黃瑞得
0860078
huangjuite@gmail.com

project6

source code

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import math
import pandas as pd
import copy


def otsu(img):
    mean_w = 1/(img.shape[0]*img.shape[1])
    his, bins = np.histogram(img, np.array(range(0, 257)))
    final_thres = -1
    final_value = -1
    intensity = np.arange(256)
    variance = np.zeros(256)
    for t in bins[1:-1]:
        pcb = np.sum(his[:t])
        pcf = np.sum(his[t:])
        wb = pcb * mean_w
        wf = pcf * mean_w

        mub = np.sum(intensity[:t]*his[:t]) / float(pcb)
        muf = np.sum(intensity[t:]*his[t:]) / float(pcf)

        value = wb*wf*(mub-muf)**2
        variance[t-1] = value

        if(value > final_value):
            final_thres = t
            final_value = value
    print("final thres: ", final_thres)
    new_img = img.copy()
    new_img[img > final_thres] = 255
    new_img[img < final_thres] = 0
    return new_img, variance
```

黃瑞得
0860078
huangjuite@gmail.com

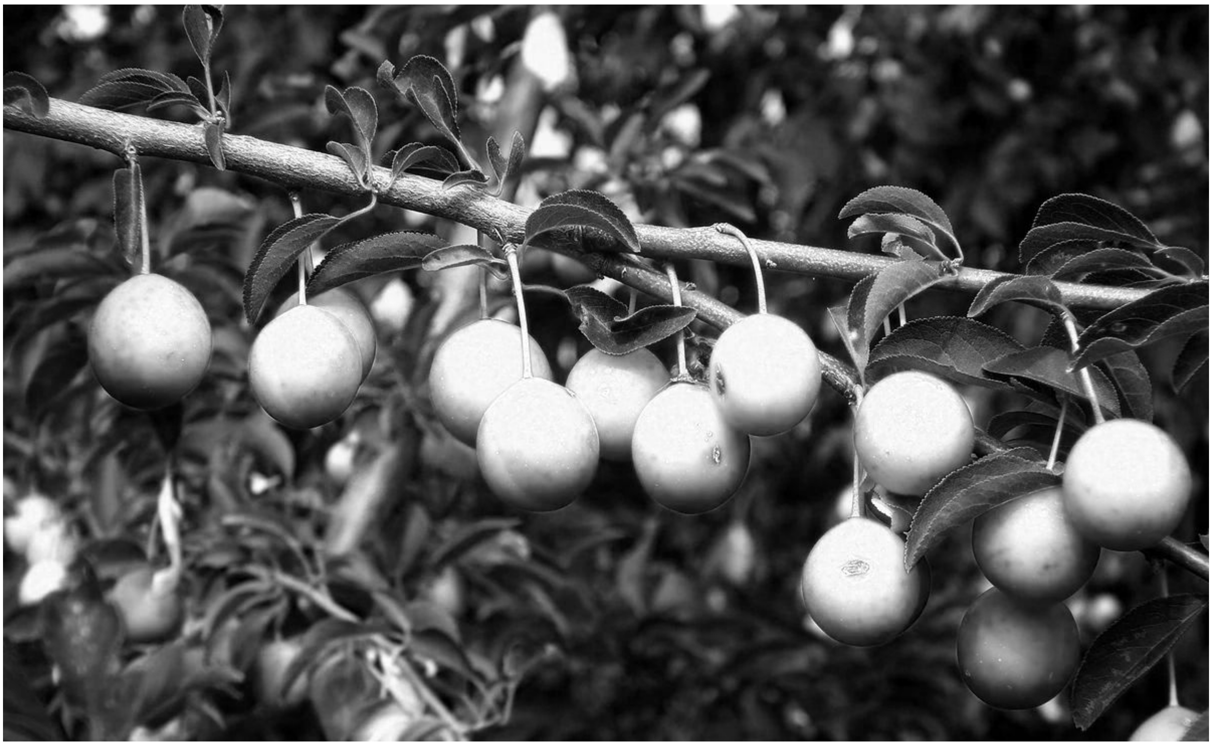
```
def kmeans(img, T=1, k=2):
    mask = None
    iteration = 0
    previous_centroid = None
    distance = np.zeros((img.shape[0], img.shape[1], k))
    rc = np.random.randint(img.shape[0], size=k)
    rr = np.random.randint(img.shape[1], size=k)
    centroid = np.zeros((k, img.shape[2]))
    for i, (c, r) in enumerate(zip(rc, rr)):
        centroid[i] = img[c, r]

