
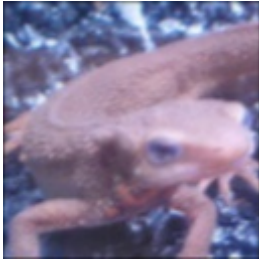


问题十一至问题二十

问题十一：均值滤波器

使用 3×3 的均值滤波器来进行滤波吧！

均值滤波器使用网格内像素的平均值。

输入 (imori.jpg)	输出(answers_image/answer_11.jpg)
	

答案


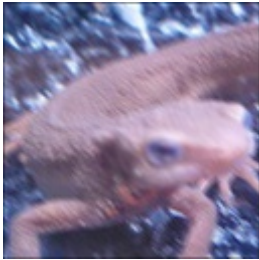
- Python >> [answers_py/answer_11.py](#)
- C++ >> [answers_cpp/answer_11.cpp](#)

问题十二：Motion Filter

使用 3×3 的Motion Filter来进行滤波吧。

Motion Filter取对角线方向的像素的平均值，像下式这样定义：

$$\begin{bmatrix} \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$$

输入 (imori.jpg)	输出(answers_image/answer_12.jpg)
	

答案

- Python >> [answers_py/answer_12.py](#)
- C++ >> [answers_cpp/answer_12.cpp](#)



问题十三：MAX-MIN滤波器

使用MAX-MIN滤波器来进行滤波吧。

MAX-MIN滤波器使用网格内像素的最大值和最小值的差值对网格内像素重新赋值。通常用于**边缘检测**。

边缘检测用于检测图像中的线。像这样提取图像中的信息的操作被称为**特征提取**。

边缘检测通常在灰度图像上进行。

输入 (imori.jpg)	输出(answers_image/answer_13.jpg)
	

答案

- Python >> [answers_py/answer_13.py](#)
- C++ >> [answers_cpp/answer_13.cpp](#)

问题十四：差分滤波器¹ (Differential Filter)

使用 3×3 的差分滤波器来进行滤波吧。

差分滤波器对图像亮度急剧变化的边缘有提取效果，可以获得邻接像素的差值。

纵向：

$$K = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

横向：

$$K = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

输入 (imori.jpg)	输出 · 纵向 (answers/answer_14_v.jpg)	输出 · 横向 (answers/answer_14_h.jpg)
		

答案

- Python >> [answers_py/answer_14.py](#)
- C++ >> [answers_cpp/answer_14.cpp](#)

问题十五：Sobel滤波器

使用 3×3 的Sobel滤波器来进行滤波吧。


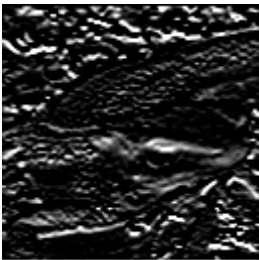
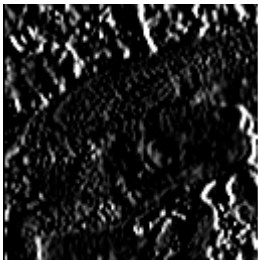
Sobel滤波器可以提取特定方向（纵向或横向）的边缘，滤波器按下式定义：

纵向：

$$K = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

横向：

$$K = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

输入 (imori.jpg)	输出 · 纵向 (answers/answer_15_v.jpg)	输出 · 横向 (answers/answer_15_h.jpg)
		

答案

- Python >> [answers_py/answer_15.py](#)
- C++ >> [answers_cpp/answer_15.cpp](#)

问题十六：Prewitt滤波器

使用 3×3 的Prewitt滤波器来进行滤波吧。


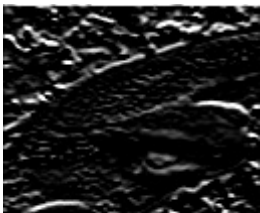
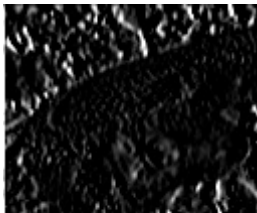
Prewitt滤波器是用于边缘检测的一种滤波器，使用下式定义：




纵向：

$$K = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

横向：

$$K = \begin{bmatrix} -1 & 0 & -1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

输入 (imori.jpg)	输出 · 纵向 (answers/answer_16_v.jpg)	输出 · 横向 (answers/answer_16_h.jpg)
		

		
输入 (imori.jpg)	(answers/answer_16_v.jpg)	(answers/answer_16_h.jpg)
答案		
<ul style="list-style-type: none"> Python >> answers_py/answer_16.py C++ >> answers_cpp/answer_16.cpp 		

问题十七：Laplacian滤波器

使用Laplacian滤波器来进行滤波吧。

Laplacian滤波器是对图像亮度进行二次微分从而检测边缘的滤波器。由于数字图像是离散的， x 方向和 y 方向的一次微分分别按照以下式子计算：

$$I_x(x, y) = \frac{I(x+1, y) - I(x, y)}{(x+1) - x} = I(x+1, y) - I(x, y)$$

$$I_y(x, y) = \frac{I(x, y+1) - I(x, y)}{(y+1) - y} = I(x, y+1) - I(x, y)$$

因此二次微分按照以下式子计算：

$$\begin{aligned}
 & I_{xx}(x, y) \\
 &= \frac{I_x(x, y) - I_x(x-1, y)}{(x+1) - x} \\
 &= I_x(x, y) - I_x(x-1, y) \\
 &= [I(x+1, y) - I(x, y)] - [I(x, y) - I(x-1, y)] \\
 &= I(x+1, y) - 2 \cdot I(x, y) + I(x-1, y)
 \end{aligned}$$

