TRAINING DETECTORS AND RECOGNIZERS IN PYTHON AND OPENCV

Sept. 9, 2014

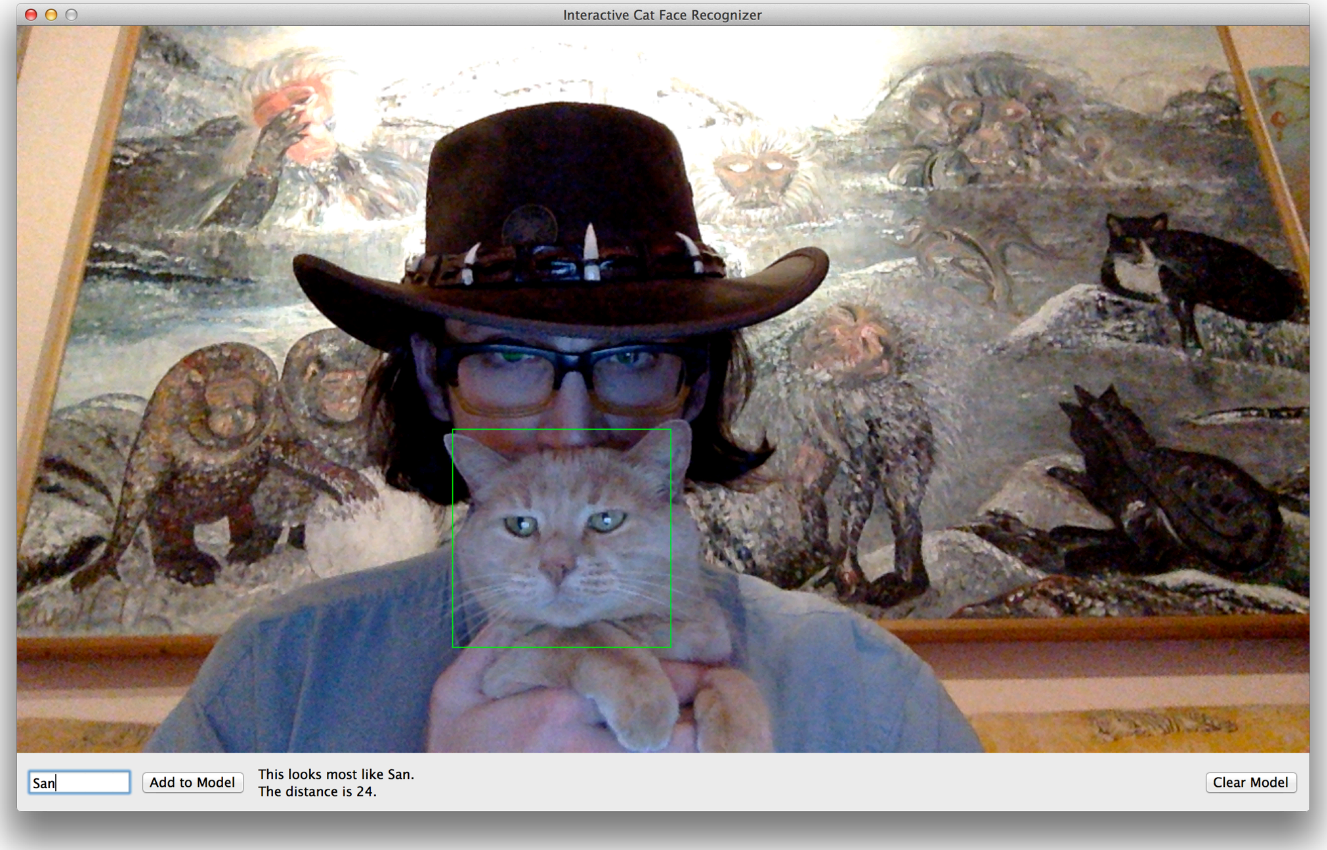
ISMAR 2014

Joseph Howse

GOALS

Build apps that learn from photos & from real-time camera input.

Detect & recognize the faces of humans & cats.



This looks most like San.
The distance is 24.