    print("kmeans----------------------")
    while 1:
        for i in range(k):
            distance[:, :, i] = np.linalg.norm(img-centroid[i], axis=2)
        mask = np.argmin(distance, axis=2)

        for i in range(k):
            centroid[i] = np.mean(img[np.where(mask == i)], axis=0)

        if previous_centroid is not None:
            error = np.linalg.norm(previous_centroid-centroid, axis=1)
            print(iteration, error)
            if np.max(error) < T:
                break

        iteration += 1
        previous_centroid = centroid.copy()

    v = np.linalg.norm(centroid, axis=1)
    if v[0] > v[1]:
        mask = 1 - mask
    return mask*255

def filter(img, mask):
    filtered = img.copy()
    filtered[np.where(mask == 0)] = [80, 80, 80]
    return filtered[:, :, ::-1]






# read image
img = plt.imread('fruit on tree.tif')
# img = np.float32(img)/255.0
img_r = img[:, :, 0].copy()
print(img_r.shape)
cv2.imwrite("img_r.png", img_r)
```

```
mask, variances = otsu(img_r)
cv2.imwrite("otsu_mask.png", mask)
plt.plot(variances)
plt.plot(np.argmax(variances), np.max(variances),
        marker="X", label="maximum")
plt.title("variance between class under different threshold")
plt.xlabel("intensity")
plt.ylabel("variance")
plt.legend()
plt.savefig("variances.png")
plt.close()

filtered = filter(img, mask)
cv2.imwrite("otsu_filtered.png", filtered)

