1. Make sure Python 2.7 is installed on your computer
2. Install Homebrew (for OS X)
   1. copy and paste the following into a terminal: ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)”
   2. brew install wget
   3. cd /usr/local
   4. find Cellar
   5. ls -1 bin
3. In terminal to download opencv and get started running tests
   1. brew tap homebrew/science
   2. brew info opencv
   3. brew install opencv
   4. cat ~/.bash\_profile | grep PYTHONPATH
   5. cd /Library/Python/2.7/site-packages
   6. ln -s /usr/local/Cellar/opencv/2.4.9/lib/python2.7/site-packages/cv.py cv.py
   7. ln -s /usr/local/Cellar/opencv/2.4.9/lib/python2.7/site-packages/cv2.so cv2.so
   8. python
      1. import cv
      2. import cv2
   9. “Download as a zip” <https://github.com/themailman05/FaceDetectionTutorial.git>
   10. Unzip the files, and read the readme.md for further instructions on how to run the programs
   11. Try out different haar cascade patterns- the default pattern works great for faces front-on, the alternative pattern sometimes has a higher success rate.

This lab introduces OpenCV ([http://opencv.org](http://opencv.org/)), and focuses on real time applications and computer vision. Using OpenCV’s focus on computer vision, we can see the applications simply through facial recognition software. The ability for the computer to get a test case of what a face is, and it’s ability to recognize other faces based on that shows it’s machine learning ability.

This can be taken to another level by importing a database of faces as well. After importing a database of faces, the computer will have access to those faces and can treat those ones differently than the faces that it doesn’t recognize (or isn’t in the test suite).

The tests suites that are provided aren’t perfect, as one can see through experiments and looking at examples provided. For example, it can only recognize based on the front of the face, not the profile. Also, the face has to be clear in order for it to be detected, and sometimes things that are not faces are detected as faces as well. This is a part of the test suite that was provided to the program, and giving it an alternative sample could change the output of what are faces according to the program. This can be seen even in comparing ‘haarcascade\_frontalface\_default.xml’ to ‘haarcascade\_frontalface\_alt.xml’

This introduction to OpenCV with simple facial recognition software through pictures and webcam has endless possibilities of applications, from creating weapons to sending a picture to a computer and seeing if the person that is in the picture is recognized based on a database of people that the computer already has. Facebook already has a type of facial recognition (that surpasses this example). When someone uploads a picture to Facebook and goes to tag people in it, Facebook puts a square around the face and the name of who it thinks it’s a picture of (based on the users previous posts with the person and how their face is structured compared with other pictures on facebook). CNN goes into the facial recognition on Facebook in the following article—the number of times it is correct compared to people tagging photos manually may be surprising—<http://money.cnn.com/2014/04/04/technology/innovation/facebook-facial-recognition/>