Computer Vision HW5: Mathematical Morphology – Gray Scaled Morphology

R10741015 鄭傑鴻

Oct. 08, 2022

Original Binary Image:

Mutual Parameters:

- **pic: original Lena.bmp image**
- **Kernel: (3,5,5,5,3): Disk shaped kernel for dilation/erosion/closing/opening**

Write programs which do gray-scale morphology on a gray-scale image:

(a) Dilation

- Description:
  Using the (3,5,5,5,3) kernel to perform dilation on the binary image, making each pixel the maximum value in the kernel.
- Algorithm:
  For every pixel in the gray-scale image:
  Apply the kernel and record the values within
  Assign the maximum value to the pixel
- Code:

```
def getValInKernel(self, refer, row, col):
    intensity_lst = []
    for point in self.kernel:
```

```
        if self.inRange(row+point[0], col+point[1]):
            intensity_lst.append(refer[row+point[0]][col+point[1]])
    return intensity_lst

def dilation(self, tgtpic):
    dilation_pic = np.copy(tgtpic)
    for i in range(dilation_pic.shape[0]):
        for j in range(dilation_pic.shape[1]):
            dilation_pic[i][j] = max(self.getValInKernel(tgtpic, i, j))
    return dilation_pic

def sequential(self):
    # Dilation
    dilation_pic = self.dilation(self.pic)
    cv2.imwrite('lena_dilation.bmp', dilation_pic)
    ……
```

- Resulting Image:

(b) Erosion

- Description

  Using the (3,5,5,5,3) kernel to perform erosion on the gray-scale image, making each pixel the minimum value in the kernel.

- Algorithm

  For every pixel in the gray-scale image:

  Apply the kernel and record the values within

  Assign the minimum value to the pixel

- Code:

```
def getValInKernel(self, refer, row, col):
    intensity_lst = []
    for point in self.kernel:
```

```
            if self.inRange(row+point[0], col+point[1]):
                intensity_lst.append(refer[row+point[0]][col+point[1]])
        return intensity_lst

    def erosion(self, tgtpic):
        erosion_pic = np.copy(tgtpic)
        for i in range(erosion_pic.shape[0]):
            for j in range(erosion_pic.shape[1]):
                erosion_pic[i][j] = min(self.getValInKernel(tgtpic, i, j))
        return erosion_pic

    def sequential(self):
    ……
    # Erosion
    erosion_pic = self.erosion(self.pic)
    cv2.imwrite('lena_erosion.bmp', erosion_pic)
    ……
```

- Resulting Image:

(c) Opening

- Description/Algorithm:

  Perform erosion first, then apply dilation. In practice, call the erosion function in part (b), then call the dilation function in part (a)

- Code:

```
    def opening(self, tgtpic):
        opening_pic = self.erosion(tgtpic)
        opening_pic = self.dilation(opening_pic)
        return opening_pic

    def sequential(self):
    ……
    # Opening
    opening_pic = self.opening(self.pic)
    cv2.imwrite('lena_opening.bmp', opening_pic)
    ……
```

- Resulting Image:

(d) Closing

- Description/Algorithm:

  Perform dilation first, then apply erosion. In practice, call the dilation function in part (a), then call the erosion function in part (b)

- Code:

```
def closing(self, tgtpic):
    # Dilation -> Erosion
    closing_pic = self.dilation(tgtpic)
    closing_pic = self.erosion(closing_pic)
    return closing_pic
def sequential(self):
……
# Closing
closing_pic = self.closing(self.pic)
cv2.imwrite('lena_closing.bmp', closing_pic)
……
```

- Resulting Image