Introduction to Marvin, the Nao Humanoid Robot

Comp 380

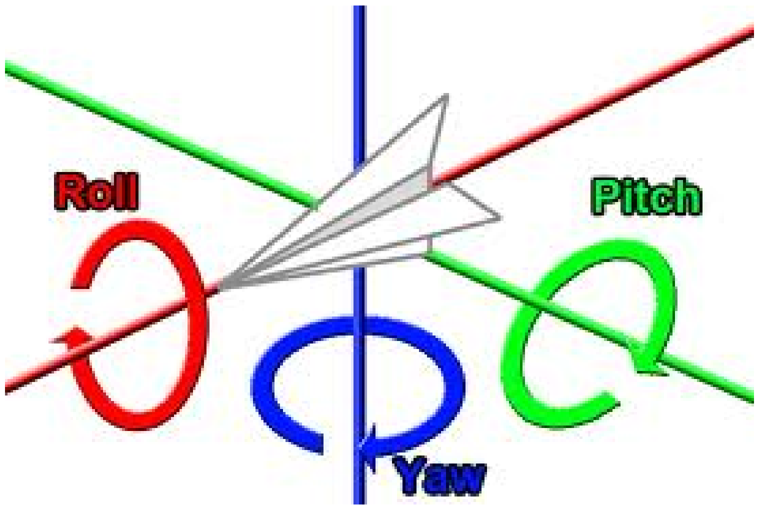
Susan Fox

February 21, 2014

1 Overview of the Nao

The Nao is the single most expensive robot we have. Treat it very gently. The Nao has the most motors and sensors of any robot in the lab. Each motor that controls a joint is paired with a sensor to detect the exact angle of that joint. In addition, the Nao has four distance sensors on its chest (and eyes?), a camera in its forehead, and multiple microphones placed strategically to allow it to localize sound.

The different directions of movement of the robot parts are defined as pitch, roll, and yaw. These directions define movement in three dimensions, based on a defining a "longitudinal" axis that goes through the object in question, and two other axes perpendicular to it. See the figure below for how pitch, roll, and yaw are defined in terms of an airplane. For the robot's head, the longitudinal axis goes through the head from back to front. For a robot limb, it goes through the length of the limb segment that is being moved.

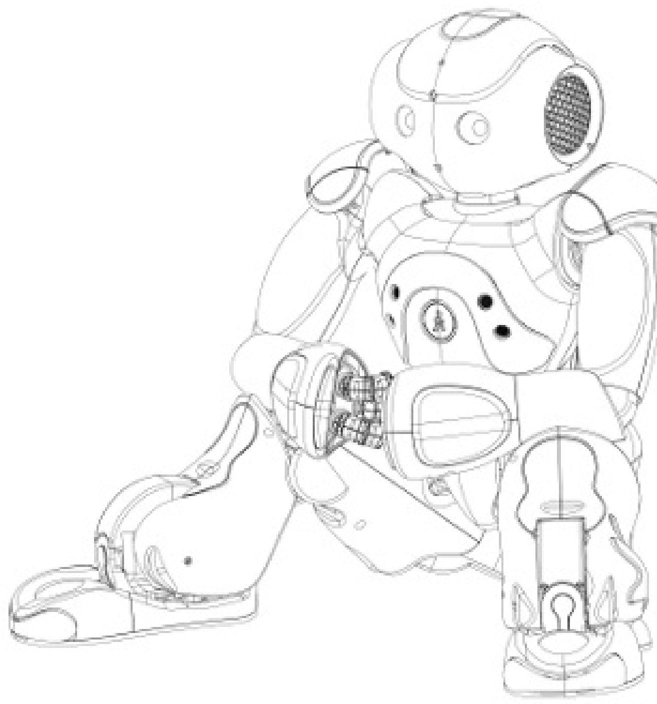


The robot's head has two controllable degrees of freedom, pitch and yaw. Pitch refers to how far up or down the head is turned, and yaw is how far to the left or right it is turned. Each robot arm has four degrees of freedom: two at the elbow (pitch and yaw) and two at the shoulder (pitch and roll). Some fancier versions of the Nao have controllable wrists and fingers as well but ours does not. Each leg on the Nao has six degrees of freedom: two at the ankle (pitch and roll), one at the knee (pitch), and three at the hip (pitch, roll, and yaw).

