实训一：OpenCV 环境的配置及使用初步

作者：杨仕龙

欢迎各位走进数字图像的世界。
在本课程中，你将学习到数字图像的基本原理和处理的基本方法，并将亲自动手实验，感受数字图像处理在真实场景中的应用。
本实验是本课程的第一次实验，它是你走向丰富多彩的图像世界的重要一步，请务必重视。
在课程的学习中，如果你遇到了问题，请和我取得联系。我的 E-MAIL 是 leland@lelandyang.com。
Never leave the classroom with a problem unsolved. Wish you a good journey in the DIP world!

1 实验目的与实验要求

1.1 实验目的

1. 复习 Python 环境的安装配置，学习安装 OpenCV 环境，为后续课程学习做准备
2. 练习 opencv-python 的一些基本函数的使用方法，完成实验任务，感受其方便之处，为后续实验奠定基础；
3. 复习在理论课程中学习到的数字图像相关基础知识，学以致用。

1.2 实验要求

按照实验任务与指导一小节的要求完成实验任务：将实验结果截图，按照给定的模板认真填写实验报告，在实验课结束之前将实验报告提交至教师机，并提交至超星平台（可以在结束之后完善你的结果之后再提交）。

2 实验任务与指导

2.1 OpenCV 及 opencv-python 简介

OpenCV 是一个开源的计算机视觉库。所谓 CV，指的是 Computer Vision，即计算机视觉。它封装了很多常用的数字图像处理和计算机视觉任务的算法，具有开箱即用的良好特征。

该库使用 C++ 编写而成，但是却不仅仅是在 C++ 中使用，它提供了各种语言的接口封装（Wrapper）。在本课程中，我们使用的是 Python 封装。具体而言，opencv-python 是一个 Python 模块，你需要安装它之后方可使用。在使用之前，你需要通过 import 导入该模块。

在实际应用中，OpenCV 常常与 Numpy、Matplotlib 以及 Pillow 等第三方模块搭配使用。特别要说明的是，OpenCV 在工业界使用广泛，因此，学习使用 OpenCV 具有非常大的实际价值。

2.2 实验环境的安装与配置

Python 的安装在大二上学期已经学习，因此不再赘述。实验室的 Python 环境存在缺陷，且由于 OpenCV 版本之间往往不兼容。因此，本课程提供 Python 3.7.3 的安装包，建议你卸载之前的 Python 环境再安装本版本。你可以通过 python —version 查看当前安装的 Python 版本。

另外，由于网络环境以及编译工具链的关系，本课程提供了已经编译的 opencv-python 模块 wheel 文件，该模块直接依赖 numpy，因此也一并提供。使用步骤如下：

1. 将提供的压缩包解压至一个文件夹，假设该文件夹的全路径为：`E:\test`；
2. Win + R 快捷键打开运行窗口，输入“cmd”打开命令窗口，在出现的命令窗口中输入以下命令：
   (a) `cd /d E:\test\`
   (b) `pip install opencv_python-4.5.5.64-cp37-none-any.whl --find-links E:\test`
3. 如果一切正常，opencv-python 开发环境已经安装好了。在命令行中运行 Python，输入 import cv2，如果不报错，表明模块已经正常安装，可供使用。
4. 按照此方法也安装好 Matplotlib 等模块（在压缩包中也已经提供。）

2.3 实验任务

2.3.1 图像读取并显示

请将下列代码输入到 IDLE，保存并运行，将运行结果截图（Alt + Print Srcreen 截图之后粘贴到 Word 文档），保存到实验报告，并填写代码分析。

```
1  import cv2 as cv
2  import matplotlib.pyplot as plt
3
4  imfile = 'lena_std.tif'
5  im = cv.imread(imfile)
6  plt.subplot(1, 2, 1)
7  plt.imshow(im)           # 请注意图像颜色，分析为什么。
8  im = cv.cvtColor(im, cv.COLOR_BGR2RGB)
9  plt.subplot(1, 2, 2)
10 plt.imshow(im)
11 plt.show()
```

2.3.2 同时显示多幅图像

在实际图像处理应用中，我们在很多情况下需要同时显示多张图像以便对比。请将实验资源文件夹里的“7.1.09.tiff”、“boat.512.tiff”、“5.2.08.tiff”以及“4.2.06.tiff”这四张图像按照两行两列的格式显示在同一个显示界面上。