for T in [1,5,10]:
   kmeans_mask = kmeans(img, T=T)
   cv2.imwrite("kmeans_mask_%d.png"%T, kmeans_mask)
   kmean_filter = filter(img, kmeans_mask)
   cv2.imwrite("kmean_filter_%d.png"%T, kmean_filter)
```

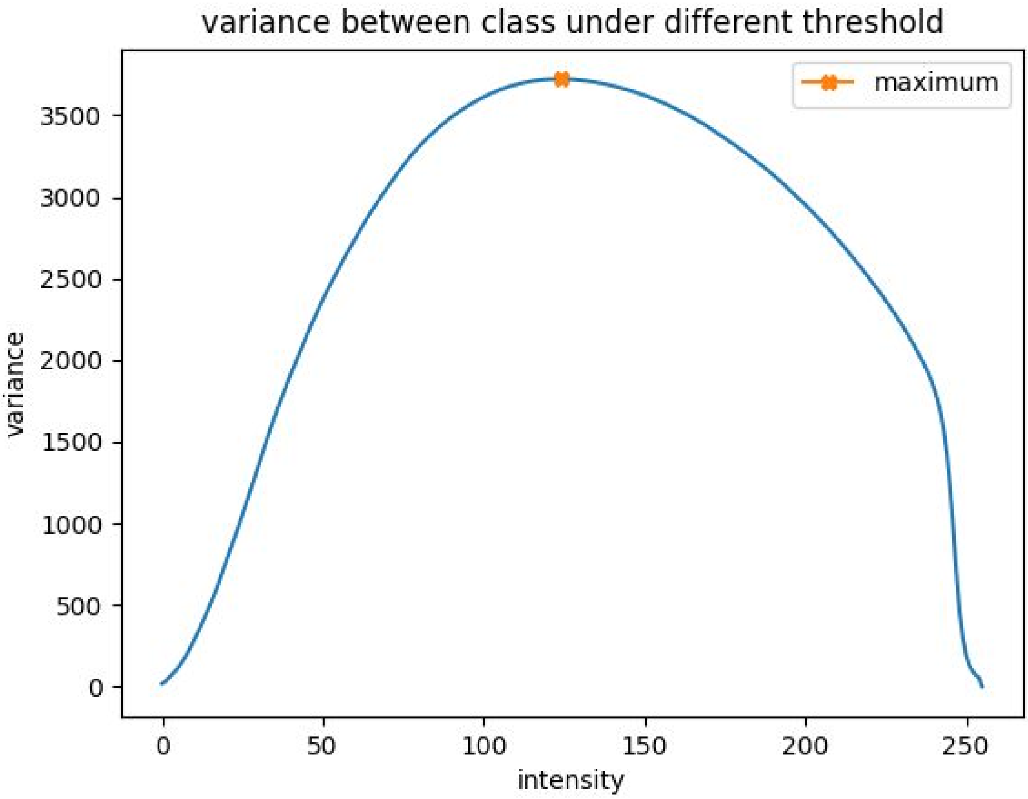
黃瑞得
0860078
huangjuite@gmail.com

Otsu’s method

R component

