东南大学
网络空间安全学院

形态学实验

学生姓名: 李盛； 学号： *229221*

Course: 图象分析与理解 – Professor: 季续
Due date: *4* 月 *11* 日， *2022* 年

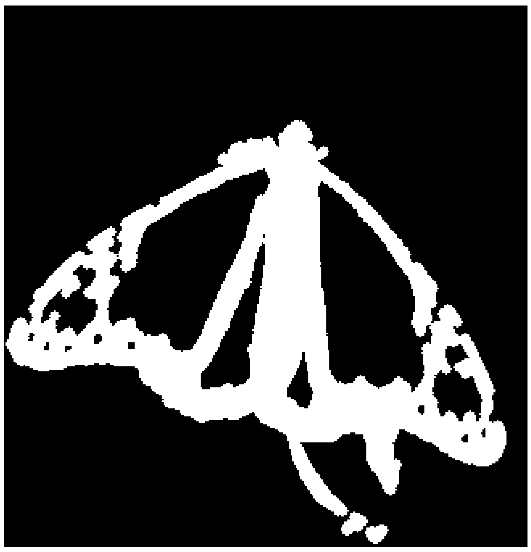
第一题

形态学实验：二值图像

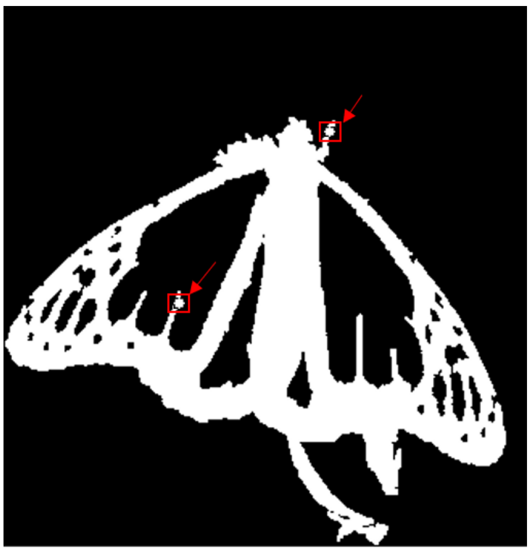
Solution.

1. Opening and closing(Let $SE_1 = B$)

   (a) The result of $I_0 \circ B$ is shown in Figure 1a. The thin protrusions are eliminated and the contour of the butterfly is smoothed by opening as shown in Figure 1b.



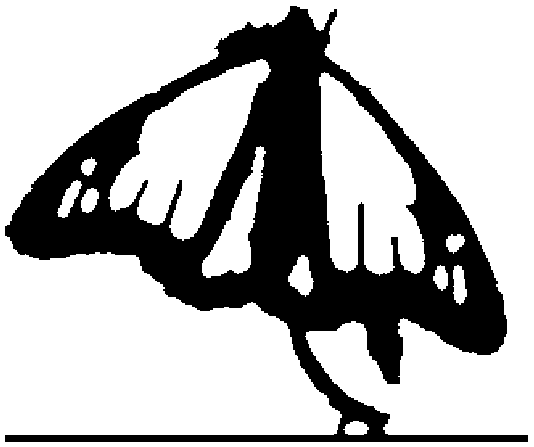
(a) $I_0 \circ B$



(b) Structure Element on top of I_0

Figure 1: Illustration of $I_0 \circ B$

   (b) The result of $I_0^c \circ \widehat{B}$ is shown in Figure 2a. The holes of the butterfly wing are narrowed or eliminated by opening as shown in Figure 2b.


