SimpleCV Documentation

Release 1.2

Ingeuitas

April 09, 2015

Contents

Contents:

CHAPTER 1

SimpleCV Cookbook

1.1 Loading and Saving Images

The central class in SimpleCV is the Image() class, which wrappers OpenCV's iplImage (bitmap) and cvMat (matrix) classes and provides basic manipulation functions.

To load an image, specify the file path in the constructor:

```
my_image = Image("path/to/image.jpg")
```

To save the image, use the save method. It will automatically use the file you loaded the image from:

```
my_image.save()
```

You can also specify a new file to save to, similar to a "Save As...", and future calls to save() will save to this new file:

```
my_image.save("path/to/new_image.jpg")
#...do some stuff...
my_image.save() #saves to path/to/new_image.jpg
```

1.2 Image Manipulation

You can scale images using the "scale" function, so for instance to create a thumbnail. Note that this will distort perspective if you change the width and height ratios:

```
thumbnail = my_image.scale(90, 90)
```

You can also look at individual pixels:

```
r, g, b = my_image[25, 45]  #get the color trio for pixel at x = 25, y = 45
```

If you use python slices, you can extract a portion of the image. This section is returned as an Image object:

```
my_section = myimage[25:50, 45:70]    #grab a 25x25 rectangle starting at x = 25, y = 45
my_section.save("path/to/new_cropped_image.jpg")
```

You can also assign using direct pixel addressing, and draw on the image using this method:

```
black = 0.0, 0.0, 0.0    #create a black color tuple
my_image[25,45] = black  #set the pixel at x = 25, y = 45 to black
my_image[25:50, 45] = black #draw 1px wide line
my_image[25:50, 45:70] = black #create a 25x25 black rectange starting at x = 25, y = 45
```

1.3 Using a Camera, Kinect, or VirtualCamera

Addressing your OpenCV supported webcam is extremely easy:

```
mycam = Camera()
img = mycam.getImage()
```

If you install the OpenKinect library and python wrapper, you can use your Xbox Kinect to get a depth map:

```
k = Kinect()
img = k.getImage() #normal, full color webcam
depth = k.getDepth() #greyscale depth map
depthdata = k.getDepthMatrix() #raw depth map, 0-2048
```

1.4 Multiple Cameras

And you can even use multiple cameras, at different resolutions:

```
mylaptopcam = Camera(0, {"width": 640, "height": 480})  #you can also control brightness, hue, gain,
myusbcam = Camera(1, {"width": 1280, "height": 720})

mylaptopcam.getImage().save("okaypicture.jpg")
myusbcam.getImage().save("nicepicture.jpg")
```

You can also initialize VirtualCameras from static data files:

```
imgcam = VirtualCamera("apples.jpg", "image")
vidcam = VirtualCamera("bananas.mpg", "video")

imgcam.getImage().save("copy_of_apples.jpg")
imgcam.getImage().save("frame_1_of_bananas.jpg")
```

You can also use a JpegStreamCamera to grab frames from an MJPG source (such as an IP Cam, the "IP Webcam" android application, or another SimpleCV JpegStream:

```
jc = JpegStreamCamera("http://myname:mypasswd@ipcamera_host/stream.mjpg")
jc.getImage().save("seeyou.jpg")
```

1.5 Colors and Levels

You can also split channels, if you are interested in only processing a single color:

```
(red, green, blue) = Camera().getImage().channels()
red.save("redcam.jpg")
green.save("greencam.jpg")
blue.save("bluecam.jpg")
```

The Image class has a builtin Histogram function, thanks to Numpy. Histograms can show you the distribution of brightness or color in an image:

```
hist = Camera().getImage().histogram(20)
brightpixels = 0
darkpixels = 0
i = 0
```

```
while i < length(hist):
  if (i < 10):
    darkpixels = darkpixels + hist[i]
  else:
    brightpixels = brightpixels + hist[i]
  i = i + 1

if (brightpixels > darkpixels):
  print "your room is bright"
else:
  print "your room is dark"
```