That's a lot of robot parts to control. Fortunately, you can create pre-defined motions quite easily, and write control programs at a high level of abstraction. There are libraries for many languages to control the Nao (Python, C++, Java, Matlab, among others), but we're going to focus for now on a graphical tool, Choregraphe. A handful of lab machines in 256 will have Choregraphe installed on them; I'll tell you which ones. The Dell laptop in the robot lab also has a version installed, along with the other libraries for various programming languages. I am hoping to refurbish the big machine in the robot lab and update it with the current versions of the Nao's software, as well.

2 Setting Up the Nao for Use

Gently carry the Nao, and its power cord, to where you are going to work with it. When off all its joints are loose, just beware! Set the robot up in a safe and stabel seated position, like this one:



You can plug in its power if you need to, but it also functions for quite a while on battery alone.

To turn Marvin on, press the button on its chest. It will take a minute or so to start up, but when it say something (a nonsense work) it will be up. Its eyes and other LEDs are lit up to show you that it is functioning. Check to see if it is on the wireless network properly by pressing its chest button. It should tell you its name, Marvin, its IP address, and its battery level.

To shut Marvin down, press and hold the chest button until the robot says something that sounds like "Nook-nook", and then all its LEDs will shut off.

3 Using Pre-Defined Behaviors

Start up Choregraphe. You will see a window with buttons acrss the top, a central workspace, a listing of bheaviors along the left, and a picture of a Nao robot on the lower right. You can add addtional tools using the View menu, but you probably don't need any of them. The first step is to connect to Marvin.

Click the green wireless button just above the central workspace, or select "Connect to..." from the Connection menu. Type in Marvin's IP address, and Choregraphe should connect automatically. The picture of the robot in the lower-right corner should change to have Marvin's colors (orange highlights), and to show Marvin's position. This picture updates based on Marvin's proprioceptive sensors.

The central workspace using a visual programming style, somewhat like Labview. You can either use pre-defined behaviors or construct your own, and hook them together in the workspace. Each behavior is defined within a "box." The listing on the left shows the pre-defined boxes/behaviors you have to choose from.

Try a very simple example. Look under Audio/Voice in the Box listing and select the `Say` box. Drag it into the central workspace. Double-click on it. This opens it up and shows you the sub-boxes that are used to define it. Notice that the bar at the top of the workspace shows you that you are now inside the `Say` behavior, which is in the `root` workspace.

In the `Say` behavior there are two sub-boxes. The first box sets the text for the robot to say, and the second box actually says the text. Notice how the boxes are hooked together by lines and hooked into the boxes at the edges of the workspace. Change the text in the box to something else. Then click on the `root` box at the top to return to the top level.

To run this behavior, we must connect it to the overall start input, and stop output: the small arrow boxes in the top left and top right corners of the workspace. Click in the start input and drag to the input on the `Say` box. Do the same for the output of the `Say` box and the overall stop output. Note that you can highlight these lines and delete them using Command-delete on the Mac.

To run the behavior, click on the green arrow button above the workspace.

3.1 Tasks to try

When running these behaviors, be sure to "spot" the robot so that it doesn't fall over. It works best on smooth surfaces, like a table-top (a big table) or a smooth floor. Carpet is not the Nao's friend.

1. Using the pre-defined behaviors, create a program that causes the robot to stand up, to say something, and then to wipe its foreheard. Make the LEDs twinkle or change color, and then have the robot sit down again.

2. Using the pre-defined behaviors, create a program that causes the robot to stand up, and then to move forward a short distance (one to three feet), turn around in place, and move back to its starting point (more or less), and sit down. Feel free to elaborate with having the robot speak, play music, flash its LEDs, whatever else makes you happy.

3. (Optional) Create a program that causes the robot to respond to touches of its head by changing the color of its eyes. Creating loops in the workspace is a bit tricky, so it's okay to just have one change and then the program quits. Can you change to a different color depending on which head sensor was pressed?