GETTING STARTED

TERMINOLOGY

- This tutorial covers detection and recognition…
  - not to be confused with tracking.
- Detect
  - Find the location of some type of object in an image.
  - “I detect that this region of the image is a human face.”
- Recognize
  - Determine the subtype or the unique identity of a detected object.
  - “I recognize that this human face is a male face.”
  - “I recognize that this human face is Joe Howse’s face.”
- Track
  - Determine whether the same detected object is present in consecutive images and, if so, how it moved.
  - “I tracked this face in images 1 and 2; it moved from here to there.”

WHY OPENCV?

- Mid-level API
    - Developer chooses algorithms and types of I/O
    - Library provides (semi-)optimized implementations
- Multi-lingual
    - C++, C, Python, Java
- Cross-platform
    - Windows, Mac, Linux, BSD, iOS, Android
    - Well supported on ARM Linux
        - I have used it on Raspberry Pi (Raspbian) and Odroid U3 (Lubuntu).
- (Semi-)optimized
    - TBB (x86, amd64, ARM), CUDA, OpenCL, Tegra 3+
    - Some functions are optimized; others are not.

SETUP

PROJECT FILES

- Angora Blue
  - My set of demo applications for face detection and recognition
  - https://bitbucket.org/Joe_Howse/angora-blue
  - https://bitbucket.org/Joe_Howse/angora-blue/downloads
- Three databases of images
  1. VOC2007 dataset
     - 10,000 annotated images of diverse subjects
  2. Caltech Faces 1999
     - 450 images of upright, frontal human faces
  3. Microsoft Cat Dataset 2008
     - 10,000 annotated images of cat faces in various orientations
  - A download script for the databases is in Angora Blue:
    - cascade_training/download_datasets.sh

DEPENDENCIES

- Python 2.7
  - A multi-paradigm scripting language
- NumPy 1.8
  - A math library for fast array operations with Pythonic syntax
- OpenCV 2.4
  - A computer vision library with lots of algorithms and I/O features
  - OpenCV Python treats images as NumPy arrays.
- WxPython 2.8, 2.9, or 3.0
  - A GUI library, wrapping native GUI libraries on each platform

WINDOWS

- Download and run the following binary installers (either 32-bit or 64-bit, depending on your needs):
  - Python 2.7
    - https://www.python.org/download
  - NumPy 1.8
    - http://sourceforge.net/projects/numpy/files/NumPy/1.8.1
  - OpenCV 2.4
    - http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.9
    - …or build from source for options such as Kinect support
  - WxPython 3.0
    - http://www.wxpython.org/download.php

MAC

- Use the MacPorts package manager.
  - http://www.macports.org/install.php
- Optionally, configure MacPorts to use my OpenCV repository, which adds Kinect support.
  - http://nummist.com/opencv/ports.tar.gz
- $ sudo port install python27 python_select py27-numpy
- $ sudo port select python python27
- $ sudo port install opencv +python27 +tbb
  - …or other variants, such as the following:
    - $ sudo port install opencv +python27 +tbb +openni
    - $ sudo port install opencv +python27 +tbb +openni_sensorkinect
- $ sudo port install wxPython-3.0

UBUNTU, DEBIAN, RASPBIAN, ETC.

- $ sudo apt-get install python-opencv
  - …or build from source for options such as Kinect support. Example:
    - http://nummist.com/opencv/install_opencv_debian_wheezy.sh
- $ sudo apt-get install python-wxgtk2.8

WORKING WITH IMAGES, CAMERAS, AND GUIS

BASIC IMAGE I/O IN OPENCV

- Read image from camera:
  - cameraID = 0 # Default camera
  - capture = cv2.VideoCapture(cameraID)
  - didSucceed, image = capture.read()
    - The captured image is always in BGR (not RGB or gray) format.
    - For optimized RGB or gray capture, use low-level camera libraries instead.
      - I like this Python wrapper for Video for Linux 2 (v4l2):
        - https://github.com/gebart/python-v4l2capture/blob/master/capture_picture.py
- Read image from file:
  - image = cv2.imread('input.png')
- Write image to file:
  - cv2.imwrite('output.png’, image)

