Image Processing

Project7 Report

電機碩乙 0850736 楊登宇

code

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
import os
import math

output_dir = os.path.join('output')
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

threshold = 10

def read_file():
    img1 = cv2.imread('image-pj7a.tif',cv2.IMREAD_COLOR)
    img1 = cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
    img2 = cv2.imread('image-pj7b.tif',cv2.IMREAD_COLOR)
    img2 = cv2.cvtColor(img2,cv2.COLOR_BGR2RGB)
    img3 = cv2.imread('image-pj7c.tif',cv2.IMREAD_COLOR)
    img3 = cv2.cvtColor(img3,cv2.COLOR_BGR2RGB)
    img4 = cv2.imread('image-pj7d.tif',cv2.IMREAD_COLOR)
    img4 = cv2.cvtColor(img4,cv2.COLOR_BGR2RGB)
    return img1, img2, img3, img4

def show_img(img,figname,gray=False):
    # plt.figure(figname)
    # plt.imshow(img)
    # plt.show()
    path = os.path.join(output_dir,figname+'.png')
    if not gray:
        cv_img = cv2.cvtColor(img,cv2.COLOR_RGB2BGR)
    else:
        cv_img = img
    cv2.imwrite(path,cv_img)
    return
```

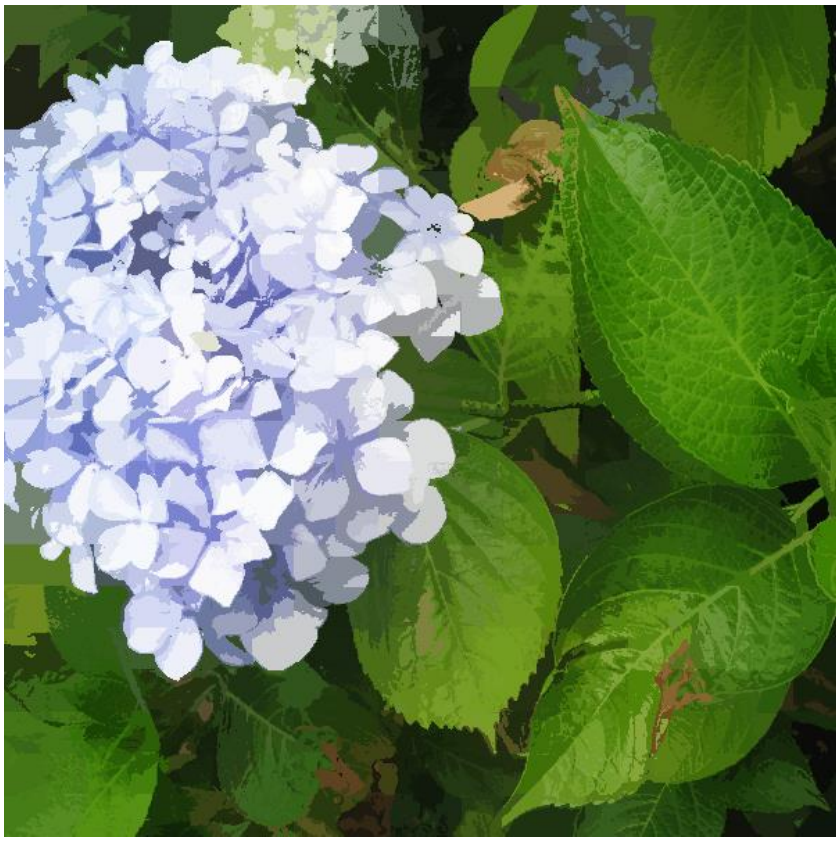
```
def gradient_pixel(img,c_x,c_y):
    gradient = 1000
    m_x = c_x
    m_y = c_y
    dir_list = [-1,1]
    for i in range(-1,2):
        for j in range(-1,2):
            if i == 0 and j == 0:
                continue
            a = c_x + i
            b = c_y + j
            try:
                x_diff = img[a-1,b] - img[a+1,b]
                y_diff = img[a,b-1] - img[a,b+1]
                diff_gradient = np.linalg.norm(x_diff)**2 + np.linalg.norm(y_diff)**2
                if diff_gradient < gradient:
                    gradient = diff_gradient
                    m_x = a
                    m_y = b
            except:
                continue
    return m_x, m_y

def distance_mean(mi,mj,c,s):
    dc = np.linalg.norm(mi[2:5] - mj[2:5])
    ds = np.linalg.norm(mi[:2] - mj[:2])

    D = (dc/c)**2 + (ds/s)**2
    D = D**0.5
    return D

def super_pixel(img, Ns, c, figname):
    Nt = img.shape[0]*img.shape[1]
    row = img.shape[0]
    col = img.shape[1]
    s = int(pow(Nt/Ns,0.5))
```

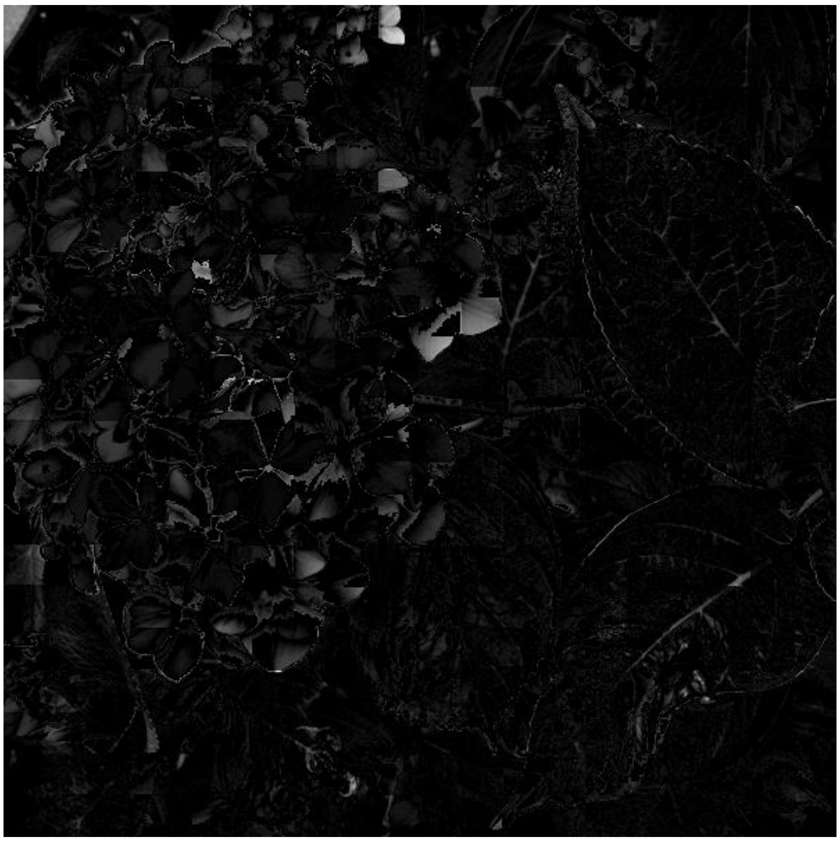
```
print("{}: Ns={}, c={}, s={}".format(figname,Ns,c,s))
m = []
distance_matrix = np.full((img.shape[0],img.shape[1]),10000)
mean_matrix = np.full((img.shape[0],img.shape[1]),-1)
# initial m
for i in range(int(row/s)):
    for j in range(int(col/s)):
        a = (i+1)*s - 1
        b = (j+1)*s - 1
        m_a, m_b = gradient_pixel(img,a,b)
        m.append([m_a,m_b,img[m_a,m_b,0],img[m_a,m_b,1],img[m_a,m_b,2]])
# print(m)
old_m = m.copy()
distance_E = 10000
it = 0
while distance_E >= threshold:
    it += 1
    for index, mi in enumerate(m):
        # xi = mi[0]
        # yi = mi[1]
        xi = old_m[index][0]
        yi = old_m[index][1]
        # center = np.array([xi,yi,img[xi,yi][0],img[xi,yi][1],img[xi,yi][2]])
        center = np.array(mi)
        for i in range(-s,s+1):
            for j in range(-s,s+1):
                a = xi + i
                b = yi + j
                try:
                    pixel = np.array([a,b,img[a,b][0],img[a,b][1],img[a,b][2]])
                    D = distance_mean(center,pixel,c,s)
                    if D < distance_matrix[a,b]:
                        distance_matrix[a,b] = D
                        mean_matrix[a,b] = index
                except:
                    continue
    new_mean = []
    mean_sum = np.zeros((len(m),5))
    mean_count = np.zeros((len(m)))
    for i in range(mean_matrix.shape[0]):
      for j in range(mean_matrix.shape[1]):
        mean_sum[mean_matrix[i,j]] += np.array([i,j,img[i,j,0],img[i,j,1],img[i,j,2]])
        mean_count[mean_matrix[i,j]] += 1
    distance_E = 0
    for i in range(len(m)):
      n_m = mean_sum[i] / mean_count[i]
      new_mean.append([int(n_m[0]),int(n_m[1]),int(n_m[2]),int(n_m[3]),int(n_m[4])])
      distance_E += np.linalg.norm(np.array(m[i])-np.array(new_mean[i]))
    print("iteration {}: E = {}".format(it,distance_E))
    m = new_mean.copy()
  generate_superpixel_img(img,m,mean_matrix,figname+"_superpixel_"+str(Ns)+"_c_
"+str(c))
  return

def generate_superpixel_img(img,m,mean_matrix,figname):
  super_img = img.copy()
  mean_pixel = np.zeros((len(m),3))
  for i in range(len(m)):
    mean_pixel[i] = m[i][2],m[i][3],m[i][4]

  for i in range(img.shape[0]):
    for j in range(img.shape[1]):
      super_img[i,j,0] = int(mean_pixel[int(mean_matrix[i,j])][0])
      super_img[i,j,1] = int(mean_pixel[int(mean_matrix[i,j])][1])
      super_img[i,j,2] = int(mean_pixel[int(mean_matrix[i,j])][2])
  # print(super_img[0,0])
  show_img(super_img,figname)



def diff_img(img1,img2,img3,img4):
  c_list = [1,10]
  Ns_list = [100,400]
  img_list = [1,2,3,4]
  img = [img1, img2, img3, img4]
```

