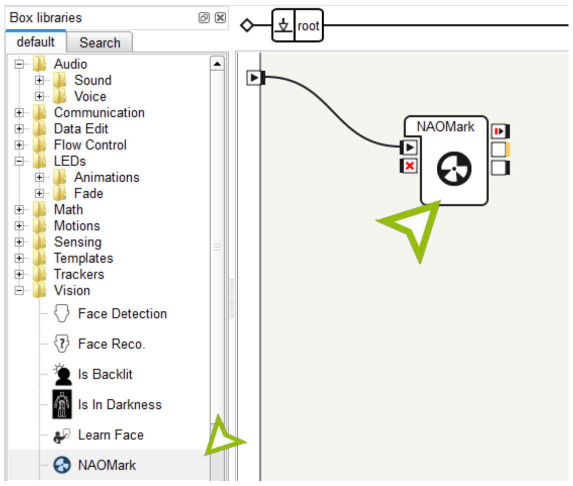
Choregraphe comes with an automated algorithm that detects these NAOMarks. The program returns their identification number, which is located on the top left of each card.  The algorithm detects the unique shape of the NAOMarks, and divides them up based on the sizes of the white and blue regions within the mark.



We will be using these NAOMarks as input to control the NAO's actions. In the earlier modules, we created a voice-controlled robot. In this exercise, we will build a robot that walks based on visual cues.

## 01/

Create a NAOMark box from the Vision category, and connect it as shown below.



The NAOMark box has two outputs. The first (middle box on the right) returns the identification number of the NAOMark detected (if any). The second (bottom-right box) triggers if no NAOMarks are detected.

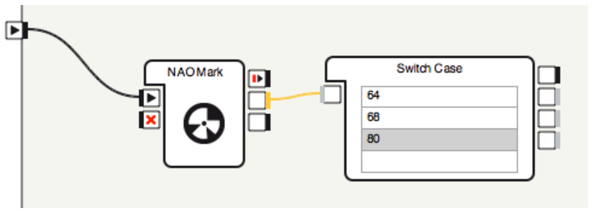
## 02/

Try running the behavior as is. Print out 3 NAOMarks, and show it to the robot. You should see the NAOMark number appear in the middle-right box of the NAOMark box in Choregraphe.

Hint: you may notice that other numbers sometimes get detected as well. For example, NAOMark 64 may be detected as 64, and sometimes 79 or 127. Make sure the three NAOMarks you choose look different from one another. So, if you picked NAOMark 1, 2 and 3, then ensure that NAOMark 1 does not ever get detected as 2 or 3.

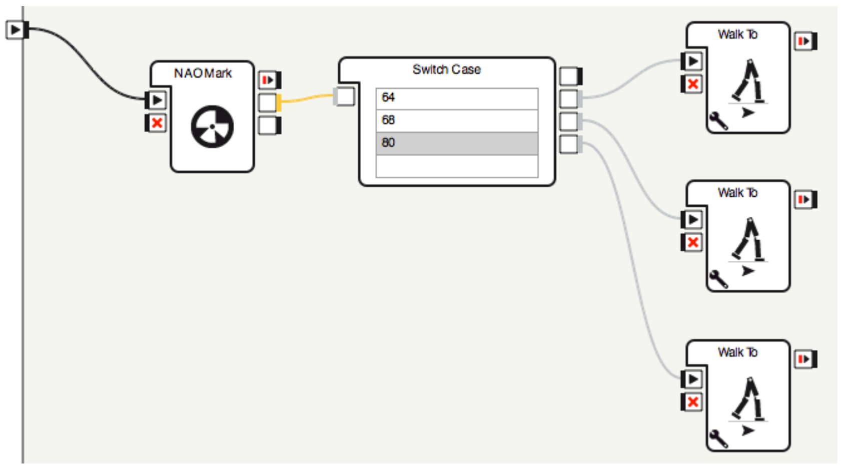
## 03/

Connect a Switch Case box to the output of the NAOMark box. Within the Switch Case box, enter the identification numbers of the three NAOMarks that you chose. In the figure below, we used NAOMarks 64, 68 and 80.