4 Creating Your Own Behaviors

To start with, realize that many of the pre-defined behaviors are written in Python down underneath. Drag a copy of the `Move To` behavior into the workspace, and double-click on it. There is Python code. Don't feel you need to understand all of it; just be aware that it is possible to write new behaviors in Python (with some effort).

We are going to use different tools to create our own motions for the robot.

First, we create a new box in the workspace. If you look under Templates You will see a Timeline box. Drag this into the workspace. The templates are empty behaviors that you can then define. You can also right-click in the workspace and make a new box, but then you have to select Timeline for the type of box.

Double-click on the Timeline box to open it. Now, at the top of the screen, you see two lines labeled Motion and Behavior layers. For now we'll just worry about the Motion line.

To create a new motion for the robot, we place body positions in the timeline at specific points, called keyframes. Then Marvin will interpolate between two adjacent keyframes, gradually changing its position from one keyframe to the next. If you put the keyframes too close together, the movements can be rapid and dangers: space them out. There are three ways of specifying keyframes: manually, using animation mode, and using recording mode. This activity will teach you the basics of two of them.

Terminology: we refer to a motor on the robot as "stiffened" when it is under power and holding the position of the robot actively. The multi-gear button on the upper right lets you stiffen or unstiffen all the motors on the robot at once. You can also stiffen just the motors you are working with, as you'll see.

4.1 Manual keyframes

Always start off by placing a keyframe at the start with the starting position of the robot in it. You can do this by clicking on the timeline at the start, right-clicking and selecting "Store joints in keyframe", and then storing the whole body.

To add additional positions manually, click on the timeline where you want the next position to go. Then, you can select motors to remember, and change the robot's position, by clicking on the relevant robot part in the picture of the robot in the lower left. This brings up a view of that part of the robot, with slider for each motor. You can move the robot limb or head to a new position using these sliders.

Each slider will tell you what motor it controls. The blue arrow on the slider indicates the robot's current position. Next to each slider is a small star-shaped circle, a record button. This indicates whether this motor is being controlled at the current keyframe. If you click one of these, then a new keyframe will automatically show up in the timeline.

At the bottom of the Motion window, there is a "Stiffen" button, which automatically turn on all the motors associated with this body part. The Mirror checkbox will automatically mirror every movement to the other paired limb.

Click on one of the arms, and stiffen all its motors. Then record the at least the two shoulder motors, and use them to raise the arm up. Afterwards, click at a new place in the timeline and move the robot back to its resting position.

To test a series of motions, click on the run button that is just beside the timeline (this runs just the motion, without requiring a whole behavior.

4.2 Animation mode

Animation mode is an alternative to creating keyframes using the "Robot view" tools. When you put the robot into animation mode, you stiffen all the robot's motors. Then, using the robot's touch sensors, you can request the head, arms, or legs to unstiffen. You then move the body part to a new location, and then stiffen the body part again. When you have all the robot's parts stiffened into the new position, you *say* "Store this position" and the program makes a new keyframe.

To stiffen/unstiffen Marvin's head, press the center touch sensor on Marvin's head.

To stiffen/unstiffen Marvin's leg, press the toe bumper on the leg in question.

According to the documentation, pressing the front and back head touch sensors will stiffen/unstiffen Marvin's arms, but I cannot seem to get that to work. As an alternative, you can use the "Robot View" and click on the arm, then use its stiffen button to toggle the arm's stiffness. This works best with at least two people.

To run the motion behavior as a full program, go back to root level, and connect the timeline to the start input and stop output. Also, note that you can right-click on the timeline box and edit it to change its name and tooltips and such.

4.3 Tasks to Try

1. Using manual mode or animation mode, create a program where the robot turns its head to the left and raises its left arm as if pointing at something. Then, have it raise its right arm, turn its head to the right, and move its left arm toward the right as well, as if it is tracking the object moving from left to right.

2. Pair up the timeline you created with the robot speaking something. To time other behaviors (like LED flashing, or speaking) with the motion you've planned, you can work with the Behavior layers to put behaviors into the timeline. See me for help.