(a) $I_0^c \circ \widehat{B}$

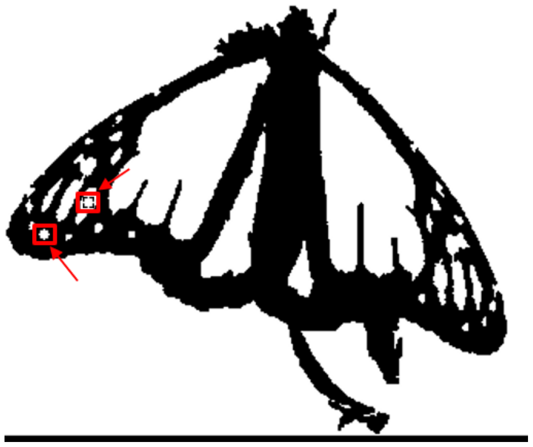
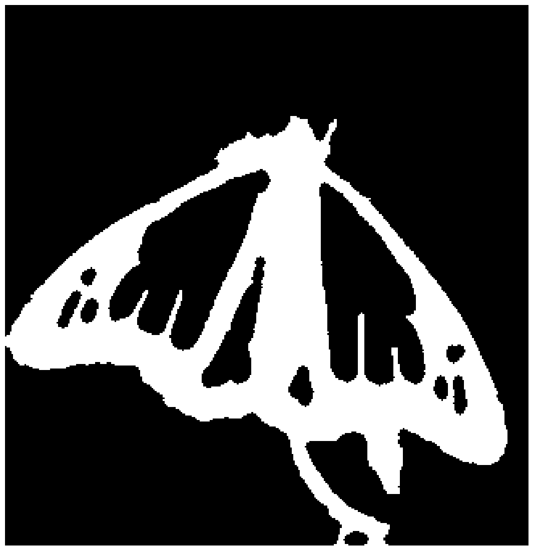

(b) Structure Element on top of I_0^c

Figure 2: Illustration of $I_0^c \circ \widehat{B}$

(c) The results of $I_0 \cdot B$ and $(I_0 \cdot B)^c$ are shown in Figure 3a and 3b respectively. Since the result of comparing the numpy arrays of $(I_0 \cdot B)^c$ and $I_0^c \circ \widehat{B}$ is True, we demonstrate the correctness of the equation $(I_0 \cdot B)^c = I_0^c \circ \widehat{B}$.


(a) $I_0 \cdot B$

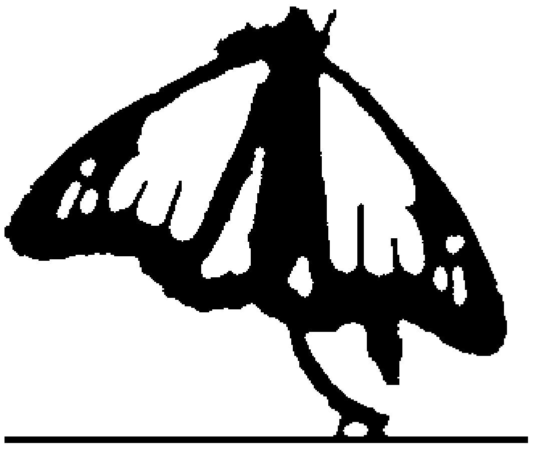
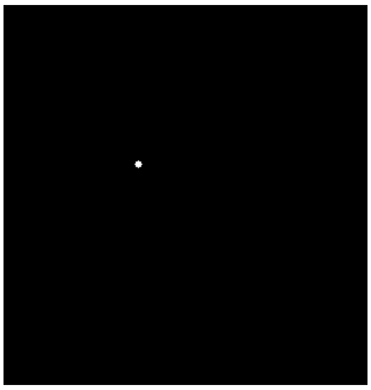
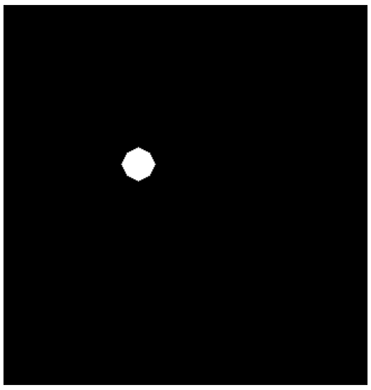