```
    for i in img_list:
        if i == 1:
            orig_img = cv2.cvtColor(img1,cv2.COLOR_RGB2GRAY)
        elif i == 2:
            orig_img = cv2.cvtColor(img2,cv2.COLOR_RGB2GRAY)
        elif i == 3:
            orig_img = cv2.cvtColor(img3,cv2.COLOR_RGB2GRAY)
        else:
            orig_img = cv2.cvtColor(img4,cv2.COLOR_RGB2GRAY)
        for c in c_list:
            for Ns in Ns_list:
                img = cv2.imread(os.path.join(output_dir,'image_'+str(i)+'_superpixel_'+str(Ns)
+'_c_'+str(c)+'.png'),cv2.IMREAD_COLOR)
                img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
                diff = cv2.subtract(orig_img,img)
                # print("diff max:{},min:{}".format(np.max(diff),np.min(diff)))
                diff = diff - np.min(diff)
                diff = diff / np.max(diff) * 255
                show_img(diff,'image_'+str(i)+'_superpixel_'+str(Ns)+'_c_'+str(c)+'_diff',gray=Tr
ue)
    return

if __name__ == "__main__":
    img1, img2, img3, img4 = read_file()
    show_img(img1,"Original_1")
    show_img(img2,"Original_2")

    super_pixel(img1,400,1, "image_1")
    super_pixel(img1,400,10, "image_1")
    super_pixel(img1,100,1, "image_1")
    super_pixel(img1,100,10, "image_1")

    super_pixel(img2,400,1, "image_2")
    super_pixel(img2,400,10, "image_2")
    super_pixel(img2,100,1, "image_2")
    super_pixel(img2,100,10, "image_2")
super_pixel(img3,400,1, "image_3")
super_pixel(img3,400,10, "image_3")
super_pixel(img3,100,1, "image_3")
super_pixel(img3,100,10, "image_3")

super_pixel(img4,400,1, "image_4")
super_pixel(img4,400,10, "image_4")
super_pixel(img4,100,1, "image_4")
super_pixel(img4,100,10, "image_4")