OPENCV IMAGES IN WX GUIS

- Run OpenCV stuff on background thread; wx on main thread
    - threading.Thread class and wx.CallAfter function
    - Demo: InteractiveRecognizer.py
        - __init__, _runCaptureLook, and _onCloseWindow methods
- Convert images from numpy.array to wx.StaticBitmap
    - wx.BitmapFromBuffer function
        - …but this function is buggy on Raspberry Pi
            - Fall back to wx.ImageFromBuffer and wx.BitmapFromImage functions
    - Demo: utils.py
        - wxBitmapFromCvImage function

DETECTING FACES

AVAILABLE DETECTION MODELS

- OpenCV supports several types of detectors, including these two:
  1. Haar cascade – relatively reliable
     - Detects light-to-dark edges, corners, and lines at multiple scales
     - http://docs.opencv.org/trunk/doc/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html#basics
  2. Local binary pattern (LBP) – relatively fast
     - Detects light-to-dark gradients at multiple scales
     - http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms
  - Both types use data stored in XML files.
  - Neither type can detect rotated or flipped objects.

USING A PRE-TRAINED DETECTOR

- OpenCV comes with XML files for many pre-trained detectors
  - Human face – frontal, profile, eyes, eyeglasses, nose, mouth
  - Human body – upper, lower, whole
  - Other – silverware, Russian license plates
- Basic usage:
  - detector = cv2.CascadeClassifier('haarcascade_eye.xml')
  - detectedObjects = detector.detectMultiScale(image)
  - for x, y, width, height in detectedObjects: # Do something for each object
- detectMultiScale has important optional arguments:
  - http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html#cascadeclassifier-detectmultiscale
- Demo:
  - InteractiveRecognizer.py: __init__ and _detectAndRecognize methods
  - InteractiveHumanFaceRecognizer.py

TRAINING A CUSTOM DETECTOR

- OpenCV provides a pair of command line tools to generate XML files for Haar or LBP detectors
  1. opencv_createsamples or opencv_createsamples.exe
  2. opencv_traincascade or opencv_traincascade.exe
  - http://docs.opencv.org/doc/user_guide/ug_traincascade.html
  - Among other parameters, they require text files listing negative and positive training images (e.g. non-faces and faces).
  - Demo:
    - cascade_training/train.sh or cascade_training/train.bat
    - The resulting XML file is used in InteractiveCatFaceRecognizer.py.

NEGATIVE TRAINING IMAGES

- Gather 1000s of images that do not contain faces.
- List the image paths in a text file.
  - Demo: cascade_training/negative_description.txt

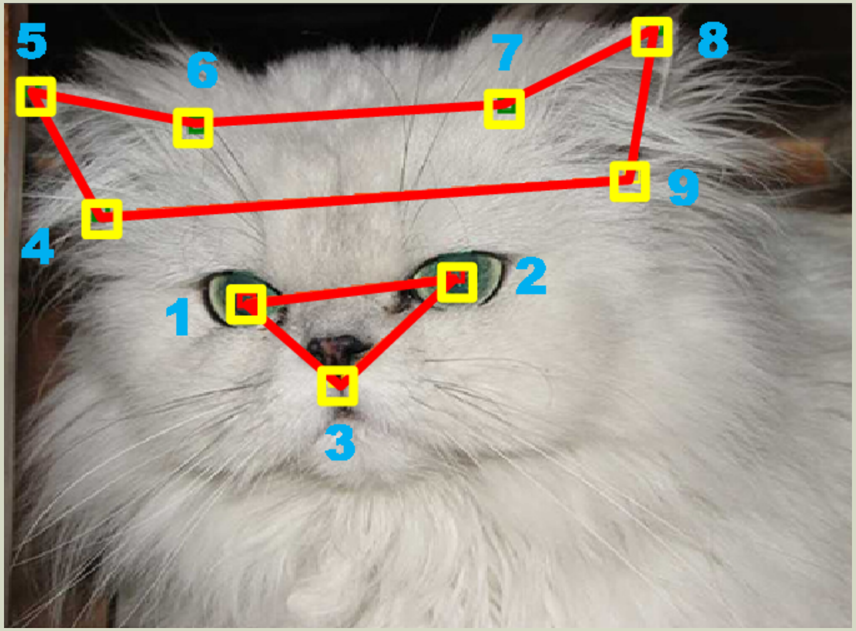
- Preprocessing:
  1. Convert to grayscale
     - cv2.cvtColor
  2. Equalize (adjust contrast)
     - cv2.equalizeHist
  - Demo: cascade_training/describe.py
    - describeNegativeHelper function