(b) $(I_0 \cdot B)^c$

Figure 3: Illustration of $I_0 \cdot B$

2. Image reconstruction


(a) $I_{r,1}$


(b) $I_{r,5}$

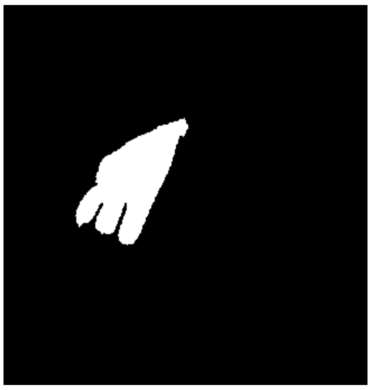
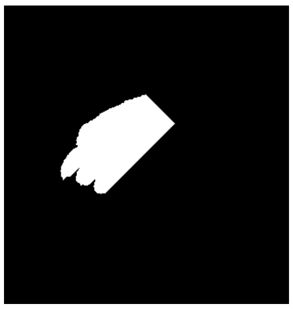
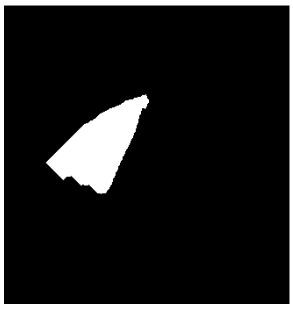
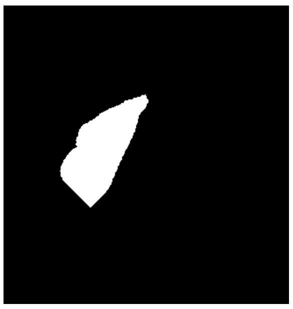

(c) $I_{r,\infty}$

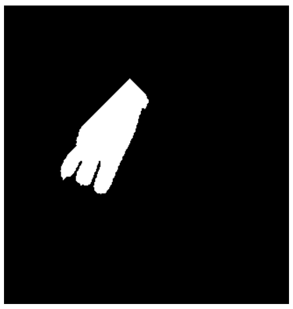
Figure 4: Figures of $I_{r,1}$, $I_{r,5}$ and $I_{r,\infty}$ iterated from the marker I_m

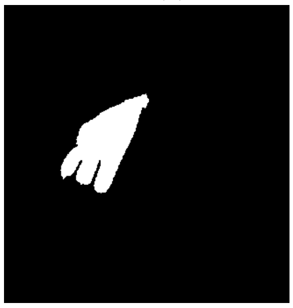
3. Convex hull


(a) $I_{ch,1,\infty}$


(b) $I_{ch,2,\infty}$


(c) $I_{ch,3,\infty}$


(d) $I_{ch,4,\infty}$


(e) $I_{r,\infty}$

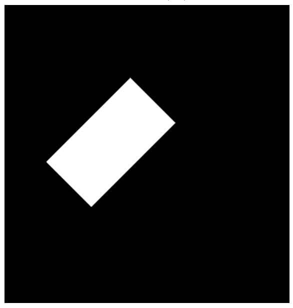

(f) I_{ch}

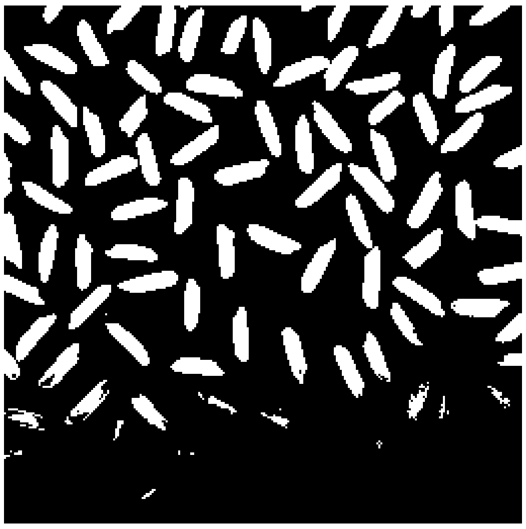
Figure 5: Results of $I_{ch,i,\infty}(i = 1, 2, 3, 4)$ and I_{ch} iterated from $I_{r,\infty}$