提示：使用 subplot() 函数实现，该函数的函数原型为：`subplot(nrows, ncols, index, **kwargs)`。完成之后请将显示的图像保存下来，粘贴至实验报告，并将你的 Python 代码粘贴至实验报告相应位置。完成之后的效果如图1所示。

2.3.3 RGB 通道提取和保存

我们常见的图像一般采取 24bit 的位深，使用 RGB 色彩模型存储。本实验任务的目的在于分别提取并保存 R、G、B 三个通道，并保存为三个 BMP 文件，请将它们粘贴到实验报告内。在本练习中，你保存的其实是三张 8bit 的灰度图。

另外，尝试将 RGB 彩色文件的两个通道置零之后显示出来，例如：保留 R（红）通道，将 G 和 B 通道置零。请注意，你保存的三个文件仍然是 24bit 的位深，因此你实际上是将另外两个通道置零。假设你的图像是 im，则该图像的每一个通道的访问方法示例如下：`im[:, :, 0]` 表示第一个通道，将它置零

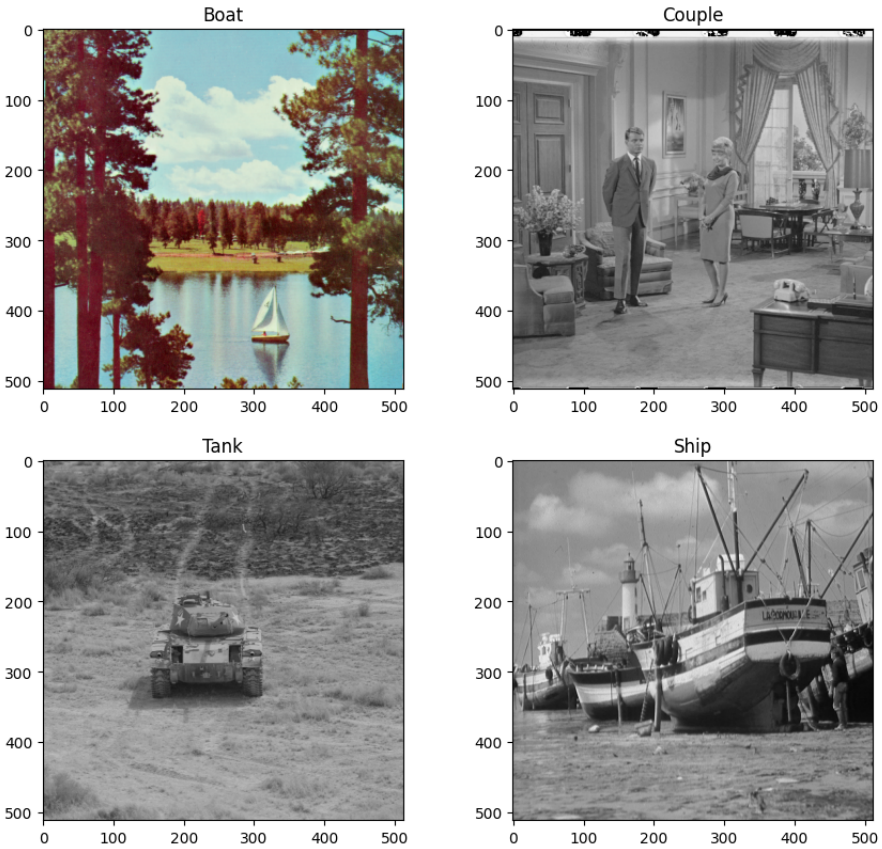

图 1: 多图显示练习

可以简单的使用：`im[:, :, 0] = 0` 即可。由于 OpenCV 中，RGB 彩图采用的是 B、G、R 的顺序存储，`im[:, :, 0]` 表示的是蓝通道。

2.3.4 彩图转换为灰度图

在本实验的资源文件夹中存放有 kodim15.png，请使用 OpenCV 读取它并且使用两种方法（OpenCV 的函数以及下面的转换公式）将其转换为灰度图。

提示：参考转换公式如 (1) 所示。建议尝试最小值法（取 R、G、B 三者的最小值）、最大值法（取 R、G、B 三者的最大值）。

$$Gray = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{1}$$

请注意：OpenCV 中，彩色图像使用 B、G、R 的顺序表示三个通道。使用 Matplotlib 显示图像。将显示的窗口截图（或者点击软盘按钮保存成 PNG 文件之后插入到 WORD 文档），将你的代码一并粘贴至你的实验报告。你的实验结果应当类似于图2所示。