POSITIVE TRAINING IMAGES

- Gather 1000s of images containing faces.
- List the image paths and face coordinates in a text file.
  - Demo: cascade_training/ positive_description.txt

- Preprocessing:
  1. Convert to grayscale
     - cv2.cvtColor
  2. Straighten
     - cv2.getRotationMatrix2D
     - cv2.warpAffine
  3. Crop
     - numpy.array slicing
       - crop = image[y:y+h, x:x+w]
  4. Equalize (adjust contrast)
     - cv2.equalizeHist
  - Demo: cascade_training/ describe.py
    - preprocessCatFace function

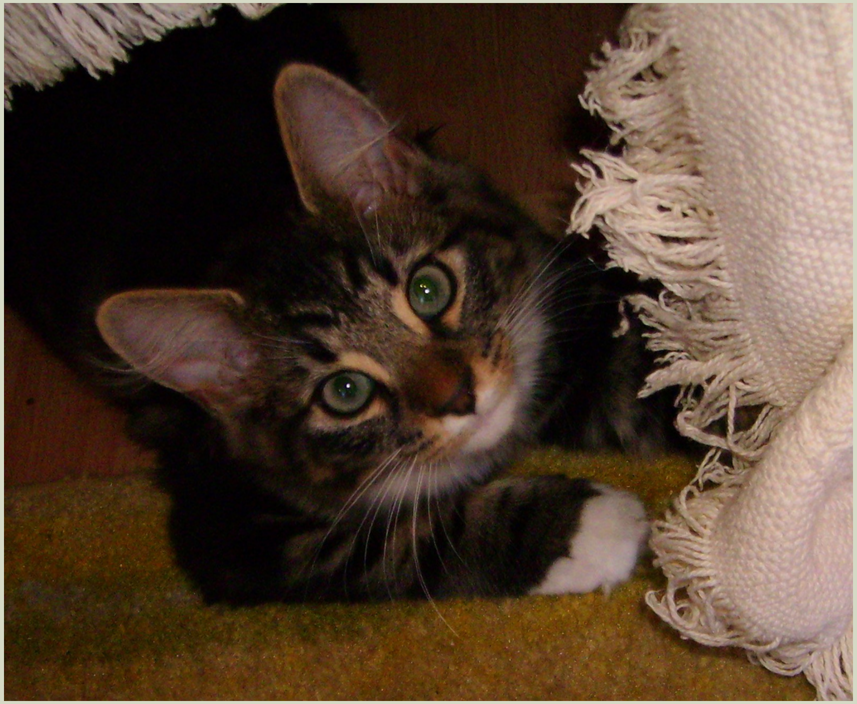
PREPROCESSING: REFERENCE POINTS



- To straighten & crop, we need reference points.
  - A person places them manually for each image!
- The Cat Dataset defines 8 reference points.
  - We use points 4 & 9 to compute face size…
  - and 1 & 2 to compute face rotation and center.