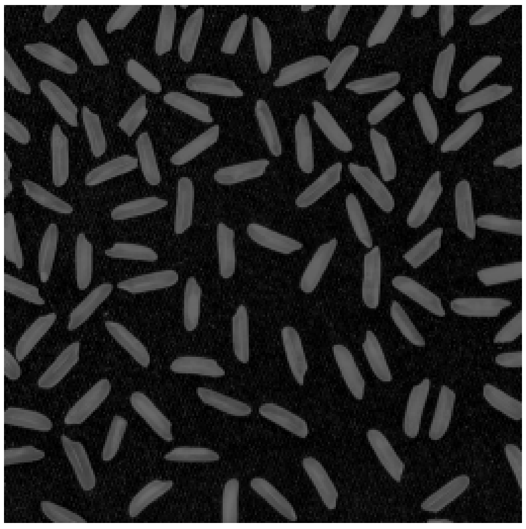
第二题

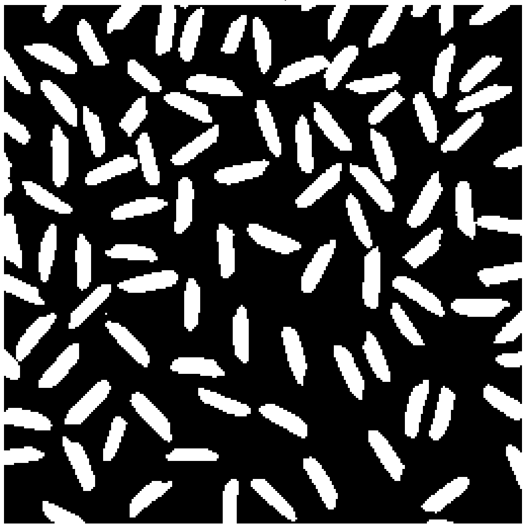
形态学实验: 灰度图像

Solution.

1. Utilize Top-hat transform to eliminate the uniform background.
    (a) Although most of noise is eliminated in the original image as is shown in Figure 7a, some rice at the bottom is also erased by the threshold. It is therefore difficult to choose the threshold for the figure with uniform background.
    (b) There is no negative value in $I_{top-hat}$ since the pixel values which are calculated by gray-scale opening are not greater than the originals for each pixel position.
    (c) The uniform background is almost eliminated by Top-hat transform after threshold segmentation as shown in Figure 7c.