diff_img(img1,img2,img3,img4)
```

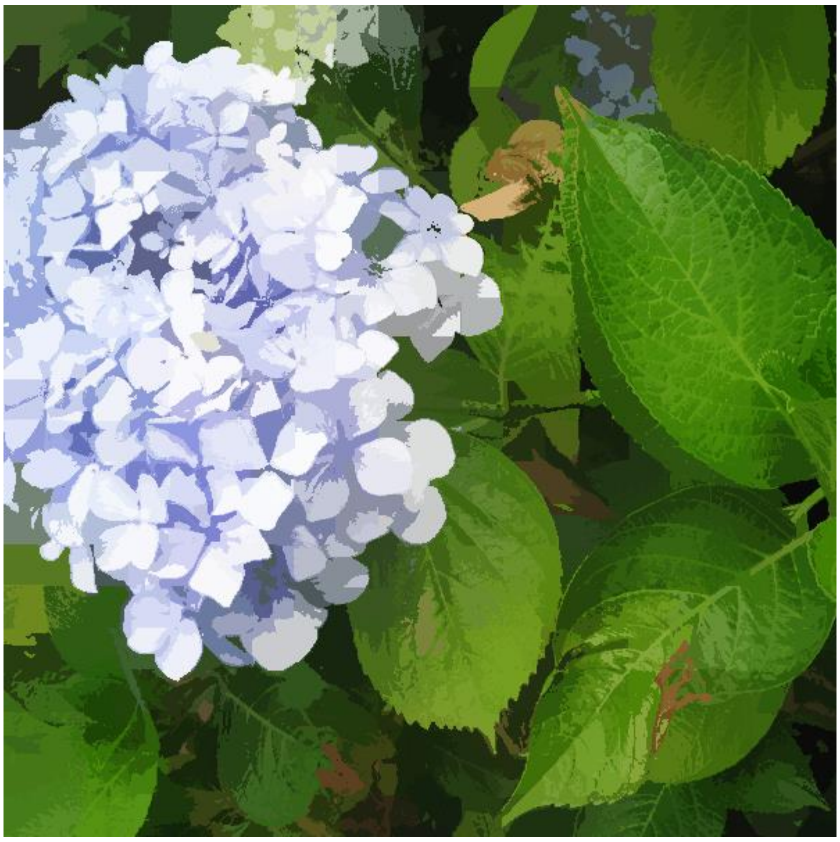
image a

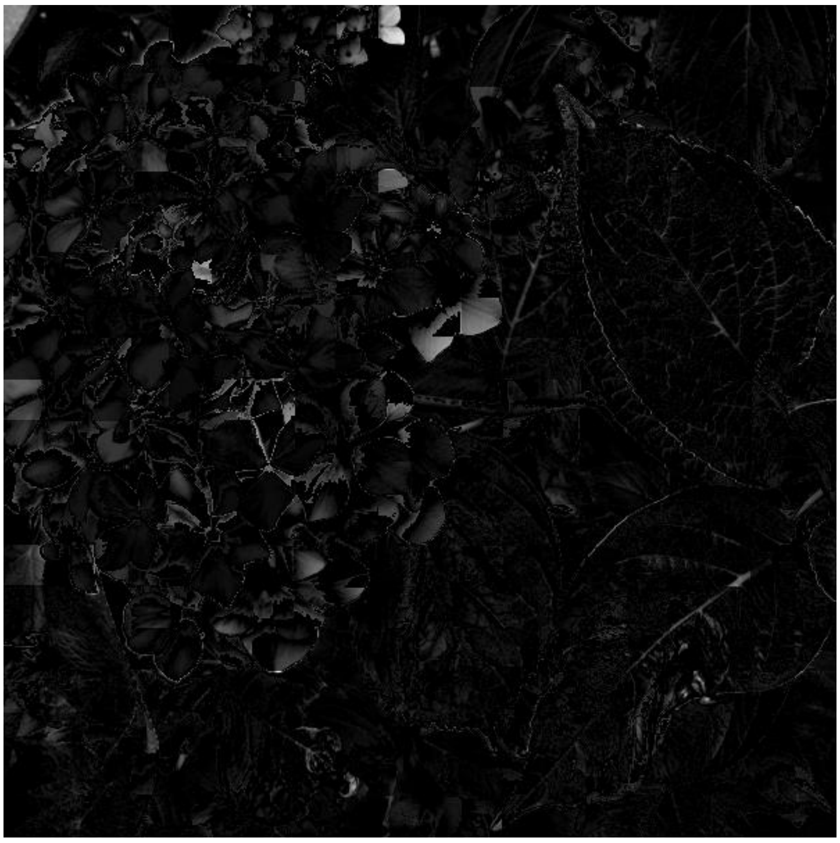
1)400 superpixel and difference images for c =1



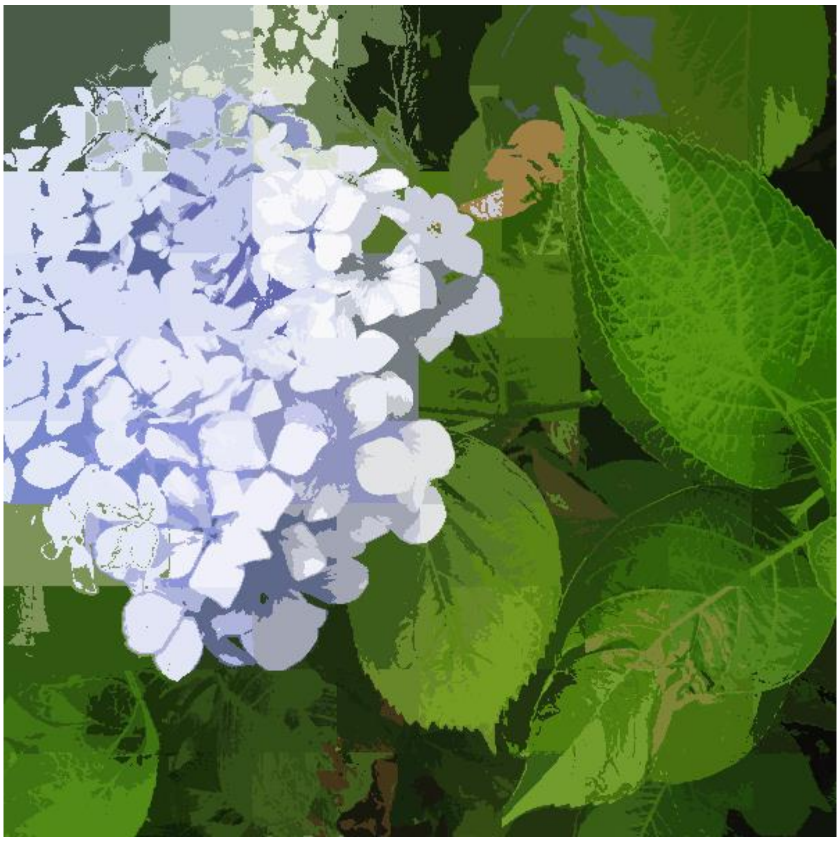


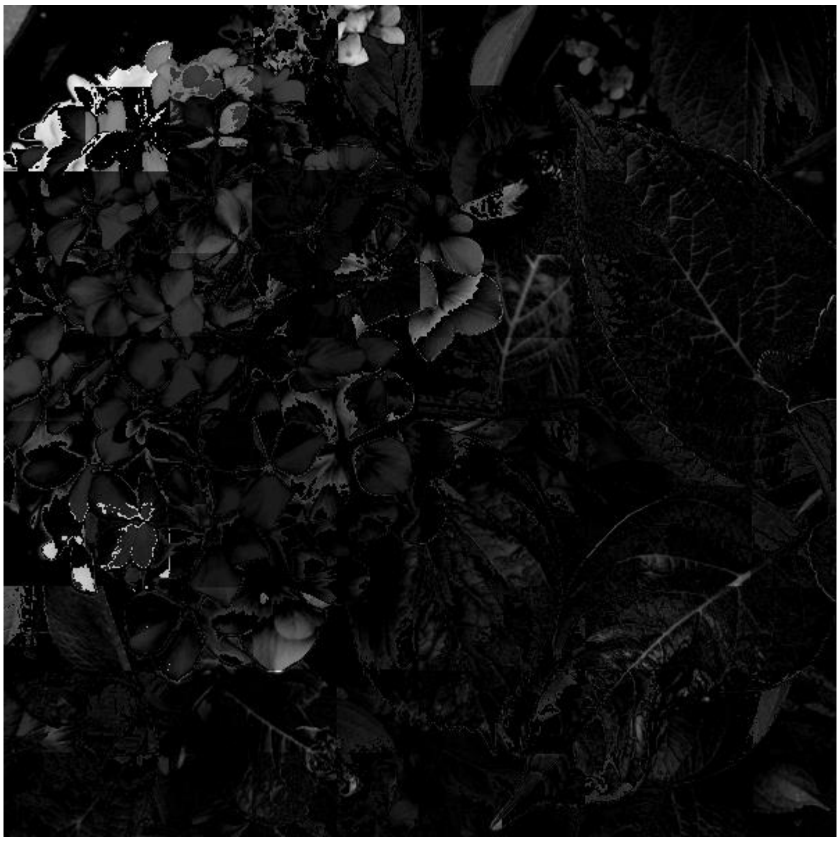
2)400 superpixel and difference images for c =10



