

问题一至问题十

问题一：通道交换

读取图像，然后将RGB通道替换成BGR通道。

下面的代码用于提取图像的红色通道。注意，`cv2.imread()` 的系数是按BGR顺序排列的！其中的变量 `red` 表示的是仅有原图像红通道的 `imori.jpg`。

```
1 import cv2
2 img = cv2.imread("imori.jpg")
3 red = img[:, :, 2].copy()
```

输入 (imori.jpg)	输出(answers_image/answer_1.jpg)
	

答案：


Python >> [answers/answer_1.py](#)

C++ >> [answers_cpp/answer_1.py](#)

问题二：灰度化 (Grayscale)

将图像灰度化吧！灰度是一种图像亮度的表示方法，通过下式计算：

$$Y = 0.2126R + 0.7152G + 0.0722B$$

输入 (imori.jpg)	输出(answers_image/answer_2.jpg)
	

答案：


Python >> [answers/answer_2.py](#)

C++ >> [answers_cpp/answer_2.py](#)

问题三：二值化 (Thresholding)

把图像进行二值化吧！二值化是将图像使用黑和白两种值表示的方法。这里我们将灰度的阈值设置为128来进行二值化，即：

$$y = \begin{cases} 0 & (\text{if } y < 128) \\ 255 & (\text{else}) \end{cases}$$

输入 (imori.jpg)	输出(answers_image/answer_3.jpg)
	

答案

Python >> [answers/answer_3.py](#)

C++ >> [answers_cpp/answer_3.py](#)

问题四：大津二值化算法 (Otsu's Method)

使用大津算法来二值化图像吧！大津算法，也被称作最大类间方差法，是一种可以自动确定二值化中阈值的算法，从类内方差和类间方差的比值计算得来：

- 小于阈值 t 的类记作0，大于阈值 t 的类记作1；
- w_0 和 w_1 是被阈值 t 分开的两个类中的像素数占总像素数的比率（满足 $w_0 + w_1 = 1$ ）；
- S_0^2 , S_1^2 是这两个类中像素值的方差；
- M_0 , M_1 是这两个类的像素值的平均值；

即：

- 类内方差： $S_w^2 = w_0 \cdot S_0^2 + w_1 \cdot S_1^2$
- 类间方差： $S_b^2 = w_0 \cdot (M_0 - M_t)^2 + w_1 \cdot (M_1 - M_t)^2 = w_0 \cdot w_1 \cdot (M_0 - M_1)^2$
- 图像所有像素的方差： $S_t^2 = S_w^2 + S_b^2 = \text{常数}$



根据以上的式子，我们用以下的式子计算分离度 X ：

$$X = \frac{S_b^2}{S_w^2} = \frac{S_b^2}{S_t^2 - S_b^2}$$

也就是说：

$$\arg \max_t X = \arg \max_t S_b^2$$

换言之，如果使 $S_b^2 = w_0 \cdot w_1 \cdot (M_0 - M_1)^2$ 最大，就可以得到最好的二值化阈值 t 。

输入 (imori.jpg)	输出 (th = 127) (answers_image/answer_4.jpg)
	

答案

Python >> [answers/answer_4.py](#)

C++ >> [answers_cpp/answer_4.py](#)

问题五：HSV变换

将使用HSV表示色彩的图像的色相反转吧！

HSV即使用**色相 (Hue)**、**饱和度 (Saturation)**、**明度 (Value)** 来表示色彩的一种方式。

- 色相：将颜色使用 0° 到 360° 表示，就是平常所说的颜色名称，如红色、蓝色。色相与数值按下表对应：

红	黄	绿	青色	蓝色	品红	红
0°	60°	120°	180°	240°	300°	360°

- 饱和度：是指色彩的纯度，饱和度越低则颜色越黯淡 ($0 \leq S < 1$) ；
- 明度：即颜色的明暗程度。数值越高越接近白色，数值越低越接近黑色 ($0 \leq V < 1$) ；

从RGB色彩表示转换到HSV色彩表示通过以下方式计算：

RGB的取值范围为 $[0, 1]$ ，令：

$$\text{Max} = \max(R, G, B)$$

$$\text{Min} = \min(R, G, B)$$

色相：

$$H = \begin{cases} 0 & (\text{if Min} = \text{Max}) \\ 60 \cdot \frac{G-R}{\text{Max}-\text{Min}} + 60 & (\text{if Min} = B) \\ 60 \cdot \frac{B-G}{\text{Max}-\text{Min}} + 180 & (\text{if Min} = R) \\ 60 \cdot \frac{R-B}{\text{Max}-\text{Min}} + 300 & (\text{if Min} = G) \end{cases}$$