## 04/

When a particular NAOMark is detected, we want the robot to perform a unique action. Drag three Walk To boxes and connect them to the Switch Case box, as shown below.



## 05/

For each of the three Walk To boxes, enter different values for x, y and theta, so that the actions taken for each NAOMark is unique.

## 06/

Run the behavior. Show the different NAOMarks to the robot, and ensure that it performs the correct action for each mark.

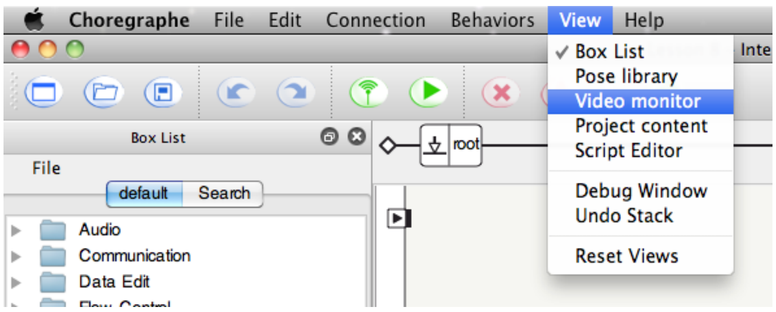
# INTERMEDIATE TASK
# OBJECT RECOGNITION

–

IN THE PREVIOUS EXERCISE, WE DETECTED NAOMARKS WITH AN ALGORITHM IN-BUILT INTO CHOREGRAPHE. IN THIS EXERCISE, WE WILL EXPLORE DEFINING AND DETECTING GENERAL IMAGES.

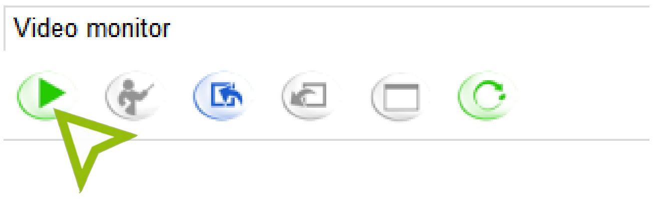
## 01/

First, we need to create a library of known objects for the NAO. To do so, connect to the robot in Choregraphe, and then click View → Video Monitor in the top menu bar. The window below should pop up. Click the New Vision Recognition Database button if it is not grayed out.



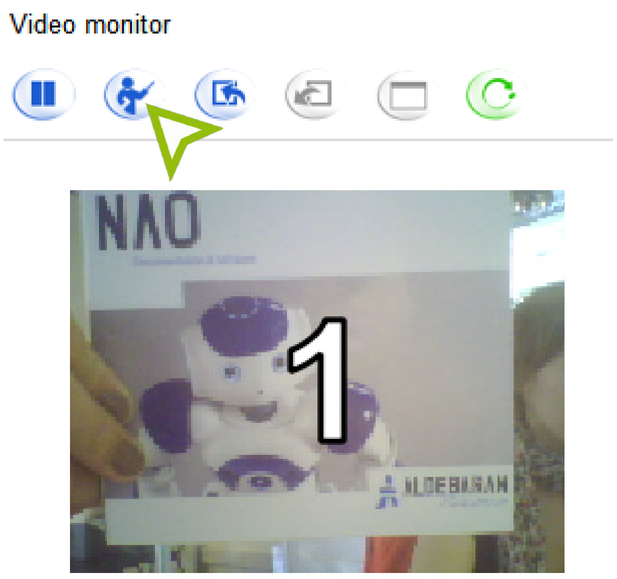
## 02/

Click the play button in the Video Monitor (not the "play behavior" button in the Choregraphe toolbar, although it looks the same), and you should see a video stream from the NAO. Move your hand in front of the NAO's face and ensure that you are streaming the video correctly. The NAO has two cameras, so check both of them to see which camera the NAO is using. You can switch the cameras using the Select Camera box under the Vision category.