(a) $I_{0,th}$


(b) $I_{top-hat}$


(c) $I_{top-hat,th}$

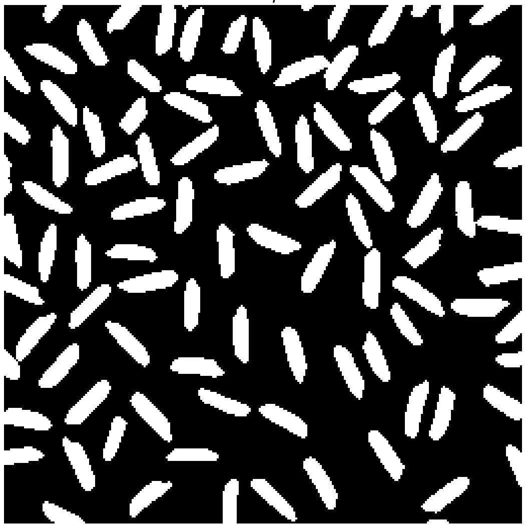
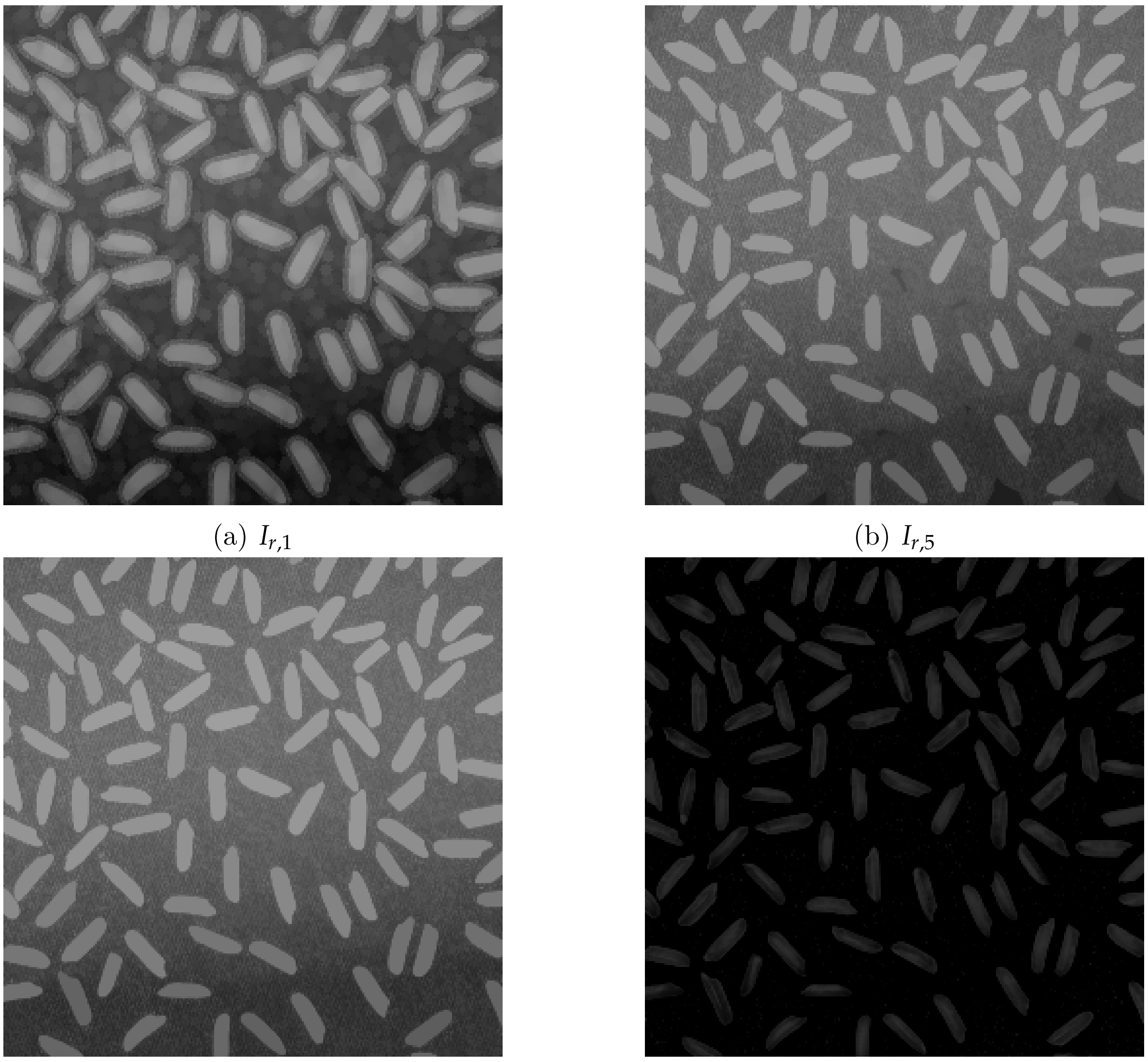

(d) $I_{top-hat,th} \circ se\_2$

Figure 6: Top-hat transform

2. H-dome



(a) $I_{r,1}$ (b) $I_{r,5}$

(c) $I_{r,\infty}$ (d) $I_0 - I_{r,\infty}$

Figure 7: H-dome: results of $I_{r,1}$, $I_{r,5}$, $I_{r,\infty}$ and $I_0 - I_{r,\infty}$