饱和度：

$$S = \text{Max} - \text{Min}$$

明度：

$$V = \text{Max}$$

从HSV色彩表示转换到RGB色彩表示通过以下方式计算：

$$\begin{aligned} C &= S \\ H' &= \frac{H}{60} \\ X &= C \cdot (1 - |H' \bmod 2 - 1|) \\ (R, G, B) &= (V - C) \cdot (1, 1, 1) + \begin{cases} (0, 0, 0) & (\text{if H is undefined}) \\ (C, X, 0) & (\text{if } 0 \leq H' < 1) \\ (X, C, 0) & (\text{if } 1 \leq H' < 2) \\ (0, C, X) & (\text{if } 2 \leq H' < 3) \\ (0, X, C) & (\text{if } 3 \leq H' < 4) \\ (X, 0, C) & (\text{if } 4 \leq H' < 5) \\ (C, 0, X) & (\text{if } 5 \leq H' < 6) \end{cases} \end{aligned}$$

请将色相反转（色相值加180），然后再用RGB色彩空间表示图片。

输入 (imori.jpg)	输出(answers_image/answer_5.jpg)
	

答案

Python >> [answers/answer_5.py](#)

C++ >> [answers_cpp/answer_5.py](#)



问题六：减色处理

这里没有找到"减色处理"准确的中文翻译，所以直译了。

——gZR

这里我们将图像的值由 256^3 压缩至 4^3 ，即将RGB的值只取 $\{32, 96, 160, 224\}$ 。这被称作色彩量化。色彩的值按照下面的方式定义：

$$\text{var} = \begin{cases} 32 & (0 \leq \text{var} < 64) \\ 96 & (64 \leq \text{var} < 128) \\ 160 & (128 \leq \text{var} < 192) \\ 224 & (192 \leq \text{var} < 256) \end{cases}$$

输入 (imori.jpg)	输出(answers_image/answer_6.jpg)
	

答案

Python >> [answers/answer_6.py](#)


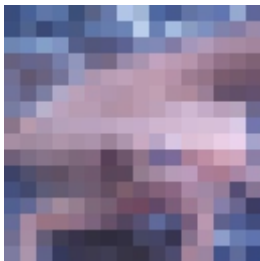
C++ >> [answers_cpp/answer_6.py](#)

问题七：平均池化 (Average Pooling)

将图片按照固定大小网格分割，网格内的像素值取网格内所有像素的平均值。我们将这种把图片使用均等大小网格分割，并求网格内代表值的操作称为池化（Pooling）。池化操作是卷积神经网络（Convolutional Neural Network）中重要的图像处理方式。平均池化按照下式定义：

$$v = \frac{1}{|R|} \cdot \sum_{i=1}^R v_i$$

请把大小为 128×128 的 `imori.jpg` 使用 8×8 的网格做平均池化。

输入 (imori.jpg)	输出(answers_image/answer_7.jpg)
	

答案

Python >> [answers/answer_7.py](#)

C++ >> [answers_cpp/answer_7.py](#)

问题八：最大池化 (Max Pooling)

网格内的值不取平均值，而是取网格内的最大值进行池化操作。

输入 (imori.jpg)	输出(answers_image/answer_8.jpg)
	

答案

Python >> [answers/answer_8.py](#)

C++ >> [answers_cpp/answer_8.py](#)

问题九：高斯滤波 (Gaussian Filter)

使用高斯滤波器 (3×3 大小, 标准差 $\sigma = 1.3$) 来对 `imori_noise.jpg` 进行降噪处理吧!

高斯滤波器是一种可以使图像平滑的滤波器，用于去除噪声。可用于去除噪声的滤波器还有中值滤波器（参见问题十），平滑滤波器（参见问题十一）、LoG滤波器（参见问题十九）。

高斯滤波器将中心像素周围的像素按照高斯分布加权平均进行平滑化。这样的（二维）权值通常被称为卷积核或者滤波器。

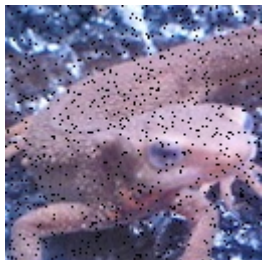

但是，由于图像的长宽可能不是滤波器大小的整数倍，因此我们需要在图像的边缘补0。这种方法称作 `Zero Padding`。并且权值 g （卷积核）要进行**归一化操作** ($\sum g = 1$)。

按下面的高斯分布公式计算权值：

$$g(x, y, \sigma) = \frac{1}{2 \cdot \pi \cdot \sigma^2} \cdot e^{-\frac{x^2 + y^2}{2 \cdot \sigma^2}}$$

标准差 $\sigma = 1.3$ 的8近邻高斯滤波器如下：

$$K = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

输入 (imori_noise.jpg)	输出(answers_image/answer_9.jpg)
	

答案

Python >> [answers/answer_9.py](#)

C++ >> [answers_cpp/answer_9.py](#)

问题十：中值滤波 (Median filter)

使用中值滤波器 (3×3 大小) 来对 `imori_noise.jpg` 进行降噪处理吧!

中值滤波器是一种可以使图像平滑的滤波器。这种滤波器用滤波器范围内 (在这里是 3×3) 像素点的中值进行滤波, 在这里也采用 `Zero Padding`。

输入 (imori_noise.jpg)	输出(answers_image/answer_10.jpg)
	

答案

Python >> [answers/answer_10.py](#)

C++ >> [answers_cpp/answer_10.py](#)