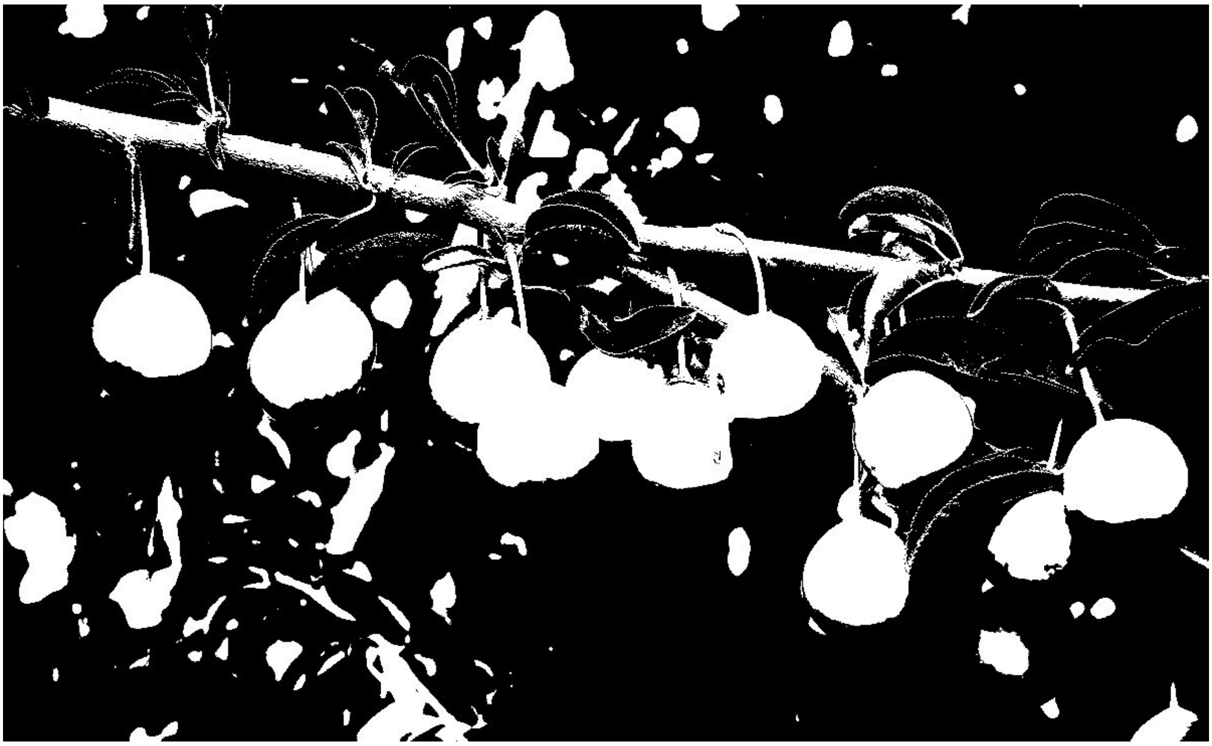


variance between class, argmax = 125

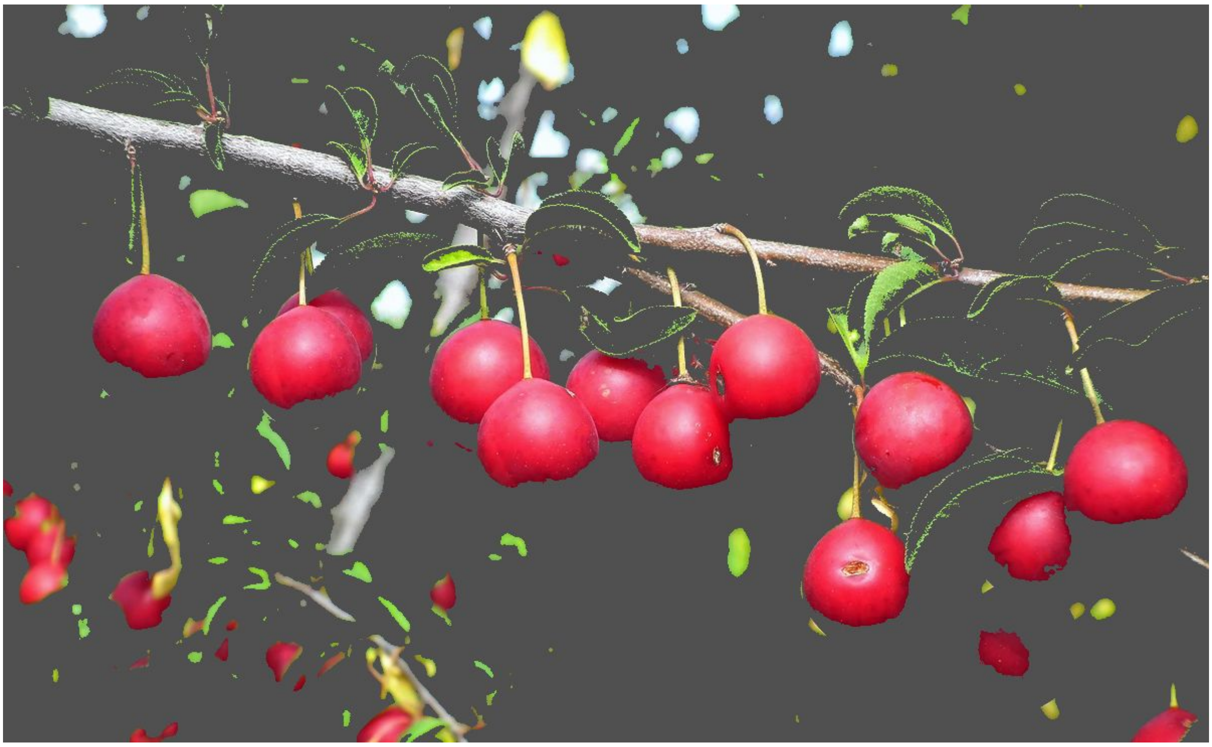


黃瑞得
0860078
huangjuite@gmail.com

threshold mask



