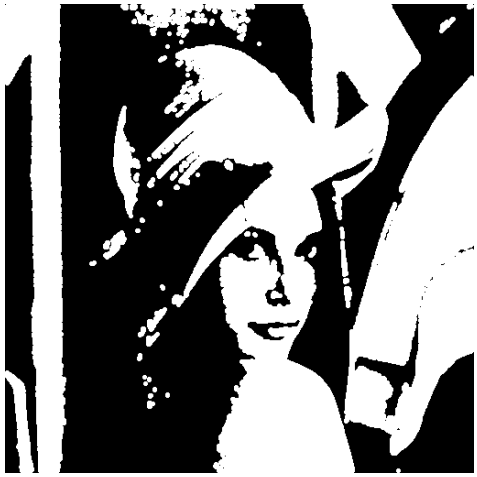
Computer Vision Homework 4　D07922015　謝銘峰

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

Dilation

Erosion



Opening

Closing

Hit-miss

(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]])
    # Load image from file.
    originalImage = Image.open('binary.bmp')
    # Get center position of kernel.
    centerKernel = tuple([x // 2 for x in kernel.shape])
    # New image with the same size and 'binary' format.
    dilationImage = Image.new('1', 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]):
            # Get pixel value in original image at (r, c).
            originalPixel = originalImage.getpixel((r, c))
            # If this pixel is object (1, white).
            if (originalPixel != 0):
                # Paste kernel on original image at (r, c).
                # 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 paste '1' value from 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])):
                                # Paste '1' value on original image.
                                dilationImage.putpixel((destX, destY), 1)

    # Save image fo file.
    dilationImage.save('dilation.bmp')
```

(b) Erosion

```
    # Get center position of kernel.
    centerKernel = tuple([x // 2 for x in kernel.shape])
    # New image with the same size and 'binary' format.
    erosionImage = Image.new('1', 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]):
            # Flag of match.
            matchFlag = True
            # 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 '1' value from 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])):
                            # If this point doesn't match with kernel.
                            if (originalImage.getpixel((destX, destY)) == 0):
                                # Clear flag of match.
                                matchFlag = False
                                break
                        # It is edge point, it will never match.
                        else:
                            # Clear flag of match.
                            matchFlag = False
                            break
            # Full kernel is match in original image at (r, c).
            if (matchFlag):
                # Paste '1' value on original image.
                erosionImage.putpixel((r, c), 1)
```

(c) Opening

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

(d) Closing

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

(e) Hit-and-miss transform

```python
def intersection(image1, image2):
    """
    :type image1: Image (from PIL)
    :type image2: Image (from PIL)
    :return type: Image (from PIL)
    """
    from PIL import Image
    # New image with the same size and 'binary' format.
    intersectionImage = Image.new('1', image1.size)
    # Scan each column in image 1.
    for r in range(image1.size[0]):
        # Scan each row in image 1.
        for c in range(image1.size[1]):
            # Get pixel value in image 1 at (r, c).
            image1Pixel = image1.getpixel((r, c))
            # Get pixel value in image 2 at (r, c).
            image2Pixel = image2.getpixel((r, c))
            # If those pixels are object (1, white).
            if (image1Pixel != 0 and image2Pixel != 0):
                # Paste '1' value on intersection image.
                intersectionImage.putpixel((r, c), 1)
            else:
                # Paste '0' value on intersection image.
                intersectionImage.putpixel((r, c), 0)
    return intersectionImage
```

```python
def erosionWithCenter(originalImage, kernel, centerKernel):
    """
    :type originalImage: Image (from PIL)
    :type kernel: numpy array
    :type centerKernel: tuple
    :return type: Image (from PIL)
    """
    from PIL import Image
    # New image with the same size and 'binary' format.
    erosionImage = Image.new('1', 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]):
            # Flag of match.
            matchFlag = True
            # 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 '1' value from 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])):
                            # If this point doesn't match with kernel.
                            if (originalImage.getpixel((destX, destY)) == 0):
                                # Clear flag of match.
                                matchFlag = False
                                break
                        # It is edge point, it will never match.
                        else:
                            # Clear flag of match.
                            matchFlag = False
                            break
            # Full kernel is match in original image at (r, c).
            if (matchFlag):
                # Paste '1' value on original image.
                erosionImage.putpixel((r, c), 1)
    # Return erosion image.
    return erosionImage
```

```python
def hitmiss(originalImage, kernel_J, centerKernel_J, kernel_K, centerKernel_K):
    """
    :type originalImage: Image (from PIL)
    :type kernel_J: numpy array
    :type centerKernel_J: tuple
    :type kernel_K: numpy array
    :type centerKernel_K: tuple
    :return type: Image (from PIL)
    """
    return intersection(erosionWithCenter(originalImage, kernel_J, centerKernel_J),
                erosionWithCenter(complement(originalImage), kernel_K, centerKernel_K))
```