1.6 Features and FeatureSets

SimpleCV has advanced feature-detection functions, which can let you find different types of features. These are returned in FeatureSets which can be addressed as a group, or filtered:

```
img = Camera.getImage()

lines = img.findLines()

corners = img.findCorners()

lines.draw(Color.RED) #outline the line segments in red
corners.draw(Color.BLUE) #outline corners detected in blue

left_side_corners = corners.filter(corners.x() < img.width / 2)
#only look at corners on the left half of the image

longest_line = lines.sortLength()[0]
#get the longest line returned
```

1.7 Blob Detection

You can use SimpleCV to find connected components (blobs) of similarly-colored pixels:

```
#find the green ball
green_stuff = Camera().getImage().colorDistance(Color.GREEN)

green_blobs = green_channel.findBlobs()
#blobs are returned in order of area, smallest first

print "largest green blob at " + str(green_blobs[-1].x) + ", " + str( green_blobs[-1].y)
```

1.8 Barcode Reading

If you load the python-zxing library, you can use Zebra Crossing to detect 2D and 1D barcodes in a number of various formats. Note that you will need to specify the location of the library either through the ZXING_LIBRARY %ENV variable, or as a parameter to findBarcode():

```
i = Camera().getImage()
barcode = i.findBarcode("/var/opt/zxing")

barcode.draw(Color.GREEN) #draw the outline of the barcode in green

i.save("barcode_found.png")
print barcode.data
```

1.9 Haar Face Detection

You can do Haar Cascade face detection with SimpleCV, but you will need to find your own Haar Cascade File:

```
i = Camera().getImage()
faces = i.findHaarFeatures("/path/to/haarcascade_frontalface_alt.xml")

#print locations
for f in faces:
  print "I found a face at " + str(f.coordinates())

#outline who was drinking last night (or at least has the greenest pallor)
faces.sortColorDistance(Color.GREEN)[0].draw(Color.GREEN)
i.save("greenest_face_detected.png")
```

1.10 Output Streams

SimpleCV uses PyGame as an interface to the Simple Directmedia Layer (SDL). This makes it easy to create interfaces using SimpleCV's Display module:

```
from SimpleCV.Display import Display

c = Camera()
d = Display()
while not d.isDone():
    c.getImage().save(d)
```

SimpleCV has an integrated HTTP-based JPEG streamer. It will use the old-school multipart/replace content type to continuously feed jpgs to your browser. To send the data, you just save the image to the js.framebuffer location:

```
import time
c = Camera()
js = JpegStreamer()  #starts up an http server (defaults to port 8080)

while(1)
  c.getImage().save(js)
  time.sleep(0.1)
```

You can also write frames directly to video, using OpenCV's VideoWriter. Note that your available formats may be dependent on your platform:

```
import time
c = Camera
vs = VideoStream("out.avi", fps=15)

framecount = 0
```

```
while(framecount < 15 * 600): #record for 5 minutes @ 15fps
    c.getImage().save(vs)
    time.sleep(0.1)
```

CHAPTER 2

Installation

*Note: There are also video tutorials located online at: <http://simplecv.org>

or on the youtube channel: <http://www.youtube.com/user/IngenuitasOfficial>

2.1 Ubuntu 10.4 or 10.11

You can now download a .deb file from SourceForge – look at http://sourceforge.net/projects/simplecv/files for an easier install.

2.2 Ubuntu 11.4

You can upgrade your OpenCV from 2.1 to 2.2 with the following command:

```
sudo add-apt-repository https://launchpad.net/~gijzelaar/+archive/opencv2 && sudo apt-get update
```

Then install the debian package

2.3 Ubuntu 11.10

To upgrade from OpenCV 2.1 to OpenCV 2.3 run the following command:

> sudo add-apt-repository ppa:gijzelaar/cuda && sudo add-apt-repository ppa:gijzelaar/opencv2.3 && sudo apt-get update

Then install the debian package

Here is the manual method for installation:

Steps:

1. apt-get install dependencies
2. download, build and install the latest version of OpenCV
3. clone and install SimpleCV