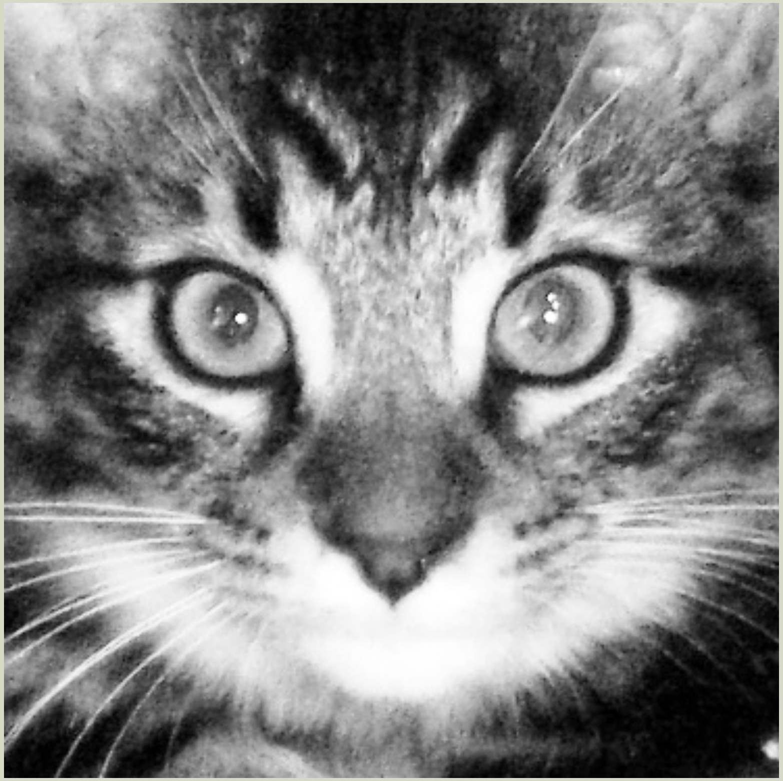
PREPROCESSING: CONVERT TO GRAY, STRAIGHTEN, CROP, EQUALIZE

Before



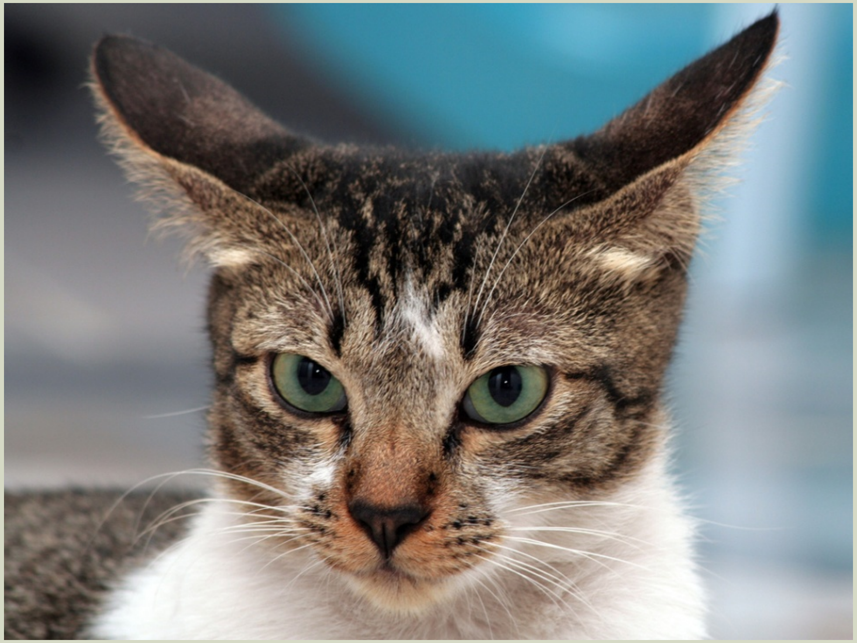
"Can I haz atan2?"