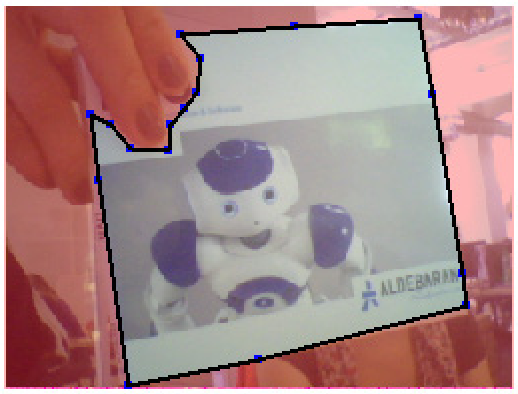
## 03/

Now, place the object you want the NAO to learn in front of the camera, and click the learn button. A countdown will appear on the Video Monitor, after which a single image will be paused and shown in the window.
Hint: For this exercise, we recommend using a flat object with pictures, such as the NAO DVD sleeve that is included in the NAO package. Other alternatives could be the cover of a book, or a page from the newspaper.



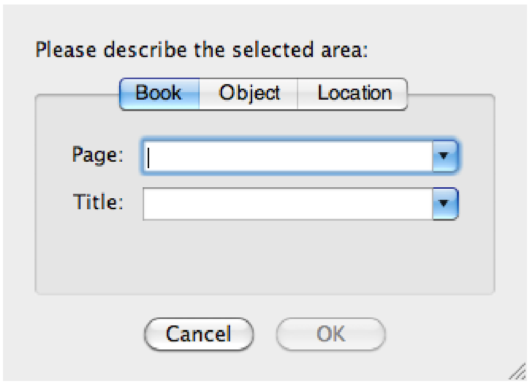
## 04/

You can now click on the border of the object to trace its outline. To end the trace, click on the first point of the border again. The features of the object will then be extracted within this outline and saved.



## 05/

Enter a name for the object, and the features will be associated with this object in the local Choregraphe database.

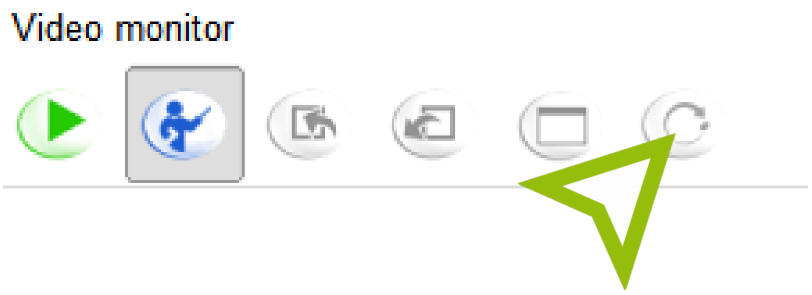


## 06/

If you want to learn more objects, you can repeat steps 2 to 5.

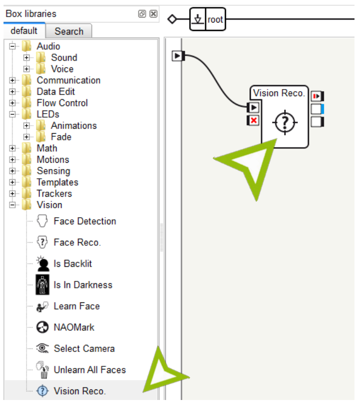
## 07/

Before we can do object recognition on the NAO, we need to send it the database of objects that we have created. Click the right-most button to send current vision recognition database to NAO.



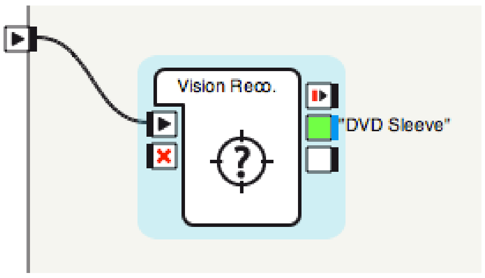
## 08/

Now that the NAO has a database of objects and their features, we can perform object recognition in Choregraphe. To do so, drag a Vision Reco box under the Vision category.



