Computer Vision 2019 Fall

Homework #4

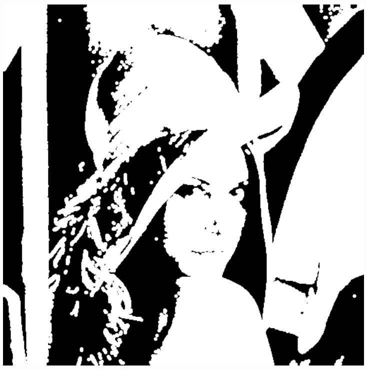
B06902059 資工三 謝宜儒

Description

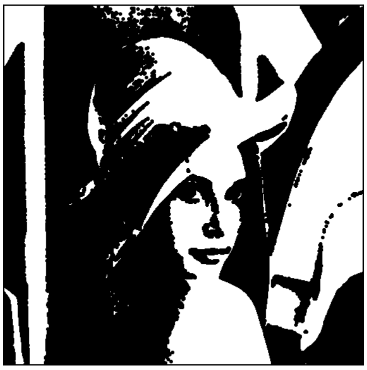
This homework focuses on binary morphology on a image.

Results

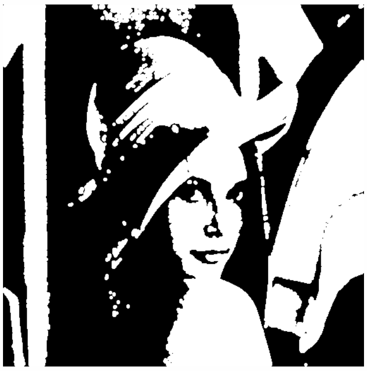
(a) Dilation



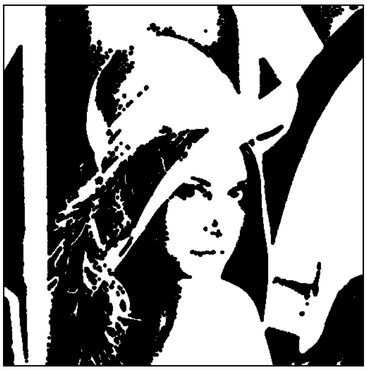
(b) Erosion



(c) Opening



(d) Closing



(e) Hit-and-miss transform

Source Code (fragment)

```
octogon = 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]])

L = np.array([[1, 1],
        [0, 1]])

class Kernel:
  def __init__(self, origin, pattern):
    offsets = []
    for i in range(pattern.shape[0]):
      for j in range(pattern.shape[1]):
        if(pattern[i][j] == 1):
          offsets.append([i - origin[0], j - origin[1]])
    self._offsets = offsets

  def offsets(self):
    return self._offsets

def binarize_img(img):
  binary_128_img = np.zeros((length, width))
  threshold = 128
  for i in range(length):
    for j in range(width):
      if(img[i][j] < threshold):
        binary_128_img[i][j] = 0
      else:
        binary_128_img[i][j] = 255

  return binary_128_img

def plot_img(img, filename, title):
  cv2.imwrite(filename, img)
  cv2.imshow(title, img)
  cv2.waitKey(0)
  cv2.destroyAllWindows()

def intersection(img1, img2):
  intersection_img = np.zeros((length, width))
  for i in range(length):
    for j in range(width):
      if(img1[i][j] == img2[i][j] and img1[i][j] == 255):
        intersection_img[i][j] = 255

  return intersection_img

def complement(img):
  complement_img = np.copy(img)
  for i in range(length):
    for j in range(width):
      complement_img[i][j] = 255 - complement_img[i][j]

  return complement_img

def dilation(img, kernel):
  dilated_img = np.zeros((length, width))
  for i in range(length):
    for j in range(width):
      if(img[i][j] == 255):
        for offset in kernel.offsets():
          if(i + offset[0] >= 0 and i + offset[0] < length
            and j + offset[1] >=0 and j + offset[1] < width):
            dilated_img[i + offset[0]][j + offset[1]] = 255
  return dilated_img
```

```
def erosion(img, kernel):
  eroded_img = np.zeros((length, width))
  for i in range(length):
    for j in range(width):
      valid = 1
      for offset in kernel.offsets():
        if(i + offset[0] >= 0 and i + offset[0] < length
          and j + offset[1] >= 0 and j + offset[1] < width):
          if(img[i + offset[0]][j + offset[1]] != 255):
            valid = 0
            break
        else:
          valid = 0
          break
      if(valid):
        eroded_img[i][j] = 255

  return eroded_img

def opening(img, kernel):
  return dilation(erosion(img, kernel), kernel)

def closing(img, kernel):
  return erosion(dilation(img, kernel), kernel)

def hit_and_miss(img, j_kernel, k_kernel):
  return intersection(erosion(img, j_kernel), erosion(complement(img),
k_kernel))
```

To run the source code, type the following line in a terminal:

```
python3 hw4.py [input image] [problem number]
```

where in this homework, the input image is **lena.bmp** and the problem numbers are **1 ~ 5** .