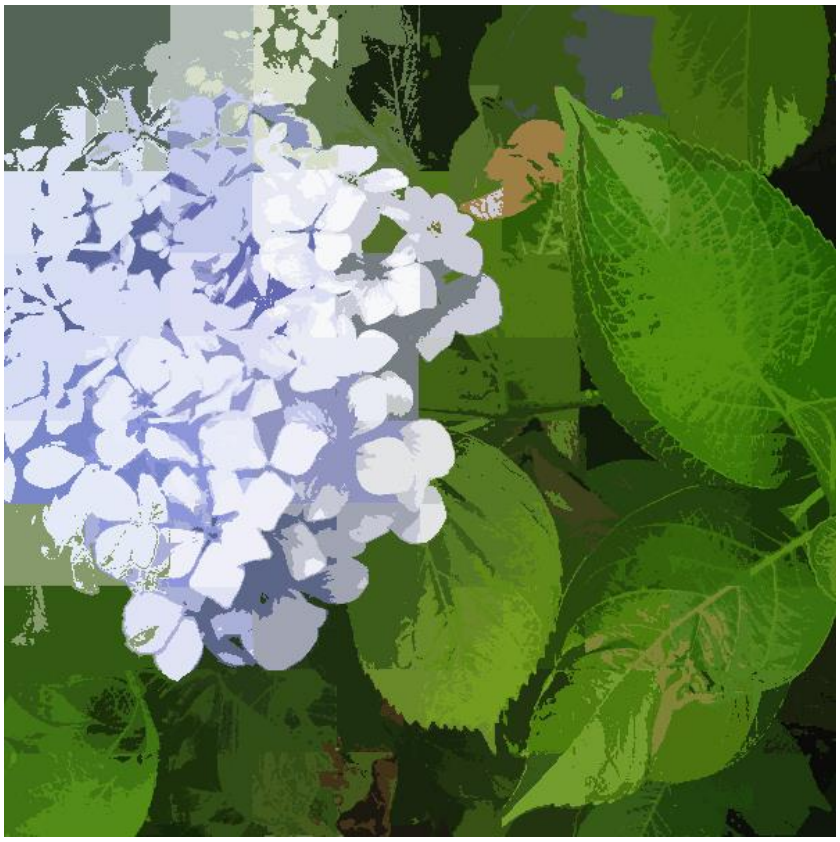


3)100 superpixel and difference images for c =1





4)100 superpixel and difference images for c =10



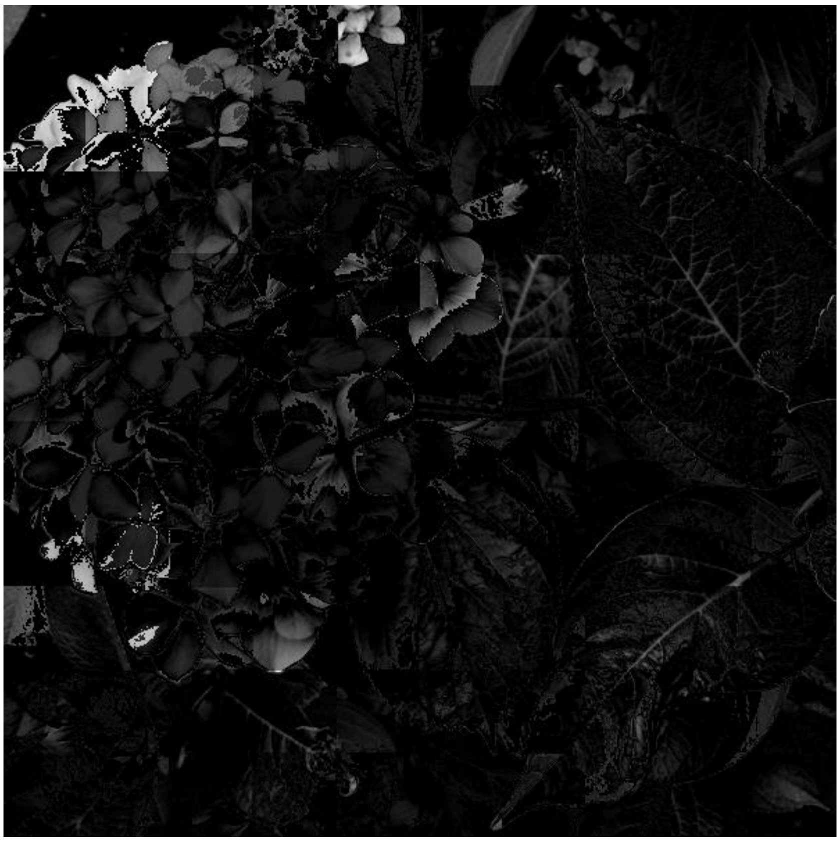
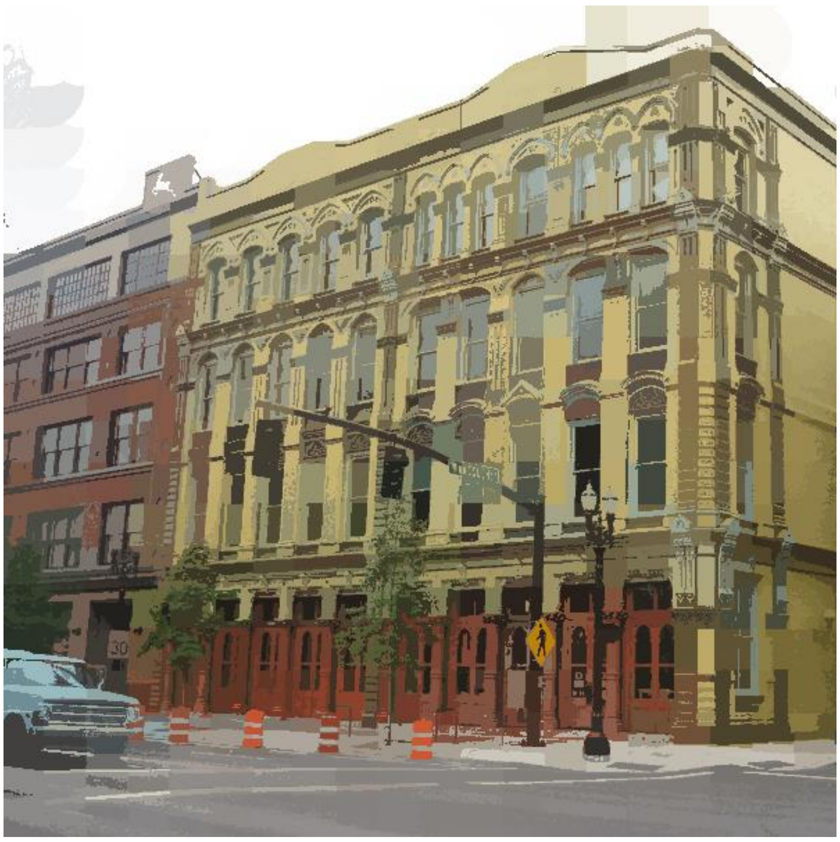
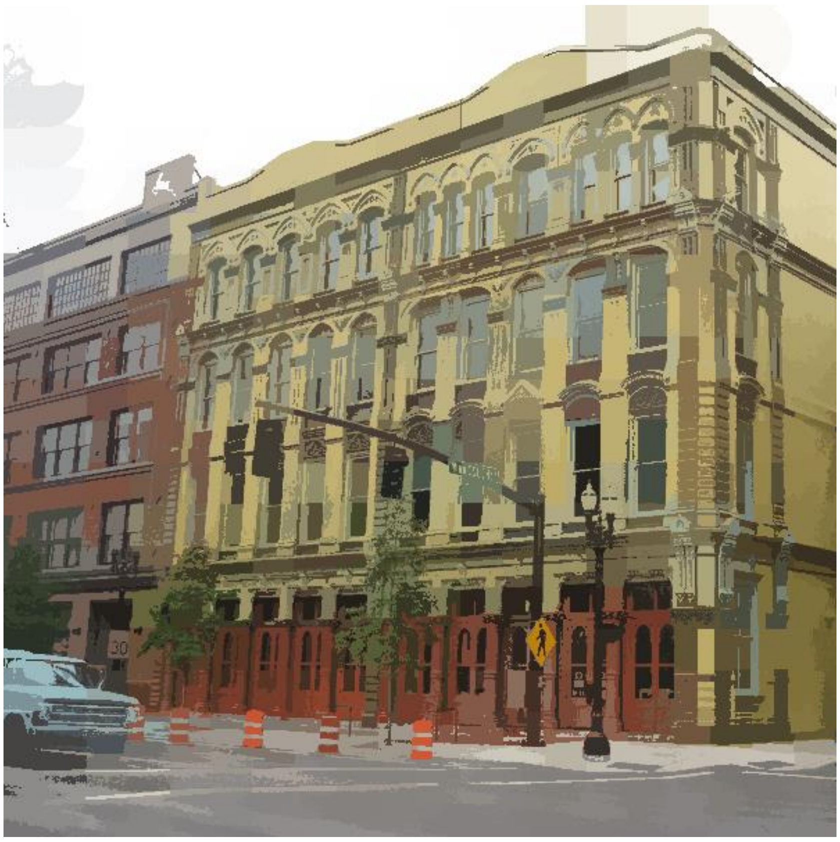


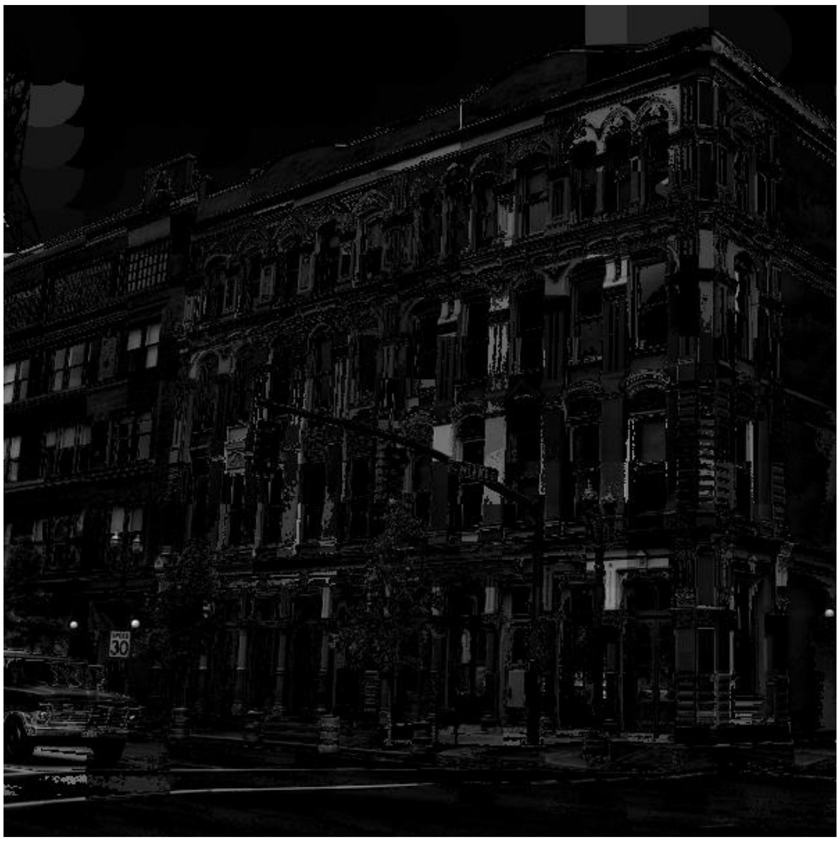
image b

1)400 superpixel and difference images for c =1