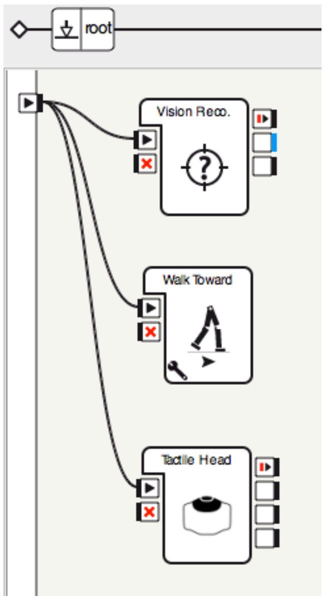
## 09/

Start the behavior, and place the object in the NAO's camera view. Ensure that the object gets detected, and the Vision Reco box returns the object's name in its output box. If the object does not get detected, repeat the steps above to create a new database and send it to the NAO.



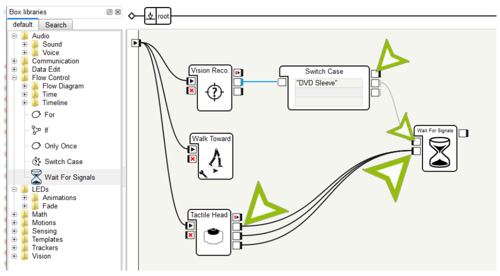
## 10/

We will now use the Vision Reco box to have the NAO walk until it sees an object that it recognizes, and is touched on the head. We will use a Walk Toward box, from the Motions category, and a Tactile Head box from the Sensors category.



## 11/

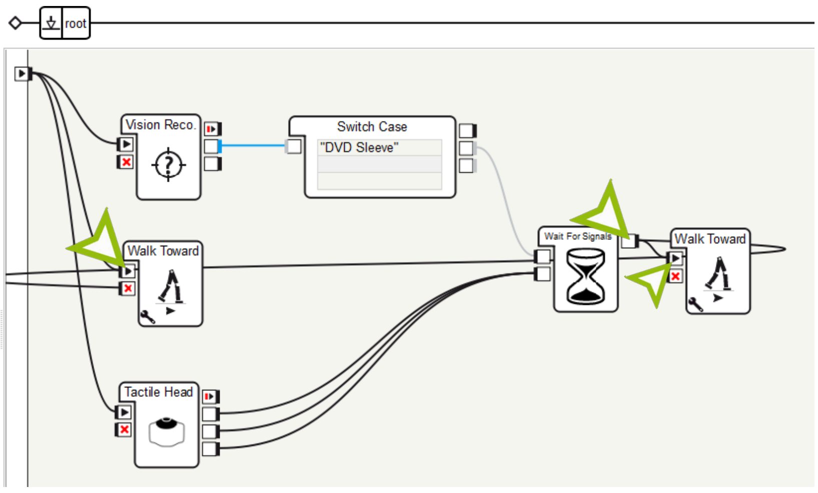
We want the NAO to stop walking when it sees an object it recognizes *and* a head sensor is touched. To do so, add a Switch Case box to check the output of Vision Reco box, and the Wait For Signals box from the Flow Control category. Connect the output of the Switch Case box to one of the inputs of the Wait For Signals box, and connect the outputs of the Tactile Head box to the other input of the Wait For Signals box.



The Wait for Signals box waits for both of its inputs before triggering its output - it performs a logical-AND operation. A logical-AND checks that all its inputs are true, e.g., true AND true is true, but true AND false is false, false AND false is false.

## 12/

When the Wait for Signals box triggers, it means that the desired object was detected, and the head sensor was touched. In this case, we want the robot to stop walking. To do so, drag another Walk Toward box, and connect it to the output of the Wait for Signals box. Set the parameters of the Walk Toward box to have x, y and theta set to 0. Also, connect the output of the Wait for Signals box to stop the previous Walk Toward box.



## 13/

Run the behavior. Check that the NAO starts walking, and stops only when the object is detected, and the head sensor is touched.