Commands:

```
sudo apt-get install -y --force-yes build-essential swig gfortran cmake gcc pkg-config libjpeg62-dev
wget http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.2/OpenCV-2.2.0.tar.bz2
bunzip2 OpenCV-2.2.0.tar.bz2
tar xf OpenCV-2.2.0.tar
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D BUILD_PYTHON_SUPPORT=ON ..
make
sudo make install
sudo cp /usr/local/lib/python2.6/site-packages/cv.so /usr/local/lib/python2.6/dist-packages/cv.so
sudo apt-get install -y --force-yes git git python-dev python-numpy python-nose python-scipy ipython
git clone git://git.code.sf.net/p/simplecv/git.git simplecv
sudo python setup.py install
```

2.4 Mac OS X (10.6 and above)

Note: While not required, it is strongly recommended that you install XCode from Apple: http://itunes.apple.com/us/app/xcode/

If you want to keep control of your usr/local or are adept at building for Unix, you may want to use the directions below. Otherwise, we recommend using our Superpack, which contains everything you need in a single package: http://sourceforge.net/projects/simplecv/files/

Steps:

1. Install Xcode http://developer.apple.com/technologies/xcode.html
2. Install homebrew https://github.com/mxcl/homebrew/wiki/installation
3. Use homebrew to install opencv and git
4. Install scipy superpack, but with ipython 0.10.2 http://stronginference.com/scipy-superpack/ (download from http://ingenuitas.com/misc/superpack_10.6_2011.05.28-ipython10.sh)
5. Install python imaging library (10.6 needs ARCHFLAGS tweak)
6. clone simplecv and python setup.py install

Commands:

```
ruby -e "$(curl -fsSLk https://gist.github.com/raw/323731/install_homebrew.rb)"
wget http://ingenuitas.com/misc/superpack_10.6_2011.05.28-ipython10.sh
sh superpack_10.6_2011.05.28-ipython10.sh
brew install opencv
ln -s /usr/local/lib/python2.6/site-packages/cv.so /Library/Python/2.6/site-packages/cv.so
brew install git
ARCHFLAGS="-arch i386 -arch x86_64" brew install PIL
git clone git://git.code.sf.net/p/simplecv/git.git simplecv
cd simplecv
python setup.py install
```

2.5 Windows 7/Vista

If you want a streamlined install which gives you all the dependencies, we recommend using the Windows Superpack, available at http://sourceforge.net/projects/simplecv/files/

If you already have Python, OpenCV or SciPy installed and want to keep things the way you like them, follow the directions below

Steps:

1. (OPTIONAL) Install MinGW for optional files and building openCV from source. Make sure to include C/C++ Compiler and msys package. http://sourceforge.net/projects/mingw/files/Automated%20MinGW%20Installer/
2. Install Python 2.7 http://www.python.org/getit/releases/2.7.1/
3. Install Python Setup Tools for Windows http://pypi.python.org/packages/2.7/s/setuptools/ (See: http://stackoverflow.com/questions/309412/how-to-setup-setuptools-for-python-2-6-on-windows)
4. Install the SciPy superpack: http://sourceforge.net/projects/scipy/files/scipy/0.9.0rc5/scipy-0.9.0rc5-win32-superpack-python2.7.exe/download
5. Install OpenCV: http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.2/ (See: http://luugiathuy.com/2011/02/setup-opencv-for-python/)
6. easy_install.exe simplecv (See: http://blog.sadphaeton.com/2009/01/20/python-development-windows-part-2-installing-easyinstallcould-be-easier.html)

CHAPTER 3

SimpleCV Package

3.1 SimpleCV Package

3.2 Camera Module

3.3 Color Module

3.4 ColorModel Module

3.5 Display Module

3.6 DrawingLayer Module

3.7 Font Module

3.8 ImageClass Module

3.9 Images Module

3.10 Stream Module

3.11 base Module

3.12 Subpackages

3.12.1 Features Package

Features Package

BOFFeatureExtractor Module

Blob Module

BlobMaker Module

Detection Module

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*