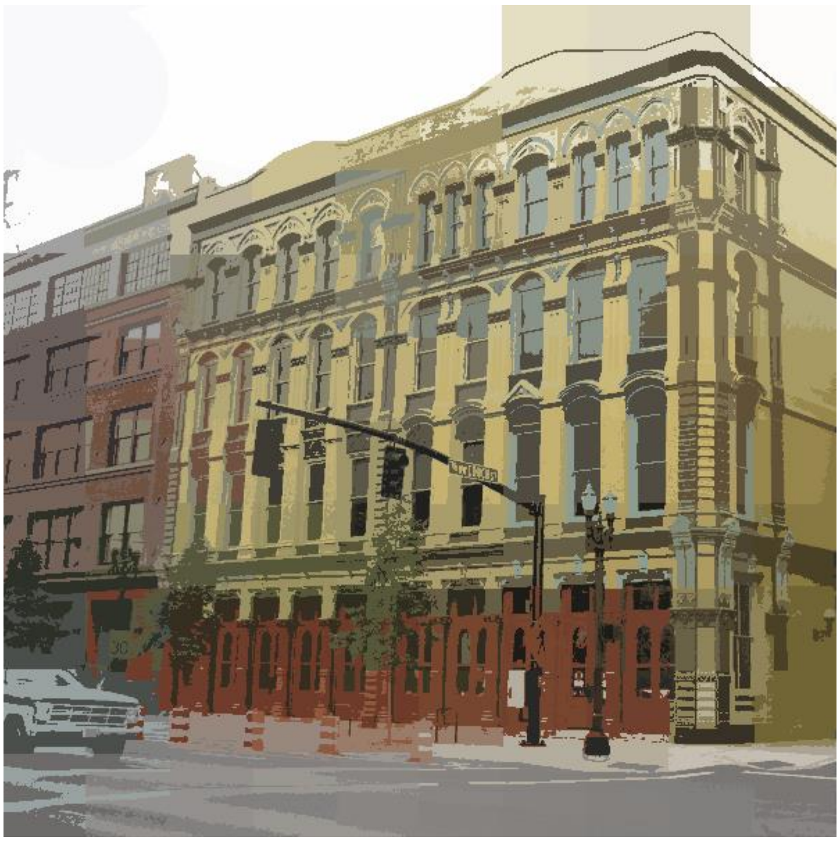


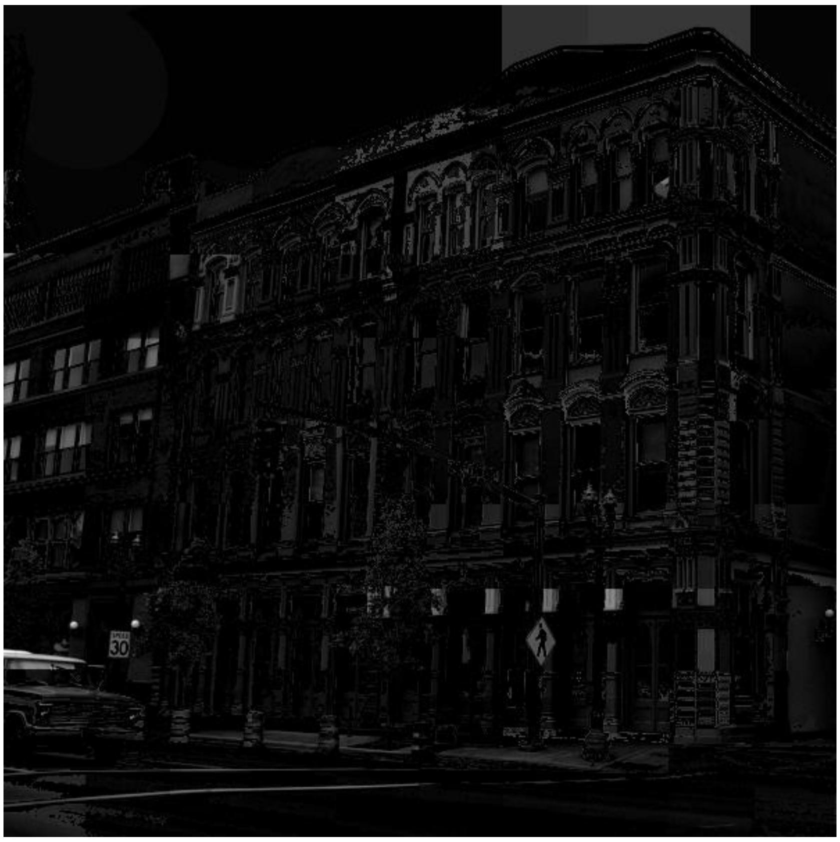
2)400 superpixel and difference images for c =10





3)100 superpixel and difference images for c =1





4)100 superpixel and difference images for c =10

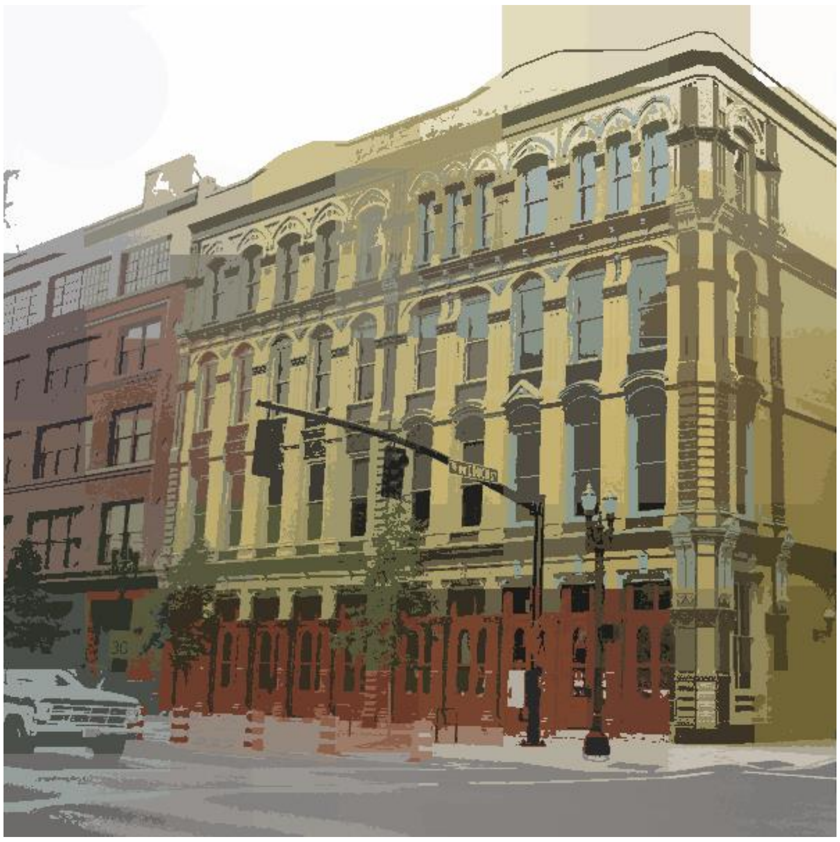
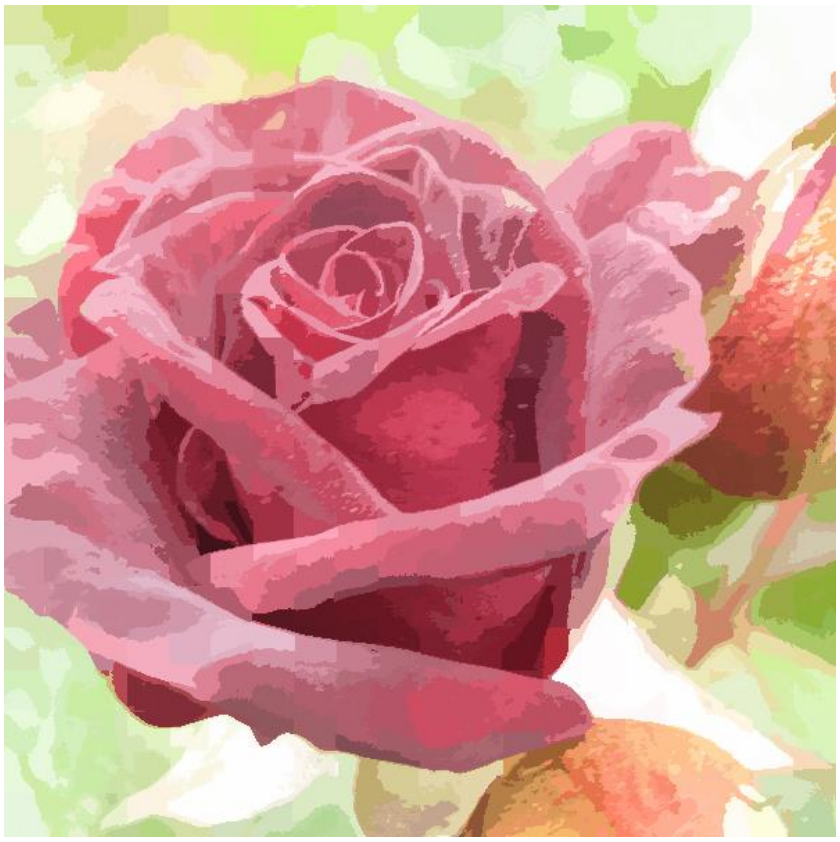