After



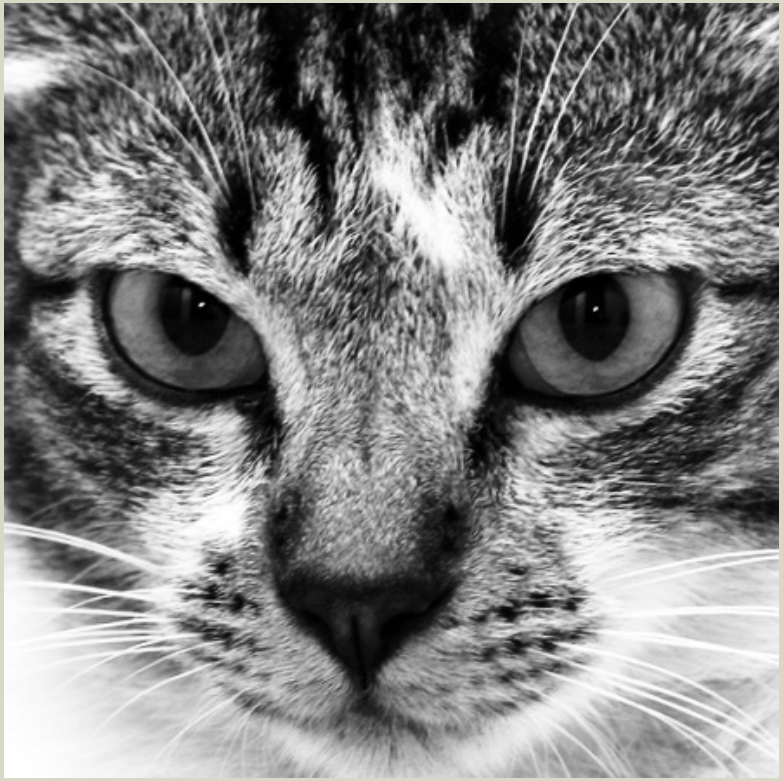
PREPROCESSING: CONVERT TO GRAY, STRAIGHTEN, CROP, EQUALIZE

Before



"My ears mock your rectangle."

After



RECOGNIZING FACES

AVAILABLE RECOGNITION MODELS

- OpenCV supports three types of recognizers:
  1. Eigenfaces – relatively reliable
     - Recognizes differences from the "average" face
     - http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec_tutorial.html#eigenfaces
  2. Fisherfaces – also relatively reliable
     - Also recognizes differences from the "average" face
     - http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec_tutorial.html#fisherfaces
  3. Local binary pattern histograms (LBPH) – relatively fast
     - Detects light-to-dark gradients at multiple scales
     - Can learn new faces one-by-one in real time
     - http://docs.opencv.org/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms
  - All types use data stored in XML files.
  - None of the types can detect rotated or flipped objects.

TRAINING AND USING A RECOGNIZER

1. Create a recognizer:
   - recognizer = cv2.createLBPHFaceRecognizer() # or Fisher or Eigen
2. Train a recognizer:
   - trainingImages = [joe0, joe1, sam0, sam1]
   - trainingLabels = numpy.array([0, 0, 1, 1])
   - recognizer.train(trainingImages, trainingLabels)
3. Get a recognition result:
   - testLabel, distance = recognizer.predict(testImage)
4. Update an LBPH recognizer with more training images:
   - # Only LBPH supports updates.
   - recognizer.update(moreTrainingImages, moreTrainingLabels)
5. Save and re-load a recognizer
   - recognizer.save('PeopleIKnow.xml')
   - recognizer.load('PeopleIKnow.xml')

- Demo: InteractiveRecognizer.py
  - __init__, _detectAndRecognize, _updateModel, _onCloseWindow methods

FURTHER READING

MY BOOKS

- Howse, J. *OpenCV Computer Vision with Python*. Packt Publishing, 2013.
  - A brief introduction to OpenCV with Python
  - Includes integration with NumPy, SciPy, OpenNI, & SensorKinect
- Howse, J. *Android Application Programming with OpenCV*. Packt Publishing, 2013.
  - A brief introduction to OpenCV with Android
- Howse, J. *OpenCV for Secret Agents*. Packt Publishing, forthcoming.
  - Intermediate to advanced OpenCV projects using Python, Raspberry Pi, Android, & other gadgets
  - Analyze images of real estate, cats, gestures, cars, heartbeats, & more.
  - Preorder & early access: http://cdn1.cf.packtpub.com/opencv-for-secret-agents/book