Appendix: Implementation code with Python

Helper Function

```python
import numpy as np
import cv2 as cv
from cv2 import BORDER_REFLECT

def read_txt(file):
# Read txt file into the numpy array
    array = np.genfromtxt(file, dtype=int, \
                          encoding=None, delimiter=",")
    binary_array = np.where(array>0, 255, 0)
    binary_array = np.asarray(binary_array, dtype="uint8" )
    return binary_array

def iteration(I_start,se1,Is):
    I_finish = cv.bitwise_and\
            (cv.dilate(I_start, se1, iterations=1), Is)
    if np.array_equal(I_start, I_finish):
        return I_finish
    else:
        return iteration(I_finish,se1,Is)

def convex_iteration(I_start,B1,B2):
    e1 = cv.erode(I_start,B1,iterations=1,\
                borderType = BORDER_REFLECT)
    e2 = cv.erode(cv.bitwise_not(I_start),B2, iterations=1)
    I_finish = cv.bitwise_or((cv.bitwise_and(e1,e2)),I_start)
    if np.array_equal(I_start, I_finish):
        return I_finish
    else:
        return convex_iteration(I_finish,B1,B2)

def mr_iteration(I_start, B, I0):
    I_finish = cv.min((cv.dilate(I_start, B)), I0)
    if np.array_equal(I_start, I_finish):
        return I_finish
    else:
        return mr_iteration(I_finish, B, I0)
```

Morphology

```python
#!/usr/bin/env python3
import cv2 as cv
from helper_function import read_txt, iteration,convex_iteration
import numpy as np

############################################################
################### Opening and Closing ####################
```

```
##############################################################

# load image txt file and save the np array as png
I0 = read_txt("../resource/ass1/butterfly.txt")
se1 = np.loadtxt("../resource/ass1/se_1.txt", \
                    delimiter=",").astype(np.uint8)

# I0 opening by se_1
I0_opening = cv.morphologyEx(I0, cv.MORPH_OPEN, se1)

# Complement of I0 opening by se_1
I0_comp = cv.bitwise_not(I0)
se1_hat = se1
I0_comp_opening = cv.morphologyEx(I0_comp, cv.MORPH_OPEN, se1_hat)

# I0 closing by B and take the complement of the result
I0_closing = cv.morphologyEx(I0, cv.MORPH_CLOSE, se1)
I0_closing_comp = cv.bitwise_not(I0_closing)

I0_opening_test = cv.imread("../img/Mor/I0_opening.png",0)
# Save results as png
cv.imwrite('../img/Mor/butterfly.png',I0)
cv.imwrite('../img/Mor/se_1.png',se1)
cv.imwrite('../img/Mor/I0_opening.png', I0_opening)
cv.imwrite('../img/Mor/I0_complement.png', I0_comp)
cv.imwrite('../img/Mor/I0_complement_opening.png', I0_comp_opening)
cv.imwrite('../img/Mor/I0_closing.png', I0_closing)
cv.imwrite('../img/Mor/I0_closing_complement.png', I0_closing_comp)

# Compare I0_closing_comp and I0_comp_opening
print(np.array_equal(I0_closing_comp, I0_comp_opening))

# ############################################################
# ############# Morphological Reconstruction ################
# ############################################################
# load the image of marker
Im = read_txt("../resource/ass1/marker.txt")
cv.imwrite("../img/marker.png", Im)

Is = I0_comp
I1 = cv.bitwise_and(cv.dilate(Im, se1, iterations=1), Is)
I2 = cv.bitwise_and(cv.dilate(I1, se1, iterations=1), Is)
I3 = cv.bitwise_and(cv.dilate(I2, se1, iterations=1), Is)
I4 = cv.bitwise_and(cv.dilate(I3, se1, iterations=1), Is)
I5 = cv.bitwise_and(cv.dilate(I4, se1, iterations=1), Is)
Ir_infinite = iteration(I5,se1,Is)

cv.imwrite("../img/Ir1.png", I1)
cv.imwrite("../img/Ir5.png", I5)
cv.imwrite("../img/Ir_infinite.png", Ir_infinite)

# ##################################################################
# ######################## Convex Hull #########################
# ##################################################################
B11 = read_txt("../resource/ass1/convex_hull_se/B11.txt")
B12 = read_txt("../resource/ass1/convex_hull_se/B12.txt")
B21 = read_txt("../resource/ass1/convex_hull_se/B21.txt")
B22 = read_txt("../resource/ass1/convex_hull_se/B22.txt")
B31 = read_txt("../resource/ass1/convex_hull_se/B31.txt")
B32 = read_txt("../resource/ass1/convex_hull_se/B32.txt")
B41 = read_txt("../resource/ass1/convex_hull_se/B41.txt")
B42 = read_txt("../resource/ass1/convex_hull_se/B42.txt")

I_ch_0 = Ir_infinite
# Compute I_ch_i_infi
I_ch_1_infi =  convex_iteration(I_ch_0, B11, B12)
I_ch_2_infi =  convex_iteration(I_ch_0, B21, B22)
I_ch_3_infi =  convex_iteration(I_ch_0, B31, B32)
I_ch_4_infi =  convex_iteration(I_ch_0, B41, B42)
I_ch12 = cv.bitwise_or(I_ch_1_infi,I_ch_2_infi)
I_ch34 = cv.bitwise_or(I_ch_3_infi,I_ch_4_infi)
I_ch = cv.bitwise_or(I_ch12,I_ch34)

# Save results as png
cv.imwrite("../img/I_ch_1_infi.png",I_ch_1_infi)
cv.imwrite("../img/I_ch_2_infi.png",I_ch_2_infi)
cv.imwrite("../img/I_ch_3_infi.png",I_ch_3_infi)
cv.imwrite("../img/I_ch_4_infi.png",I_ch_4_infi)
cv.imwrite("../img/I_ch.png",I_ch)

# Check if I_ch is the convex hull
print(np.array_equal(cv.bitwise_or(I_ch,I_ch_0), I_ch))
```