# INTERMEDIATE TASK
# OBJECT RECOGNITION

–

WE MADE THE NAO PERFORM OBJECT RECOGNITION ON A FLAT OBJECT LIKE A DVD SLEEVE IN THE PREVIOUS EXERCISE. HOWEVER, MANY OBJECTS IN OUR ENVIRONMENT HAVE COMPLEX SHAPES, AND LOOK DIFFERENT FROM DIFFERENT PERSPECTIVES. FOR EXAMPLE, A MUG LOOKS DIFFERENT WHEN YOU LOOK AT IT FROM THE SIDE COMPARED TO LOOKING AT IT FROM ABOVE. A BOOK LOOKS DIFFERENT DEPENDING ON WHICH PAGE IT IS OPEN TO.

## 01/

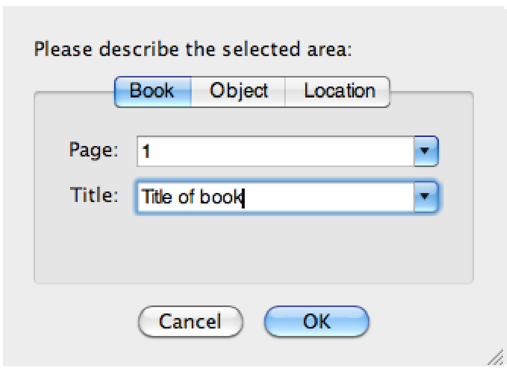
For this exercise, we will use a complex object such as a book or a brochure, that either can change its shape (like opening a book) or has a complex shape (like a mug).
Hint: use an object with multiple colors or a complex texture.

## 02/

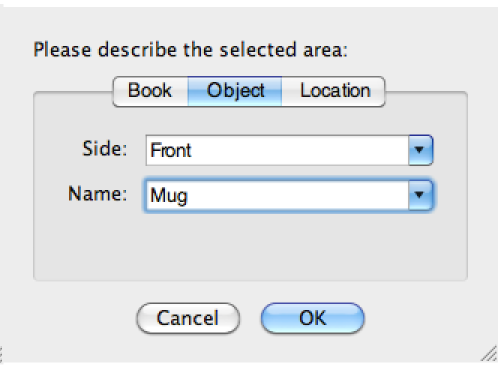
Create a new vision recognition database in Choregraphe (refer to step 1 of the previous exercise).

## 03/

Repeat steps 2-5 of the previous exercise to save different images of the object. However, besides giving the object the same name, also fill in the field on the page of the book, or side of the object.


Please describe the selected area:

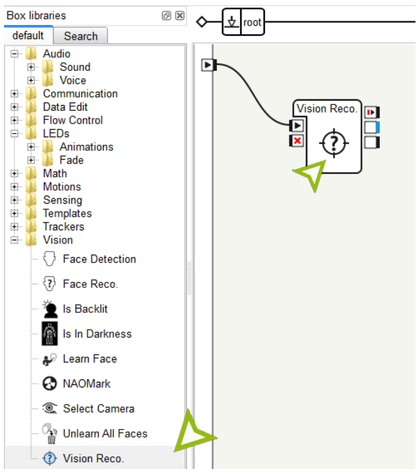
Page: 1
Title: Title of book


Please describe the selected area:

Side: Front
Name: Mug

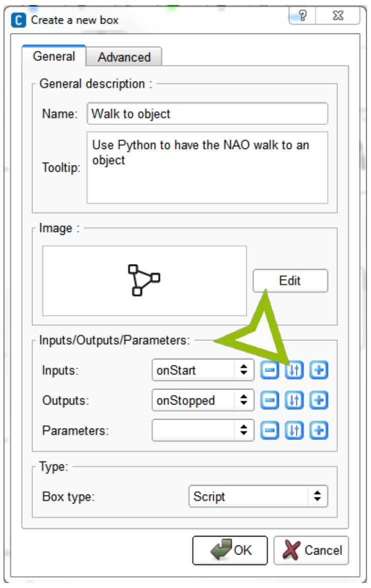
## 04/

Drag out a Vision Reco box, and run the behavior. Ensure that the object can be detected from different distances and angles.



