OpenCV 3.3.0 Starter Kit

by David Scherfgen
Version 1.0 (2017-09-18)

What is OpenCV and (why) should I use it?

OpenCV is a very popular Open Source library for Computer Vision tasks.

Why you should use it for your project:

- You can concentrate on the actual task and, not on the implementation of basic algorithms.
- OpenCV is mature and fast (uses CPU-specific optimizations and can work with GPUs).
- It is available for many platforms and programming languages.
- It contains many basic and advanced image processing / Computer Vision algorithms as well as functions to load/save images and videos and create simple GUIs.
- If you prefer another Computer Vision / image processing framework (e. g. MATLAB's Image Processing Toolbox), you're welcome to use that. Just don't do everything from scratch – time is too short for that!

About the Starter Kit

The OpenCV Starter Kit contains a compiled and ready-to-use version of OpenCV 3.3.0 for Windows and an example program in C++ and in Python. Sorry, Linux and Mac guys, you'll have to get it working by yourself.

OpenCV has been compiled using Microsoft Visual Studio 2017, targeting the Win32 platform (32-bit, not 64-bit). This way, it can be used on both 32-bit and 64-bit machines. If you want to program in C++, installing Microsoft Visual Studio 2017 is recommended. I cannot guarantee that earlier versions will work. The 2017 Community edition is available for free at https://www.visualstudio.com/thank-you-downloading-visual-studio/?sku=Community&rel=15 and is completely fine for our purposes.

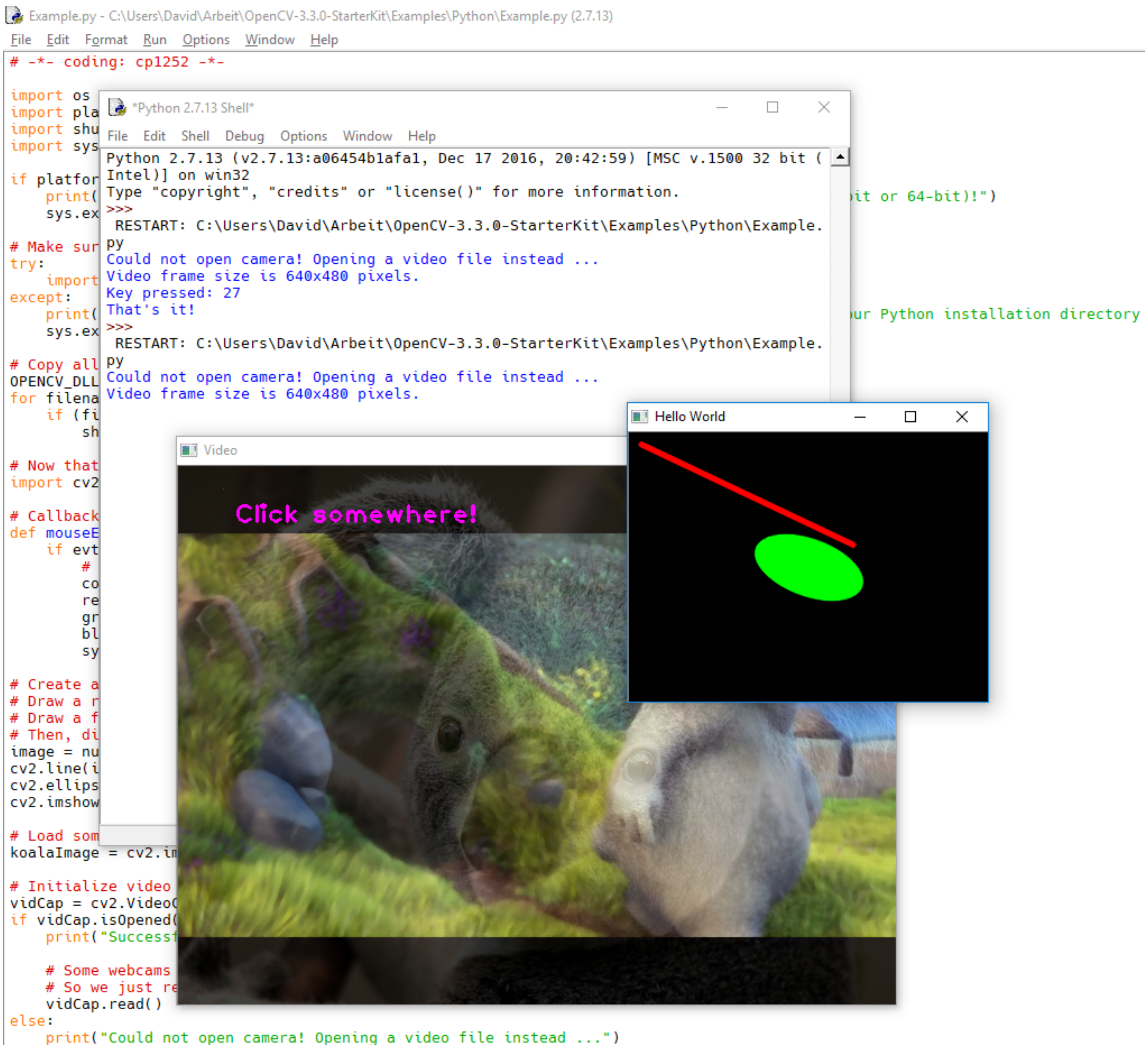
Also, two Python modules are included: One for Python 2.7 and one for Python 3.6. These modules also target the Win32 platform, so make sure to use a 32-bit Python version.