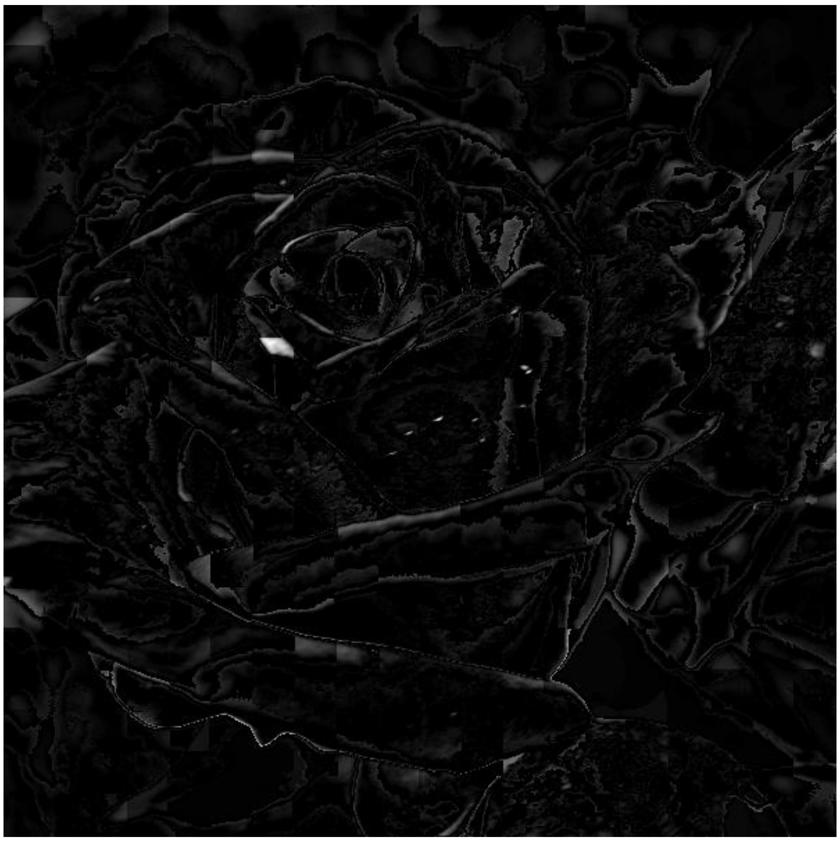


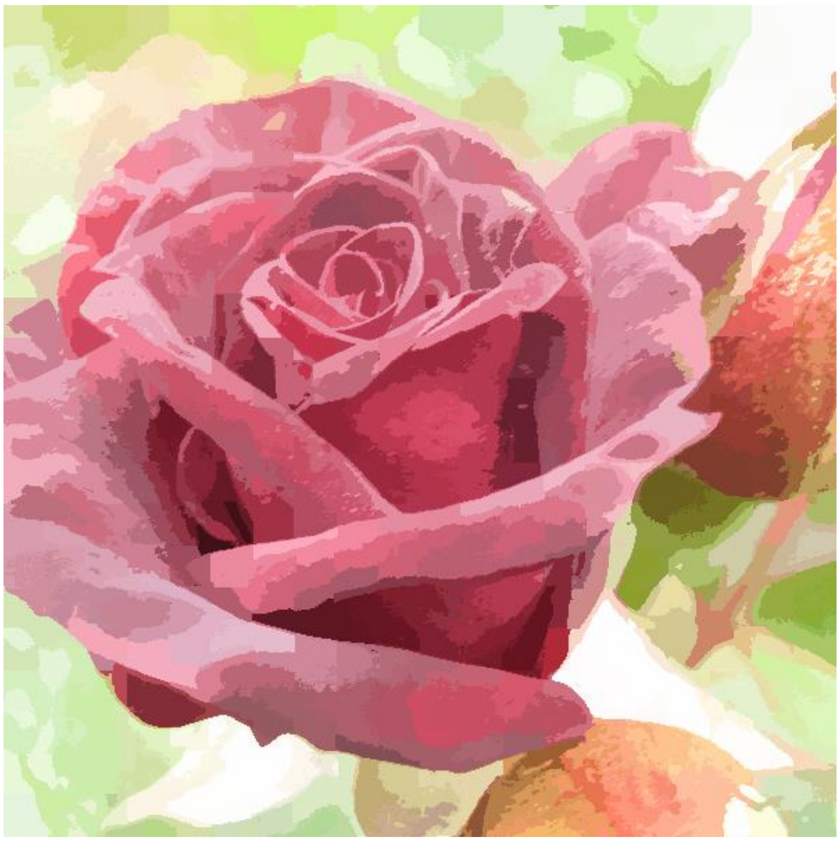
image c

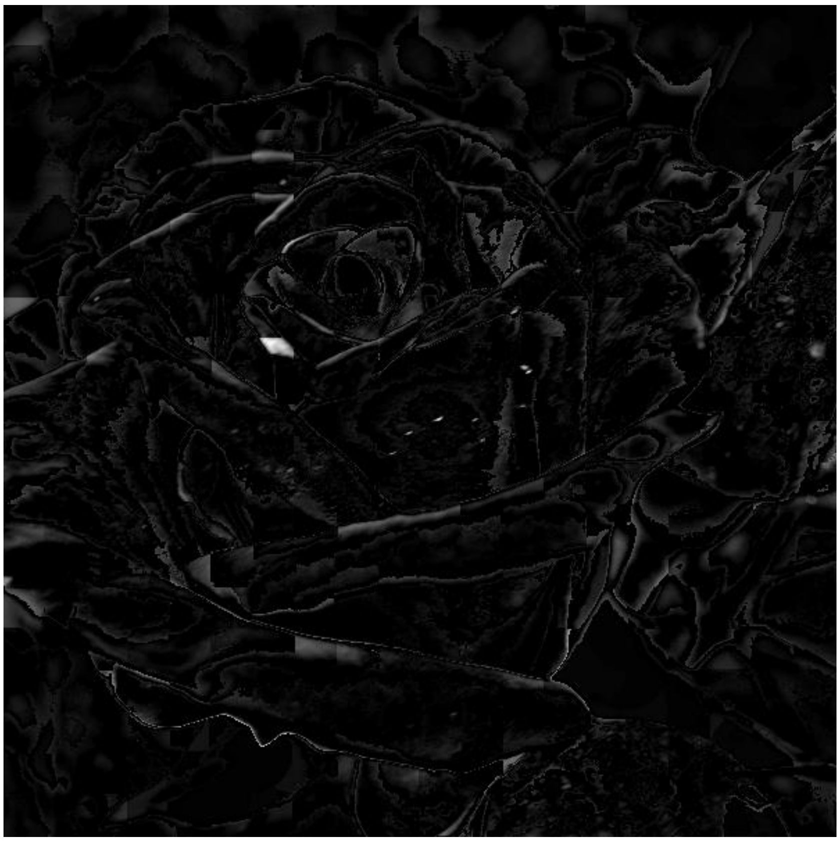
1)400 superpixel and difference images for c =1



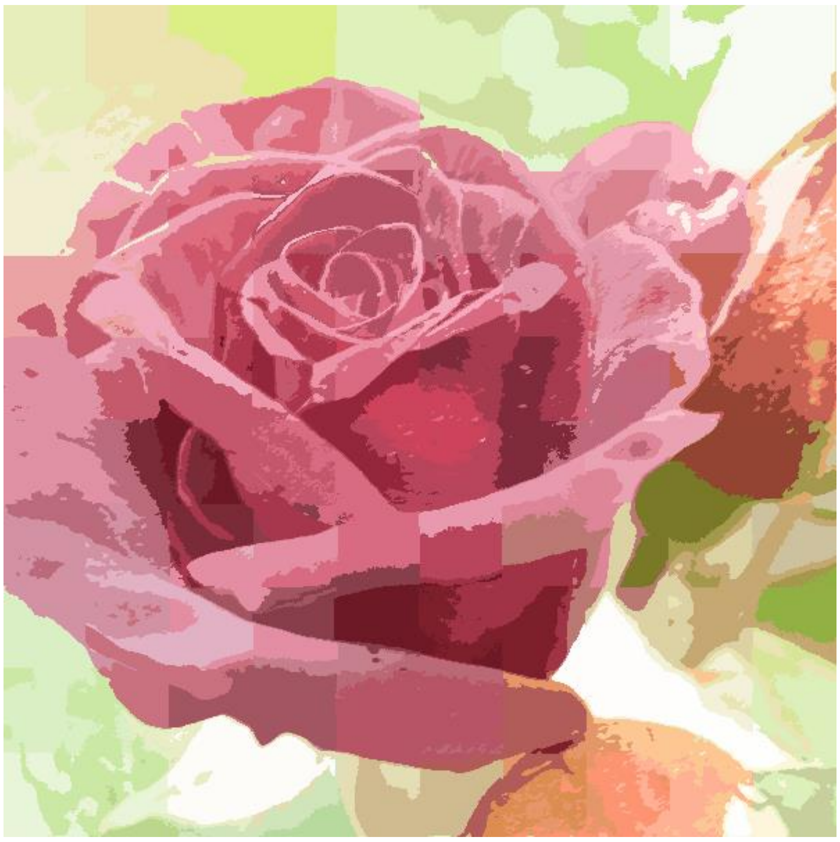


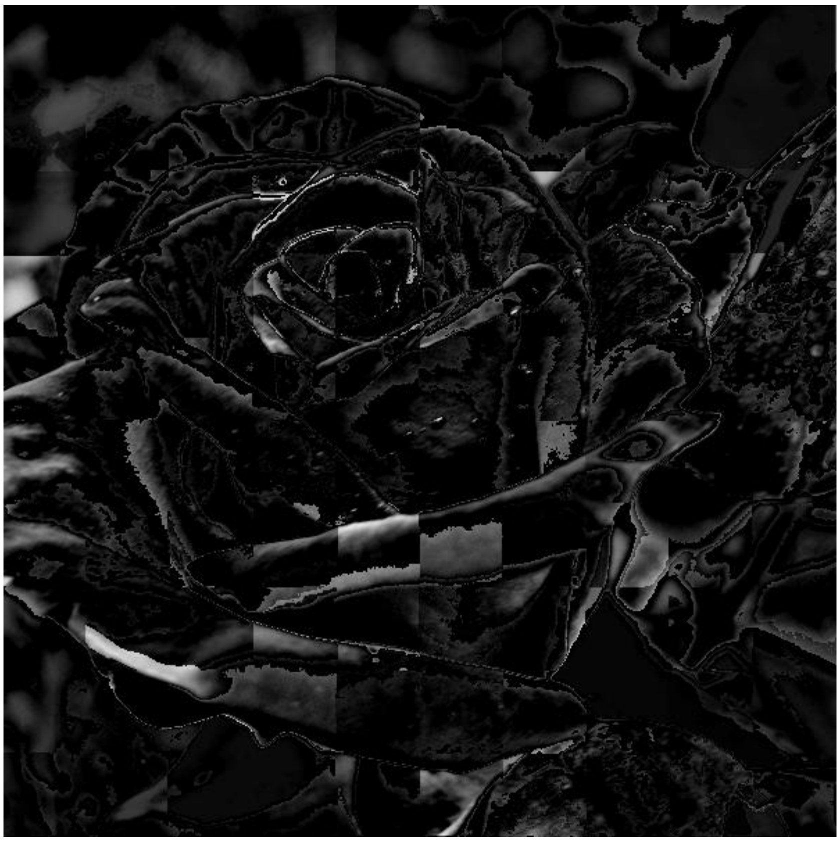
2)400 superpixel and difference images for c =10