OTHER BOOKS

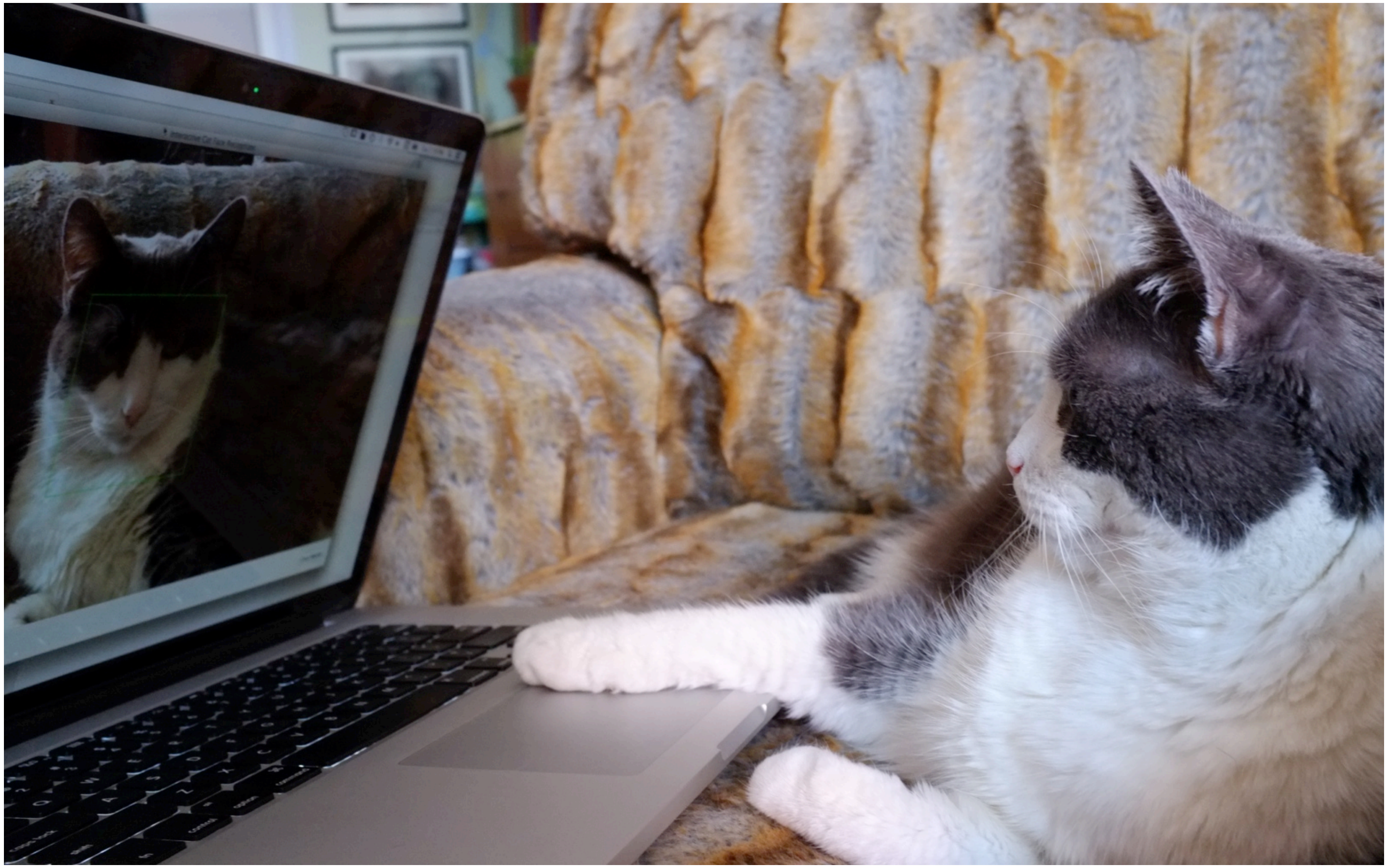
- Baggio, D. L. et al. *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing, 2012.
  - Intermediate to advanced OpenCV projects using C++
  - Includes advanced chapters on human face detection, tracking, and recognition
- Laganière, R. *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, 2011.
  - Concise code samples in C++ for many popular algorithms

PAPER

- Zhang, W., Sun, J., and Tang, X. Cat Head Detection - How to Effectively Exploit Shape and Texture Features, *Proc. of European Conf. Computer Vision*, vol. 4, pp. 802-816, 2008.
  - http://research.microsoft.com/pubs/80582/eccv_cat_proc.pdf

WEBSITES

- Python 2.7 docs
  - https://docs.python.org/2
- NumPy docs
  - http://docs.scipy.org/doc/numpy/reference
- OpenCV docs
  - http://docs.opencv.org
- WxPython docs
  - http://wiki.wxpython.org
- Support site for my books
  - http://nummist.com/opencv
- Abid Rahman K.’s OpenCV Python blog
  - http://opencvpython.blogspot.com
- KittyDar: A cat detector in JavaScript
  - http://harthur.github.io/kittydar



DISCUSSION

Let us reflect on what we have learned.