2.3.5 生成随机灰度图像

本实验采用 Python 的 random 模块随机生成 $p \in [0, 255]$ 的像素点（numpy.zeros 或者 numpy.ones），填充到矩阵中构成一张大小为 512×512 的随机图像，请将此图像和你使用到的源代码粘贴到你的实验报告。生成的图像应该形如图3所示。

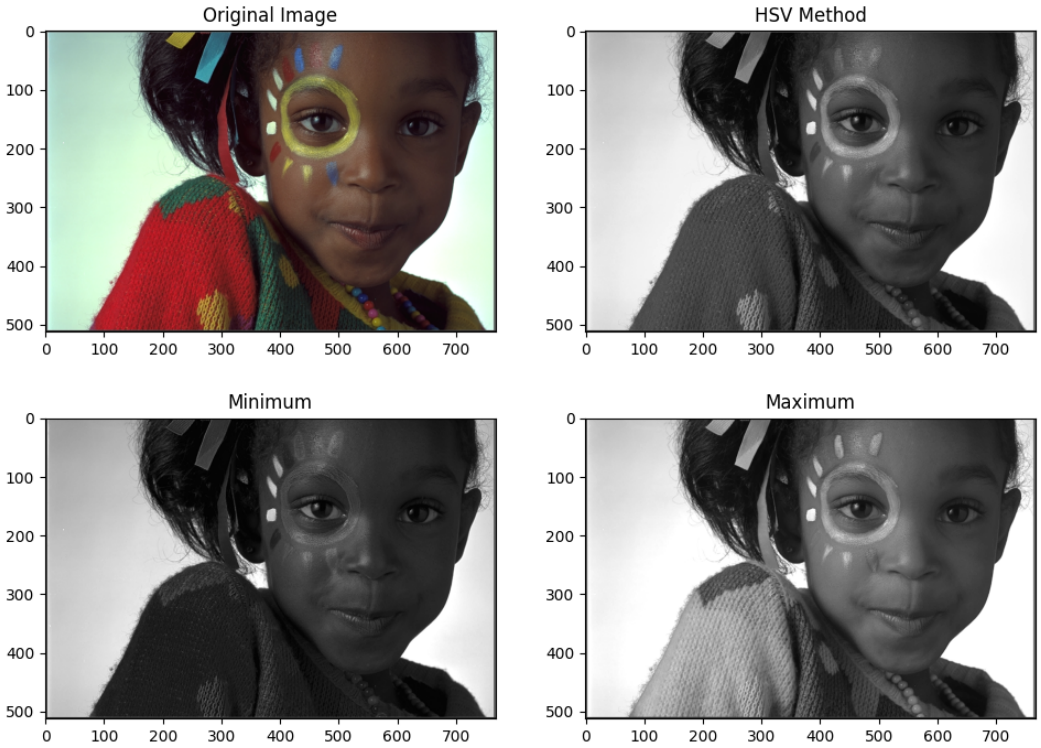

图 2: 彩图转灰阶图像

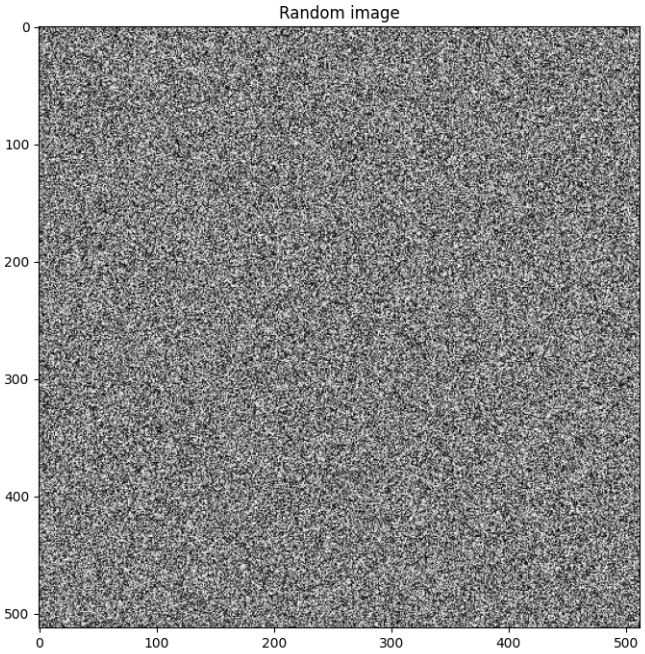

图 3: 随机生成图像