Author: Kevin Kowalski (Caltech '12, B.S. Computer Science)
Neural Signal Processing Laboratory (http://www.nsplab.org), UCLA
Last revision: 1 June 2012

Using the Kinect

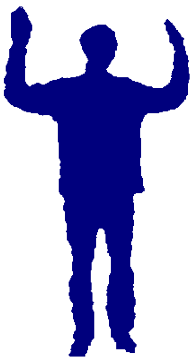
Step-by-step instructions for setting up the Kinect to interface with Matlab are as follows. We've tested these instructions with 32-bit and 64-bit Windows 7 (running 32-bit and 64-bit versions of Matlab, respectively). Before you begin, you will need a recent (tested on 2011a) version of Matlab. If you are running a 64-bit version of Matlab on a 64-bit Windows computer, then you will also need to have some kind of C++ compiler installed: Matlab has a built-in 32-bit compiler that should suffice if you have a 32-bit installation, and gcc should work if you are running Linux or MacOS (untested). Free Matlab-compatible C++ compilers include Microsoft Visual C++ 2010 Express (available here) and Microsoft Windows SDK 7.1 (available here).

1. Install the latest unstable OpenNI Binaries from here. When choosing between 32-bit and 64-bit versions, make sure that this matches the architecture of your Matlab installation.
2. Install the sensor drivers from here.
3. Install the latest unstable NITE (OpenNI Compliant Middleware Binaries) from here.
4. Connect the Kinect to the PC.
5. Download Matlab wrapper functions for OpenNI from here, and run `compile_cpp_files` in Matlab. You will need to call `compile_cpp_files` with the location of your OpenNI installation. For example, from the Matlab prompt, you might type `compile_cpp_files ('C:\Program Files\OpenNI\')` if the openNI installation is in `C:\Program Files\OpenNI\`.
6. (Optional) Place `runKinectTrialSet.m` in the same folder as the examples and `compile_cpp_files.m`, and run it.

For your convenience, we have collected all the above files for a 32-bit Matlab installation on a Windows computer in `tutorial_files.zip`.

When using the Kinect to capture arm movements in Matlab, it is important to note that the refresh rate of the Kinect is 30 Hz, so it is only possible to update the intended velocity of the subject every 33 ms. Additionally, if one wants to take full advantage of the Kinect's refresh rate, any filtering code that runs in each time step should finish in < 33 ms, which may be an issue on a standard desktop computer that predates 2012.

An example for controlling a cursor with the Kinect can be found in `runKinectTrialSet.m`. This file should be placed in the same folder as the examples for the Matlab wrapper functions. Before the Kinect can track hand movements, the device must be calibrated by holding the "phi pose" (shown below) until the Kinect recognizes it. Stand about 5 ft. away from the camera, with the camera at chest level.



The phi calibration pose for the Kinect. Image courtesy of http://www.greenfoot.org/doc/kinect/calibration.html

The Spike Simulation System

We simulate motor cortical neuron spiking activity as a Bernoulli process, governed by a velocity-driven cosine-shaped tuning curve (Moran & Schwartz, J. Neurophys., 1999, and Truccolo et. al, J. Neurophys., 2005). The Bernoulli method of spike generation is implemented in `generateNeuralSignals.m`.

Assume that the spiking activity of each motor neuron is characterized as a Poisson process. Each neuron i has an associated vector $\theta_i = \begin{bmatrix} a_i & b_i & c_i \end{bmatrix}^T$ such that given an intended cursor velocity $v = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$, the Poisson intensity λ is given by

$$\lambda^i(v) = \exp(a_i v_x + b_i v_y + c_i).$$

If this maximum spiking rate is set to be one per time step, then the occurrence of a spike can be well-approximated as a Bernoulli random variable, given by:

$$\Pr(n_k^i = 1 \mid v) = \lambda(v)\Delta t_k.$$

In contradistinction to the Time Rescaling Theorem (Brown et al., Neural Computation, 2001) which requires multiple iterations per time step, this method is more practical for achieving real-time simulation on a standard desktop computer (e.g. Intel Core i7 2600K 3.4GHz Quad Core processor with 16 GB RAM).