Computer Vision Homework 5　D07922015　謝銘峰

Write a program to generate (Using kernel 3-5-5-5-3):

Dilation

Erosion

Opening

Closing

(a) Dilation

```python
if __name__ == '__main__':
    from PIL import Image
    import numpy as np

    # Define kernel for dilation.
    kernel = np.array([\
        [0, 1, 1, 1, 0], \
        [1, 1, 1, 1, 1], \
        [1, 1, 1, 1, 1], \
        [1, 1, 1, 1, 1], \
        [0, 1, 1, 1, 0]])
    # Define center of kernel for dilation.
    centerKernel = (2, 2)
    # Load image from file.
    originalImage = Image.open('lena.bmp')
    # New image with the same size and 'grayscale' format.
    dilationImage = Image.new('L', originalImage.size)
    # Scan each column in original image.
    for r in range(originalImage.size[0]):
        # Scan each row in original image.
        for c in range(originalImage.size[1]):
            # Record local max. pixel value.
            localMaxPixel = 0
            # Scan each column in kernel.
            for x in range(kernel.shape[0]):
                # Scan each row in kernel.
                for y in range(kernel.shape[1]):
                    # Only check value '1' in kernel.
                    if (kernel[x, y] == 1):
                        # Calculate destination x, y position.
                        destX = r + (x - centerKernel[0])
                        destY = c + (y - centerKernel[1])
                        # Avoid out of image range.
                        if ((0 <= destX < originalImage.size[0]) and \
                            (0 <= destY < originalImage.size[1])):
                            # Get pixel value in original image at (destX, destY).
                            originalPixel = originalImage.getpixel((destX, destY))
                            # Update local max. pixel value.
                            localMaxPixel = max(localMaxPixel, originalPixel)
            # Paste local max. pixel value on original image.
            dilationImage.putpixel((r, c), localMaxPixel)
    # Save image fo file.
    dilationImage.save('dilation.bmp')
```

(b) Erosion

```
if __name__ == '__main__':
    from PIL import Image
    import numpy as np

    # Define kernel for erosion.
    kernel = np.array([\
        [0, 1, 1, 1, 0], \
        [1, 1, 1, 1, 1], \
        [1, 1, 1, 1, 1], \
        [1, 1, 1, 1, 1], \
        [0, 1, 1, 1, 0]])
    # Define center of kernel for erosion.
    centerKernel = (2, 2)
    # Load image from file.
    originalImage = Image.open('lena.bmp')
    # New image with the same size and 'grayscale' format.
    erosionImage = Image.new('L', originalImage.size)
    # Scan each column in original image.
    for r in range(originalImage.size[0]):
        # Scan each row in original image.
        for c in range(originalImage.size[1]):
            # Record local min. pixel value.
            localMinPixel = 255
            # Scan each column in kernel.
            for x in range(kernel.shape[0]):
                # Scan each row in kernel.
                for y in range(kernel.shape[1]):
                    # Only check value '1' in kernel.
                    if (kernel[x, y] == 1):
                        # Calculate destination x, y position.
                        destX = r + (x - centerKernel[0])
                        destY = c + (y - centerKernel[1])
                        # Avoid out of image range.
                        if ((0 <= destX < originalImage.size[0]) and \
                            (0 <= destY < originalImage.size[1])):
                            # Get pixel value in original image at (destX, destY).
                            originalPixel = originalImage.getpixel((destX, destY))
                            # Update local min. pixel value.
                            localMinPixel = min(localMinPixel, originalPixel)
            # Paste local min. pixel value on original image.
            erosionImage.putpixel((r, c), localMinPixel)
    # Save image fo file.
    erosionImage.save('erosion.bmp')
```

(c) Opening

```
def opening(originalImage, kernel, centerKernel):
    """
    :type originalImage: Image (from PIL)
    :type kernel: numpy array
    :type centerKernel: tuple
    :return type: Image (from PIL)
    """
    return dilation(erosion(originalImage, kernel, centerKernel), kernel, centerKernel)
```

(d) Closing

```
def closing(originalImage, kernel, centerKernel):
    """
    :type originalImage: Image (from PIL)
    :type kernel: numpy array
    :type centerKernel: tuple
    :return type: Image (from PIL)
    """
    return erosion(dilation(originalImage, kernel, centerKernel), kernel, centerKernel)
```