Enhanced visualization: assignment 1

pyOpenGL and OpenCV

In the first part of this assignment you will be asked to install the computer vision library OpenCV in Python and the computer graphics library OpenGL. Installation in Python in this case means that we are going to install the Python's bindings for these libraries[1].

[1] A binding is a wrapper, that is a container, around a C code so that we can easily use it in another language.

In the following, the given installation instructions may change from one computer to another. Look at INTERNET forums if you have problems.

It is assumed that you already have Anaconda with Python installed in your machine. If that is not the case, install Anaconda. You can download Anaconda and read installation instructions at the following website:

`https://www.anaconda.com/download/`

If you do not have the numerical algebra library *numpy* installed, install it by opening a command window and using the command line: pip install numpy (or conda install numpy if the first is not available).

Install and test OpenCV

Windows: the easiest way is to install using a wheel which can be downloaded from

`http://www.lfd.uci.edu/~gohlke/pythonlibs/`

for example the file

opencv_python−3.1.0−cp35−cp35m−win32.whl

stands for OpenCV 3.1.0 version for Python 3.5 32 bit installation. Save the file in a separate folder.

After downloading it, open a command window and change directory to where the file has been saved. Then enter the command line

```
pip install "opencv_python−3.1.0−cp35−cp35m−win32.whl"
```

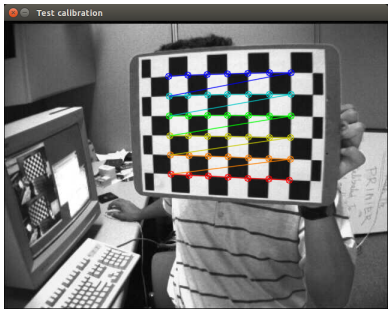

Figure 1: Window displaying the result of the OpenCV test code in Python

Linux: the easiest way in Linux is to open a terminal and enter the command line

```
conda install −c https://conda.binstar.org/menpo opencv
```

Mac: you can install it directly with a command line in a terminal:

```
conda install --channel https://conda.anaconda.org/menpo opencv3
```

You can test if the installation succeeded by downloading from *Jalon* and running the first code cell of the Jupyter notebook "assignment_1.ipynb". Do not forget to also download the file "left1 .jpg". The test should should display the image in Figure 1.

Install and test pyOpenGL

Windows: similarly to OpenCV, the easiest way is to install using a wheel which can be downloaded from

`http://www.lfd.uci.edu/~gohlke/pythonlibs/#pyopengl`

You need to install pyOpenGL and pyOpenGL-accelerate. Choose the right versions for your Python installation and follow the same instructions as for OpenCV.

Linux: normally you can install directly with pip install command. If you have never written code using the OpenGL library, you may need to install the OpenGL libraries *GL*, *GLU* and *GLUT*. Thus in a terminal enter the following command lines (the first command will require you to type your super user password):

```
sudo apt-get install freeglut3-dev
pip install pyopengl PyOpenGL_accelerate
```

You can also test your installation by running the second code cell of the Jupyter notebook. The test should should display the image in Figure 2.

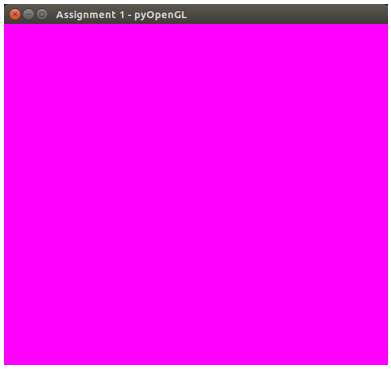

Figure 2: Window displaying the result of the pyOpenGL test code in Python

OpenCV code

Now you are going to analyze what the first code cell in "assignment_1.ipynb" does.

1. Explain what commands #1 to #4 do in this code.
2. Plot 3D points from the coordinates given by the columns of array objp. What do these points represent?
3. Can you give a suggestion on why obtaining the colored points on the image is important? If yes, what is your suggestion?

pyOpenGL code

Now you are going to focus on the second cell of the code. The "main" function works similarly as in C language. Note that "main" calls other functions defined above it.

1. How do you change the following parameters : the background color to cyan? The window size to 400×400 pixels? The name of the window?[2]
2. After changing the name of the window to "Test 1", the color of the background to cyan and the size of the window to 400×400 pixels, uncomment the lines in the middle of the draw function. If you run the cell, what do you see?

[2] Some Windows users may need to cast the window name in binary, just put a "b" before the window name as commented in the code.