## 05/

Create a custom Python box and connect it to the output of the Vision Reco box. Change the input type of the Python box to be String, instead of "Bang".



Name: Walk to object

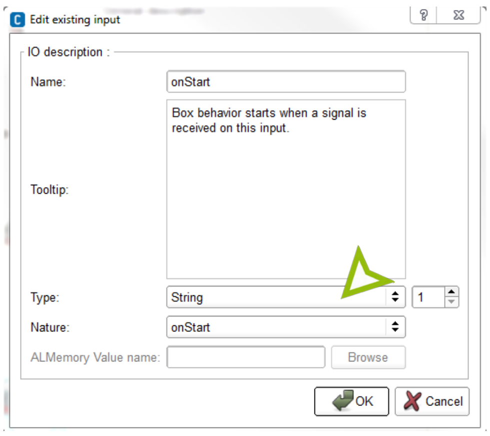
Tooltip: Use Python to have the NAO walk to an object

Inputs: onStart

Outputs: onStopped

Parameters:

Box type: Script



Name: onStart

Tooltip: Box behavior starts when a signal is received on this input.

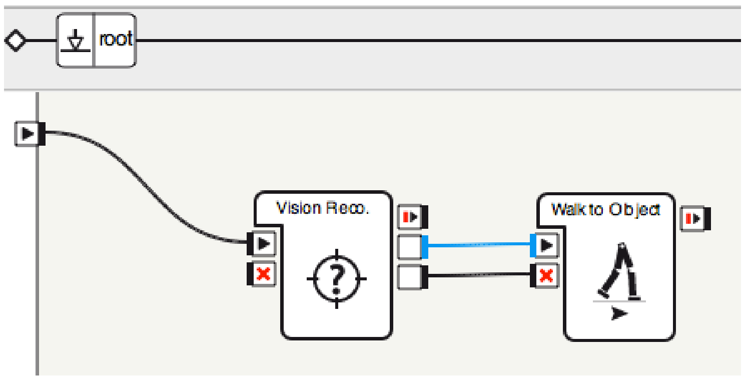
Type: String 1

Nature: onStart

ALMemory Value name:

## 06/

Connect the outputs of the Vision Reco box to the Python box, such that onStart is triggered when an object is detected, and the x is triggered when no object is detected.



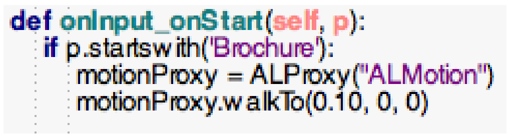
## 07/

When the object is detected, the Vision Reco box returns the name of the object as well as the side/ page. For example, page 1 of the book would be returned as a string "book 1", and page 2 would be "book 2". For the purposes of this exercise, we don't distinguish between the pages and sides and only care that the book was detected. However, we do care that the book was detected and not the mug, if there are multiple objects in the database.

To do so, we can check that the input starts with the name of the object, e.g., "book 1" and "book 2" both start with "book".

## 08/

Enter the following Python code into the custom Python box. The startswith function, as its name suggests, return true if the string does start with the prefix you provide. In the example below, we check that the string p starts with the prefix "Brochure". So, inputs such as "Brochure 1",



```
def onInput_onStart(self, p):
    if p.startswith('Brochure'):
        motionProxy = ALProxy("ALMotion")
        motionProxy.walkTo(0.10, 0, 0)
```

"Brochure 2", "Brochure front" would be accepted, but "Mug top" and "Mug side" would not.

Thus, when the object is detected, we use the motionProxy to walk the robot forward 10cm. Otherwise, the behavior doesn't do anything.

## 09/

Run the behavior. Place the object within the camera view of the NAO, and it should walk towards it. Once the NAO is close to the object, it will be outside the camera's field-of-view and not be seen by the NAO. As such, it will not be detected and the NAO will no longer walk forward.

**Warning**: be careful that the NAO does not walk over the object. It may cause damage to the object, the NAO, or both.