Getting Started with the Starter Kit

Extract the Zip file to a convenient location.

If you want to work with C++, open the "Example.sln" solution in the "Examples\C++" folder with Microsoft Visual Studio 2017. Build (compile) and run the project. Everything should work out of the box. When building the project, the OpenCV DLLs are automatically copied to the "Debug" and "Release" output directories so that the program can find and load them. This is done by telling Visual Studio to run the batch file "CopyDLLs.bat" when building the project (this option can be found in the project properties).

If you want to work with Python, you'll first have to install the NumPy library. Fortunately, this is quite easy using Python's package manager "pip". Python 3 should already have it. In Python 2.7, I had to install it myself. Open a command prompt with administrator rights, navigate to your Python installation path (e. g. “CD C:\Python27”) and then enter “python -m ensurepip --upgrade”. Once pip is installed, use it to install NumPy. Again, in a command prompt with administrator rights, navigate to your Python installation path, then enter “Scripts\\pip install numpy”. The NumPy package should now be downloaded from the internet and installed. Afterwards, you should be able to run the “Example.py” script in the “Examples\Python” folder. This script also copies OpenCV’s Python modules and its DLL files into the current directory to make sure that they can be loaded.


Example.py - C:\Users\David\Arbeit\OpenCV-3.3.0-StarterKit\Examples\Python\Example.py (2.7.13)
File Edit Format Run Options Window Help
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Video
Hello World
Click somewhere!

```
# -*- coding: cp1252 -*-

import os
import pla
import shu
import sys

if platfor
    print(                                        it or 64-bit)!")
    sys.ex

# Make sur
try:
    import
except:
    print(                                        ur Python installation directory
    sys.ex

# Copy all
OPENCV_DLL
for filena
    if (fi
        sh

# Now that
import cv2

# Callback
def mouseE
    if evt
        #
        co
        re
        gr
        bl
        sy

# Create a
# Draw a r
# Draw a f
# Then, di
image = nu
cv2.line(i
cv2.ellips
cv2.imshow

# Load som
koalaImage = cv2.im

# Initialize video
vidCap = cv2.VideoC
if vidCap.isOpened(
    print("Successf

    # Some webcams
    # So we just re
    vidCap.read()
else:
    print("Could not open camera! Opening a video file instead ...")
```

```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (
Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\David\Arbeit\OpenCV-3.3.0-StarterKit\Examples\Python\Example.
py
Could not open camera! Opening a video file instead ...
Video frame size is 640x480 pixels.
Key pressed: 27
That's it!
>>>
 RESTART: C:\Users\David\Arbeit\OpenCV-3.3.0-StarterKit\Examples\Python\Example.
py
Could not open camera! Opening a video file instead ...
Video frame size is 640x480 pixels.
```

Creating a New Project

I recommend to take the C++/Python example as a starting point for your own project. The Visual Studio project is already configured to link to all OpenCV libraries and has the correct paths set.

OpenCV Documentation and Tutorials / Further Help

Get documentation/tutorials from here: http://docs.opencv.org/3.3.0/

In case of problems, always use your brain first, then your debugger, then Google and then maybe http://answers.opencv.org/questions/. If none of that helped, you write me an e-mail to david.scherfgen@h-brs.de.