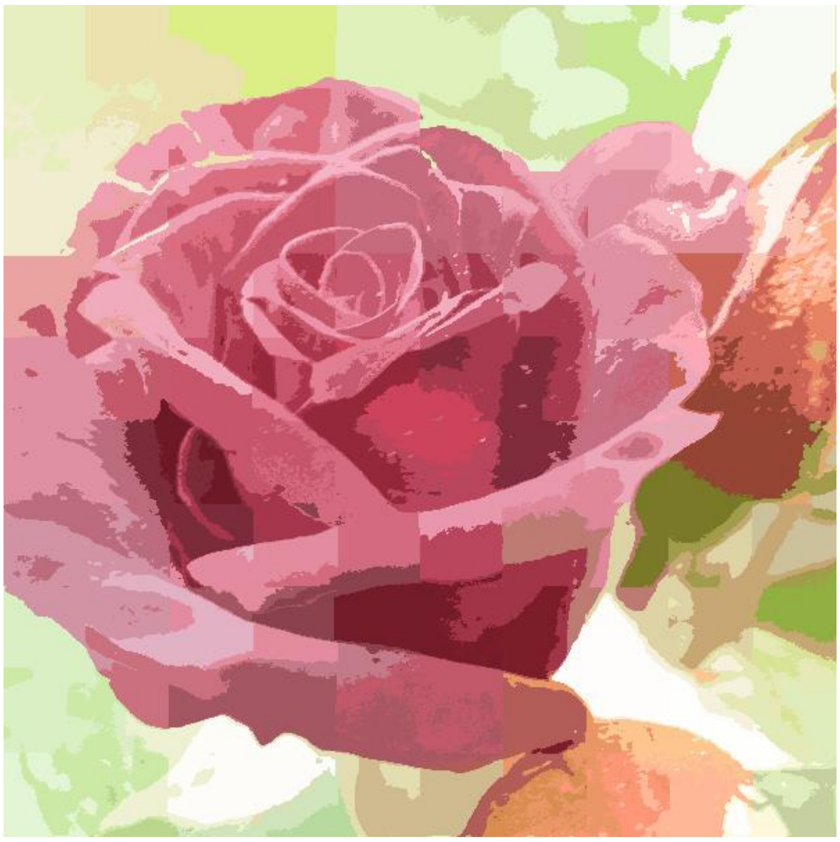


3)100 superpixel and difference images for c =1





4)100 superpixel and difference images for c =10



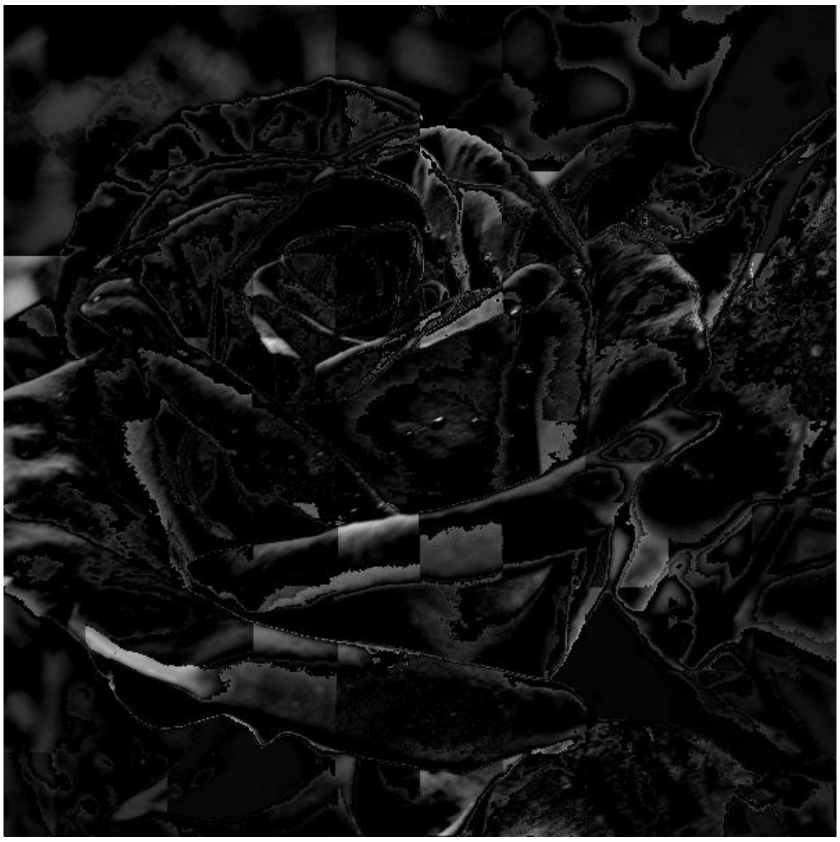
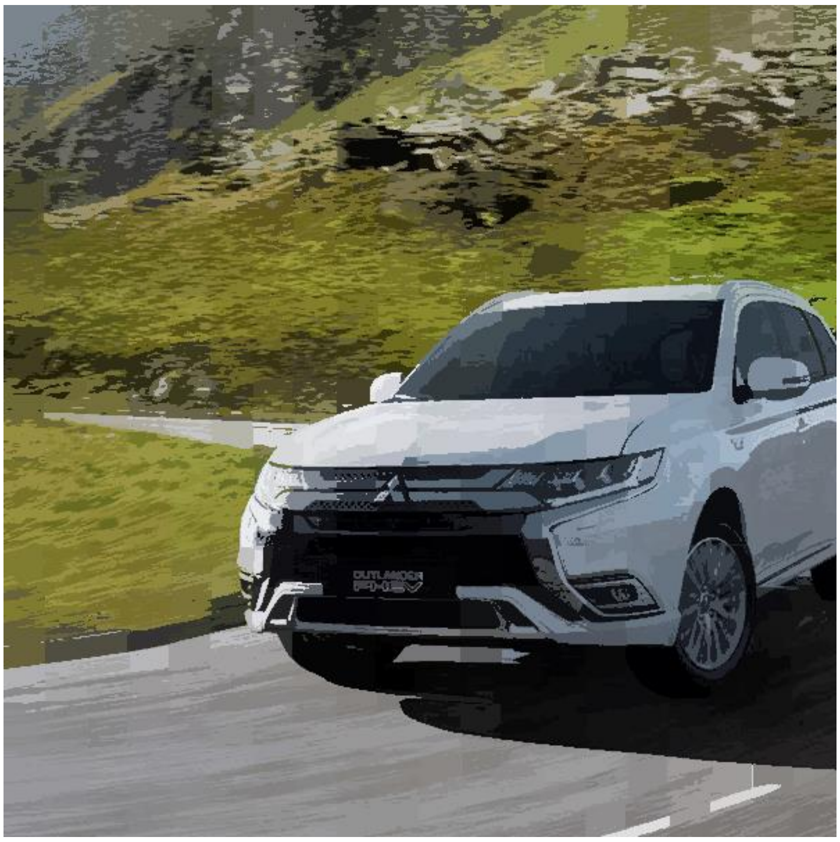
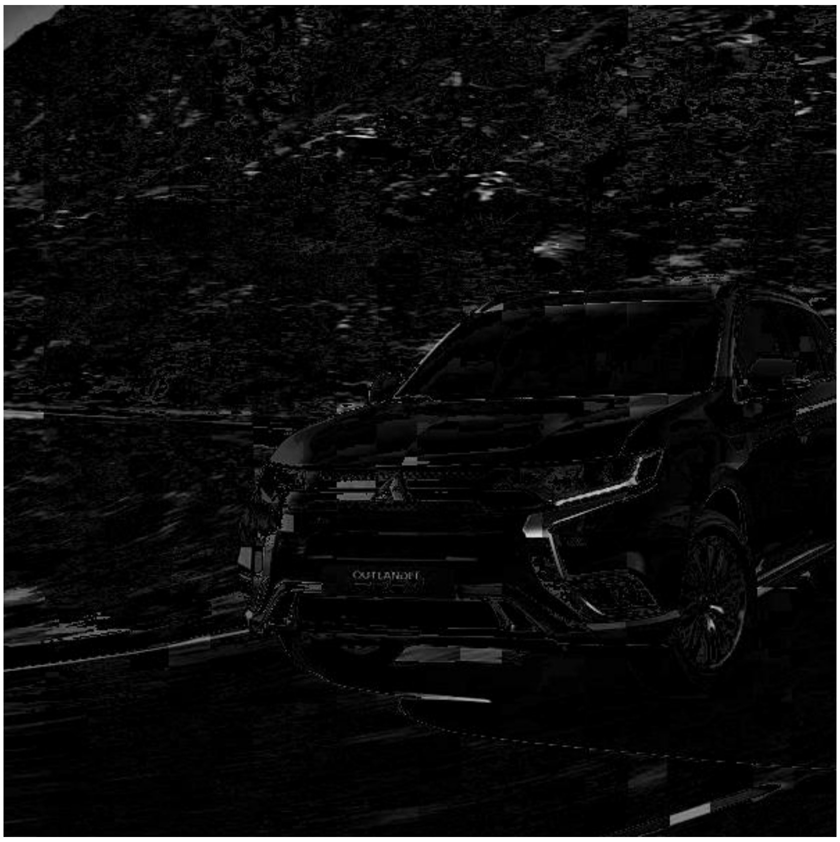


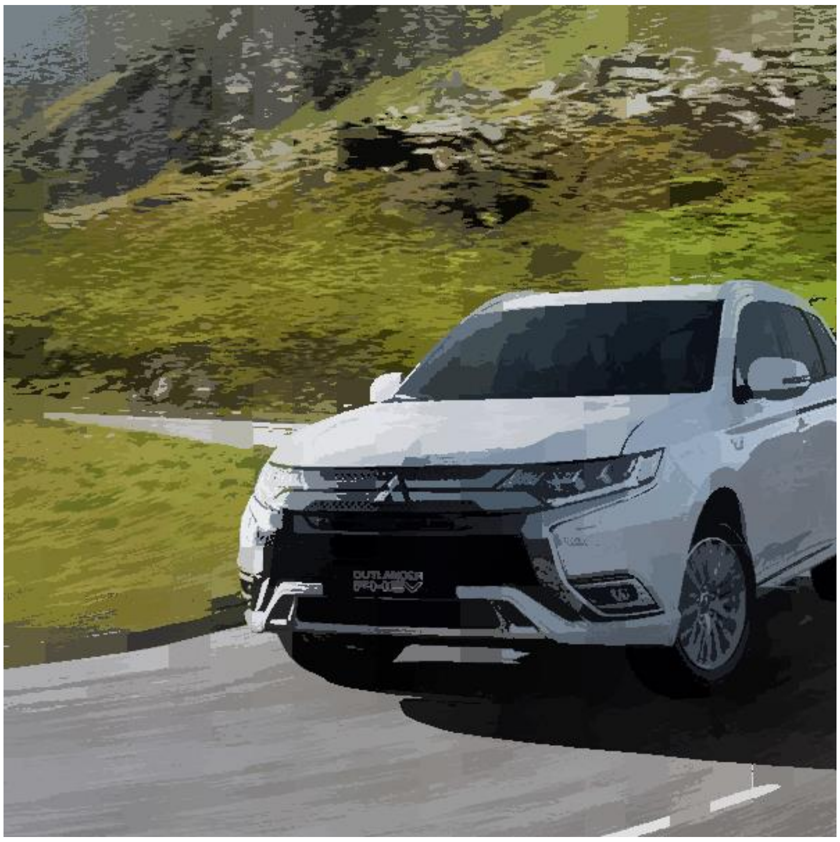
image d