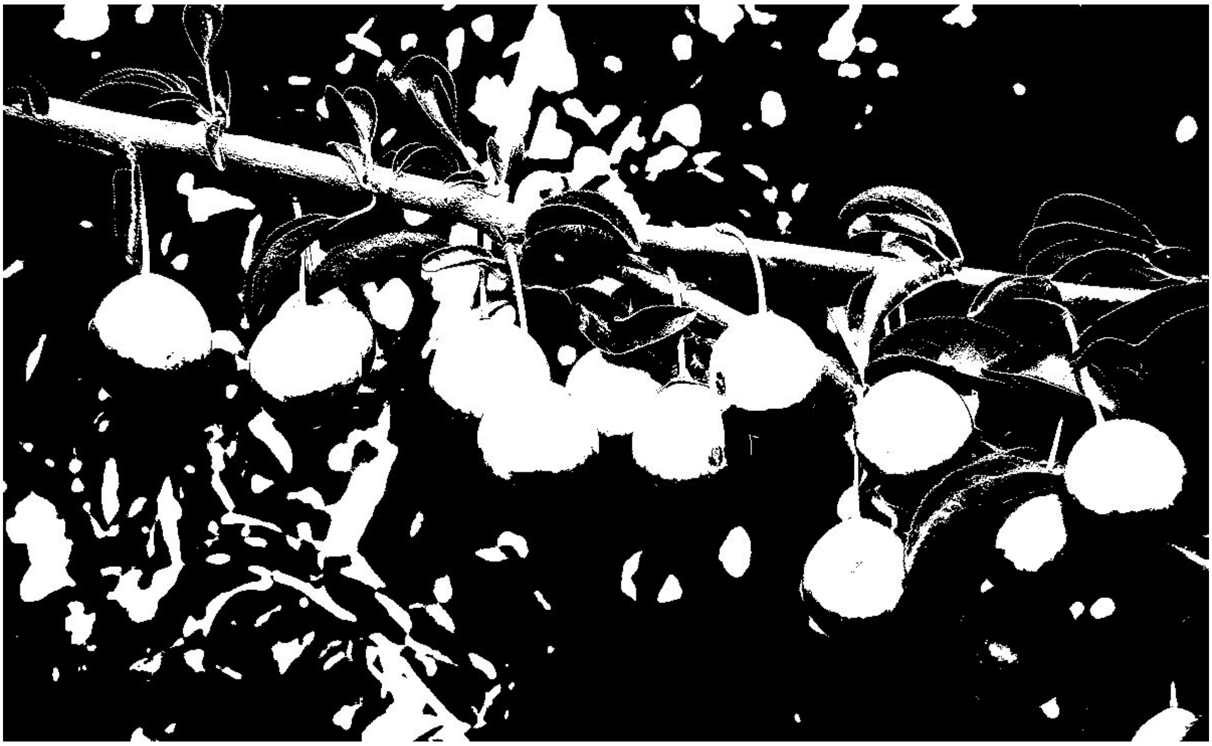
filtered image

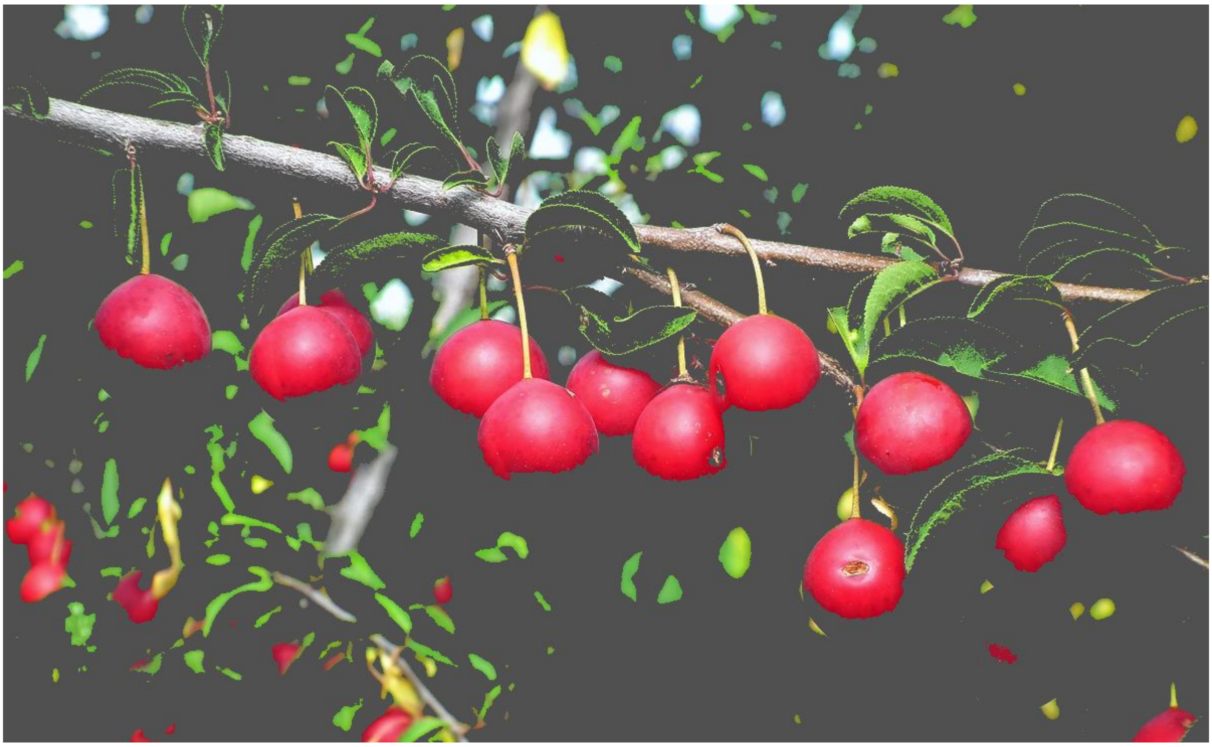


黃瑞得
0860078
huangjuite@gmail.com

k-means clustering