Grayscale

```
#!/usr/bin/env python3
import cv2 as cv
from helper_function import read_txt, mr_iteration
import numpy as np

# Top-hat transform
I0 = cv.imread("../resource/ass2/rice.png", cv.IMREAD_GRAYSCALE)
I0_th = I0.copy()
I0_th[I0_th <= 150] = 0
I0_th[I0_th > 150] = 255

# compute I_top-hat
```

```
se1 =  np.loadtxt("../resource/ass2/se_1.txt",\
                 delimiter=",").astype(np.uint8)
I0_opening = cv.morphologyEx(I0, cv.MORPH_OPEN, se1)
I_tophat = I0   I0_opening

# compute I_top−hat−th
I_tophat_th = I_tophat.copy()
I_tophat_th[I_tophat_th <= 60] = 0
I_tophat_th[I_tophat_th > 60] = 255

# opening, remove the light point
se2 = np.loadtxt("../resource/ass2/se_2.txt", \
                 delimiter=",").astype(np.uint8)
I_remove = cv.morphologyEx(I_tophat_th, cv.MORPH_OPEN, se2)

cv.imwrite('../img/Gray/I0th.png',I0_th)
cv.imwrite('../img/Gray/I_tophat.png',I_tophat)
cv.imwrite('../img/Gray/I_tophat_th.png',I_tophat_th)
cv.imwrite('../img/Gray/I_remove.png',I_remove)

# H−dome
Im = I0.copy()
Im = np.int16(Im) # marker image
Im = Im   45
B3 = np.loadtxt("../resource/ass2/se_3.txt", \
                 delimiter=",").astype(np.uint8)

Ir0 = Im
Ir1 = cv.min((cv.dilate(Ir0,B3)),np.int16(I0))
Ir2 = cv.min((cv.dilate(Ir1,B3)),np.int16(I0))
Ir3 = cv.min((cv.dilate(Ir2,B3)),np.int16(I0))
Ir4 = cv.min((cv.dilate(Ir3,B3)),np.int16(I0))
Ir5 = cv.min((cv.dilate(Ir4,B3)),np.int16(I0))
Ir_infi = mr_iteration(Ir0, B3, np.int16(I0))

cv.imwrite("../img/Gray/B3.png",B3)
cv.imwrite("../img/Gray/Imarker.png", Im)
cv.imwrite("../img/Gray/Ir1.png", Ir1)
cv.imwrite("../img/Gray/Ir5.png", Ir5)
cv.imwrite("../img/Gray/Ir_infi.png", Ir_infi)
cv.imwrite("../img/Gray/I0 Ir_infi.png", I0 Ir_infi)
```