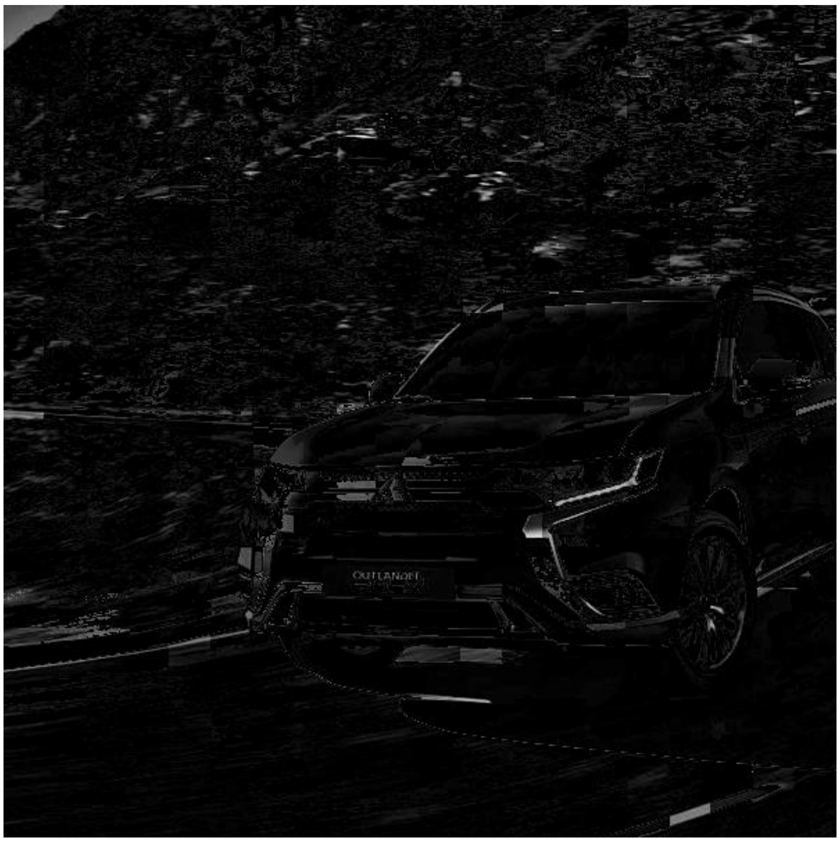
1)400 superpixel and difference images for c =1



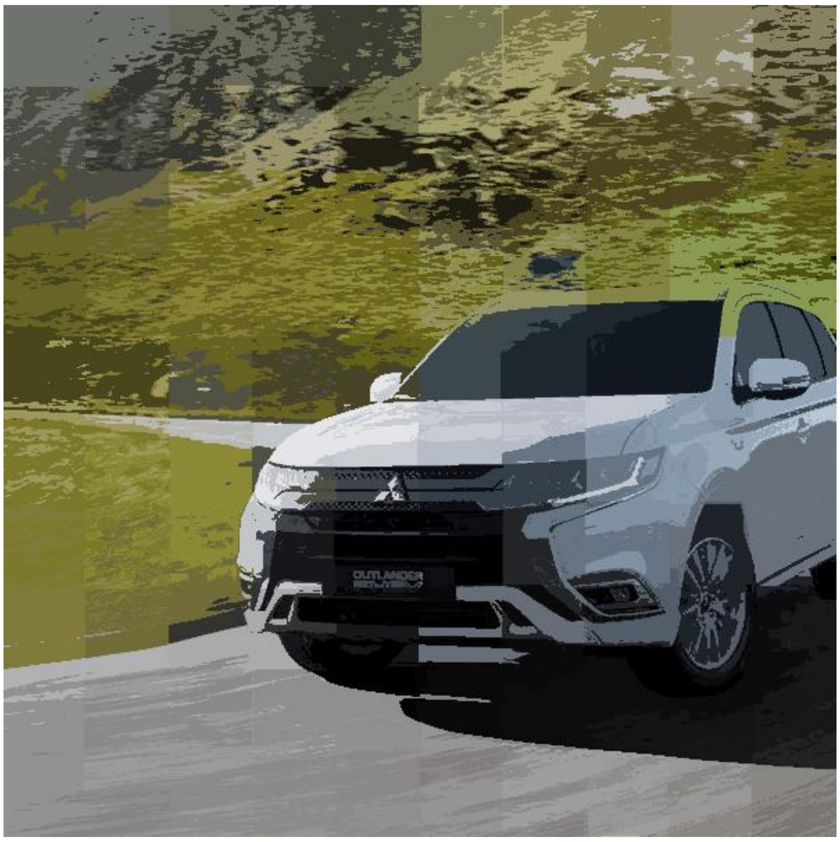


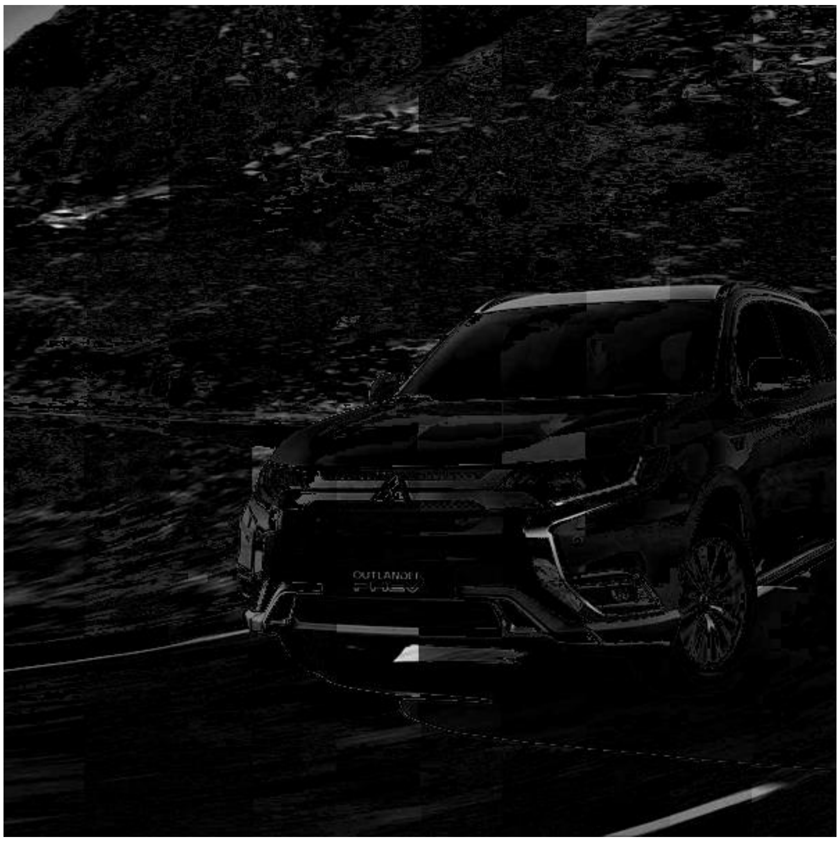
2)400 superpixel and difference images for c =10





3)100 superpixel and difference images for c =1





4)100 superpixel and difference images for c =10