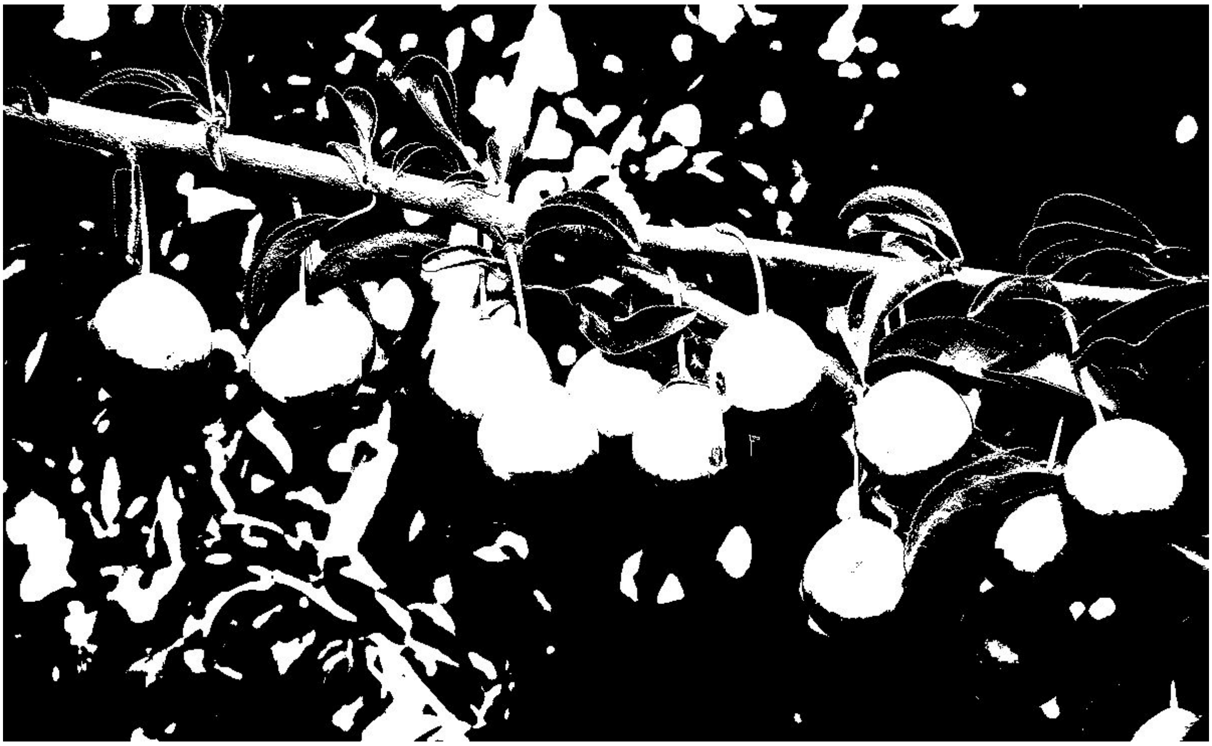
T=1

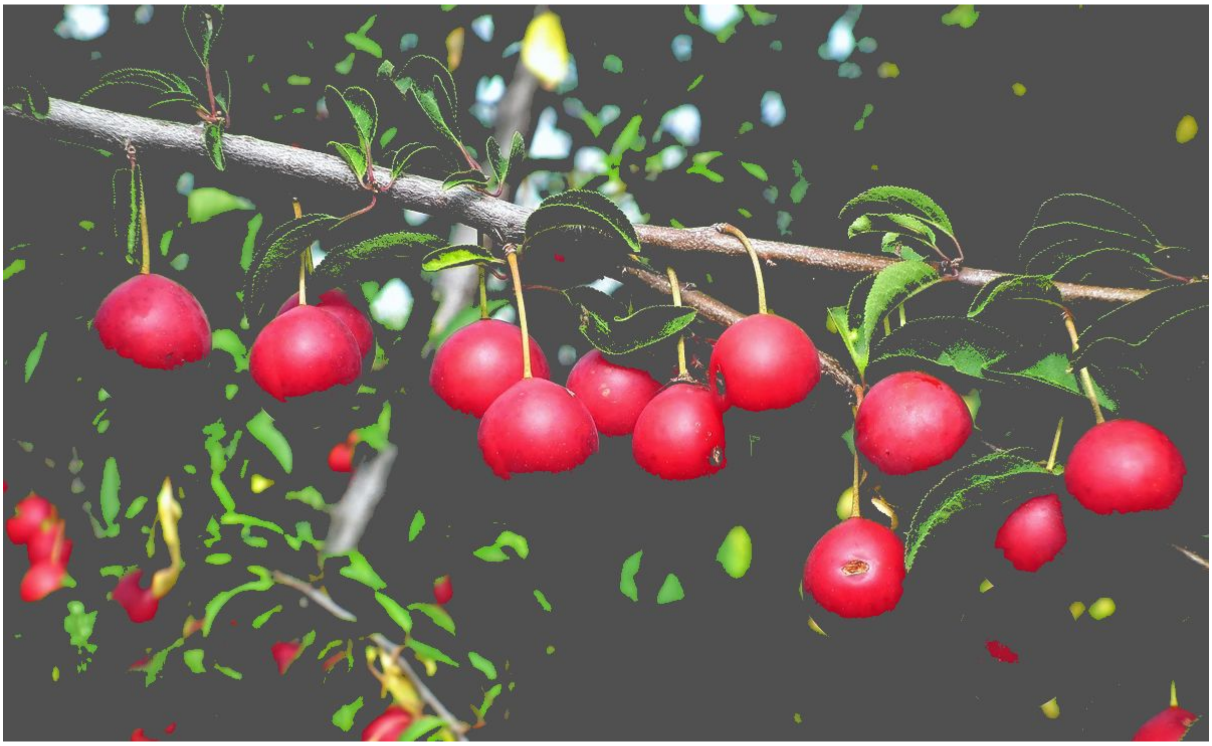




黃瑞得
0860078
huangjuite@gmail.com

T=5





黃瑞得
0860078
huangjuite@gmail.com

T=10