同理：


$$I_{yy}(x, y) = I(x, y+1) - 2 \cdot I(x, y) + I(x, y-1)$$

特此，Laplacian 表达式如下：

$$\begin{aligned}
 & \nabla^2 I(x, y) \\
 &= I_{xx}(x, y) + I_{yy}(x, y) \\
 &= I(x-1, y) + I(x, y-1) - 4 \cdot I(x, y) + I(x+1, y) + I(x, y+1)
 \end{aligned}$$

如果把这个式子表示为卷积核是下面这样的：

$$K = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

输入 (imori.jpg)	输出(answers/answer_17.jpg)
	

答案

- Python >> [answers_py/answer_17.py](#)

- C++ >> [answers_cpp/answer_17.cpp](#)

问题十八：Emboss滤波器

使用Emboss滤波器来进行滤波吧。

Emboss滤波器可以使物体轮廓更加清晰，按照以下式子定义：

$$K = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

输入 (imori.jpg)	输出(answers/answer_18.jpg)
	

答案

- Python >> [answers_py/answer_18.py](#)
- C++ >> [answers_cpp/answer_18.cpp](#)

问题十九：LoG滤波器

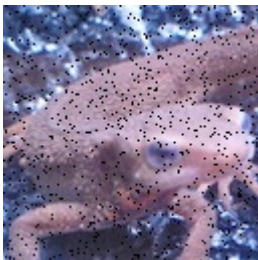

使用LoG 滤波器，来对 imori_noise.jpg 检测边缘吧！

LoG即高斯-拉普拉斯（Laplacian of Gaussian）的缩写，使用高斯滤波器使图像平滑化之后再使用拉普拉斯滤波器使图像的轮廓更加清晰。

为了防止拉普拉斯滤波器计算二次微分会使得图像噪声更加明显，所以我们首先使用高斯滤波器来抑制噪声。

LoG 滤波器使用以下式子定义：

$$\text{LoG}(x, y) = \frac{x^2 + y^2 - s^2}{2 \cdot \pi \cdot s^6} \cdot e^{-\frac{x^2 + y^2}{2 \cdot s^2}}$$

输入 (imori_noise.jpg)	输出(answers_image/answer_19.jpg)
	

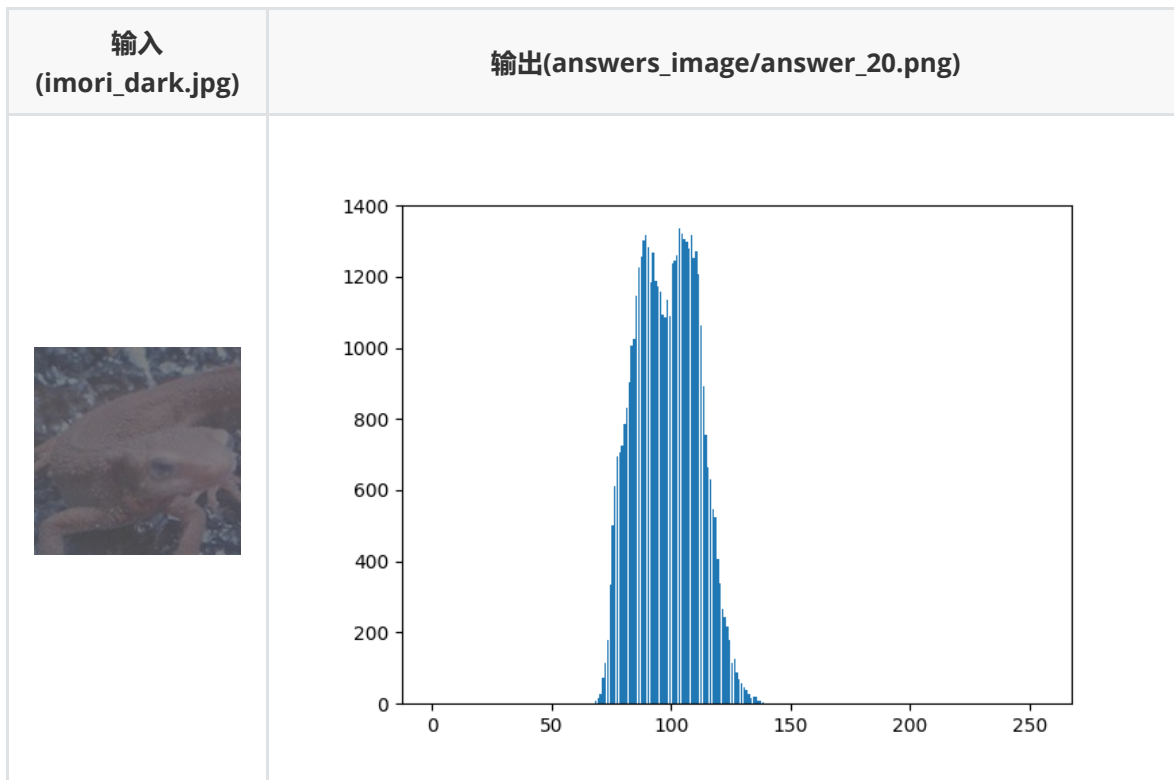
答案

- Python >> [answers_py/answer_19.py](#)
- C++ >> [answers_cpp/answer_19.cpp](#)

问题二十：直方图

使用 `Matplotlib` 来绘制 `imori_dark.jpg` 的直方图吧！

直方图显示了不同数值的像素出现的次数。在 `Matplotlib` 中有 `hist()` 函数提供绘制直方图的接口。



答案

- Python >> [answers_py/answer_20.py](#).

1. 原文是“微分フィルタ”，对应的英文应该是“Differential Filter”。考虑到这里处理离散的变量，应该是“差分滤波器”？问题十七：Laplacian滤波器中的“二次微分”没有什么太大歧义就没作处理。 [↩](#)