影像處理導論HW7

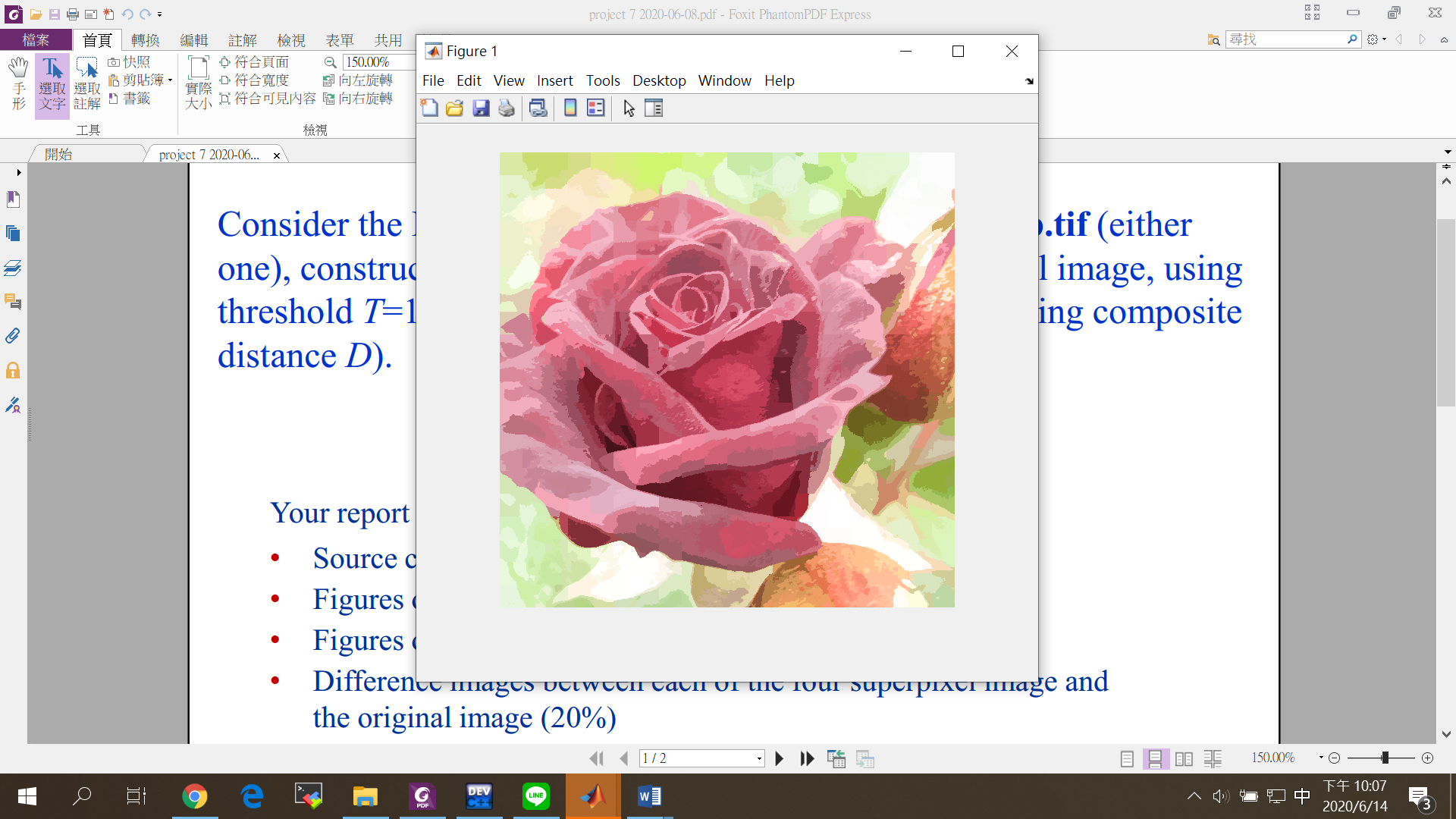
宋其諭0510888

Project goal

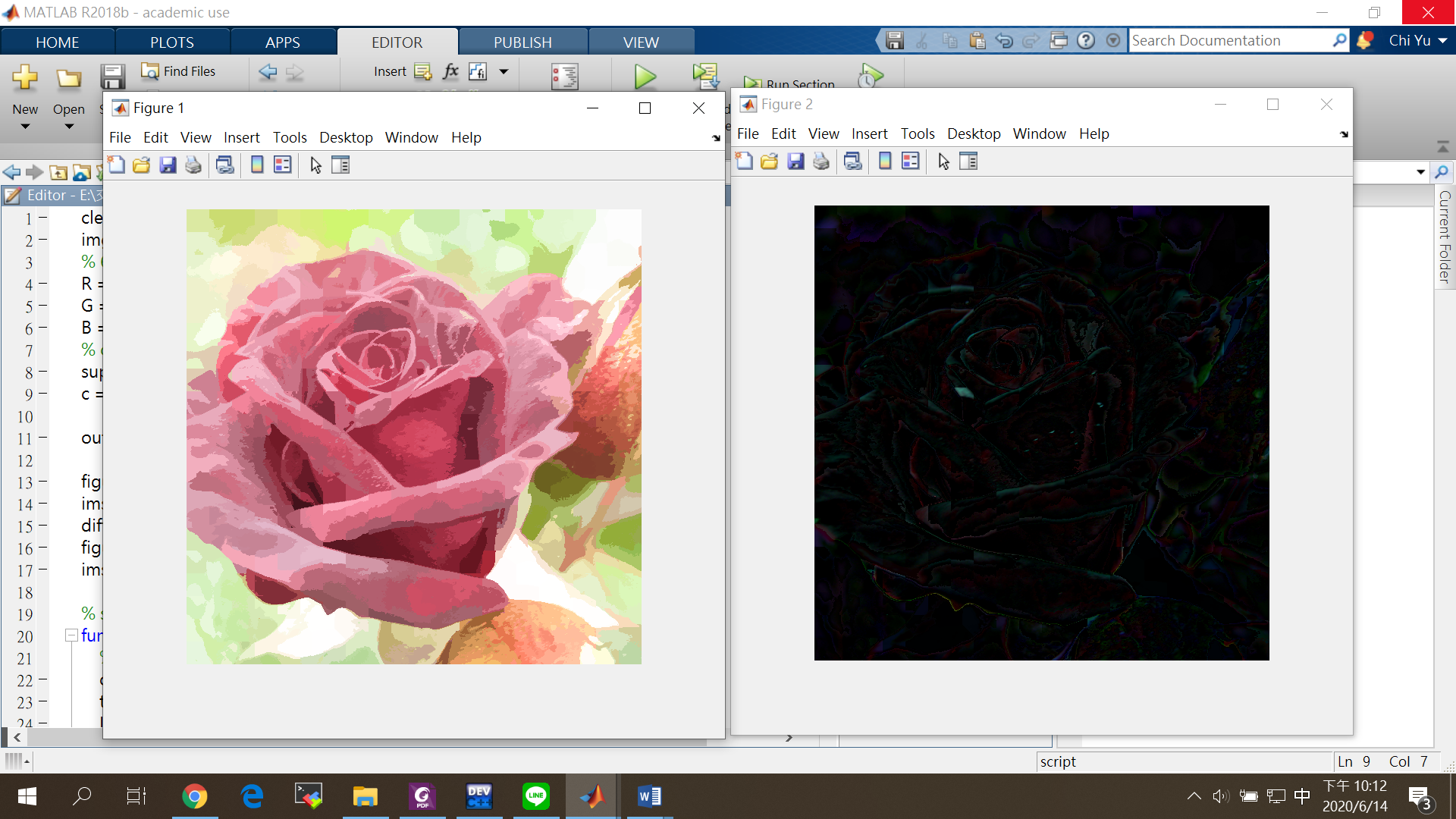
Consider the RBG image**, image-pj7a.tif** or **image-pj7b.tif** (either  
one), construct 400-superpixel image and 100-superpixel image, using  
threshold *T*=10 and *c* = 1 and 10 (*c*: constant for computing composite  
distance *D*).

1. Figures of 400-superpixel images for *c*=1 and *c*=10 (30%)

C=1,400 superpixel

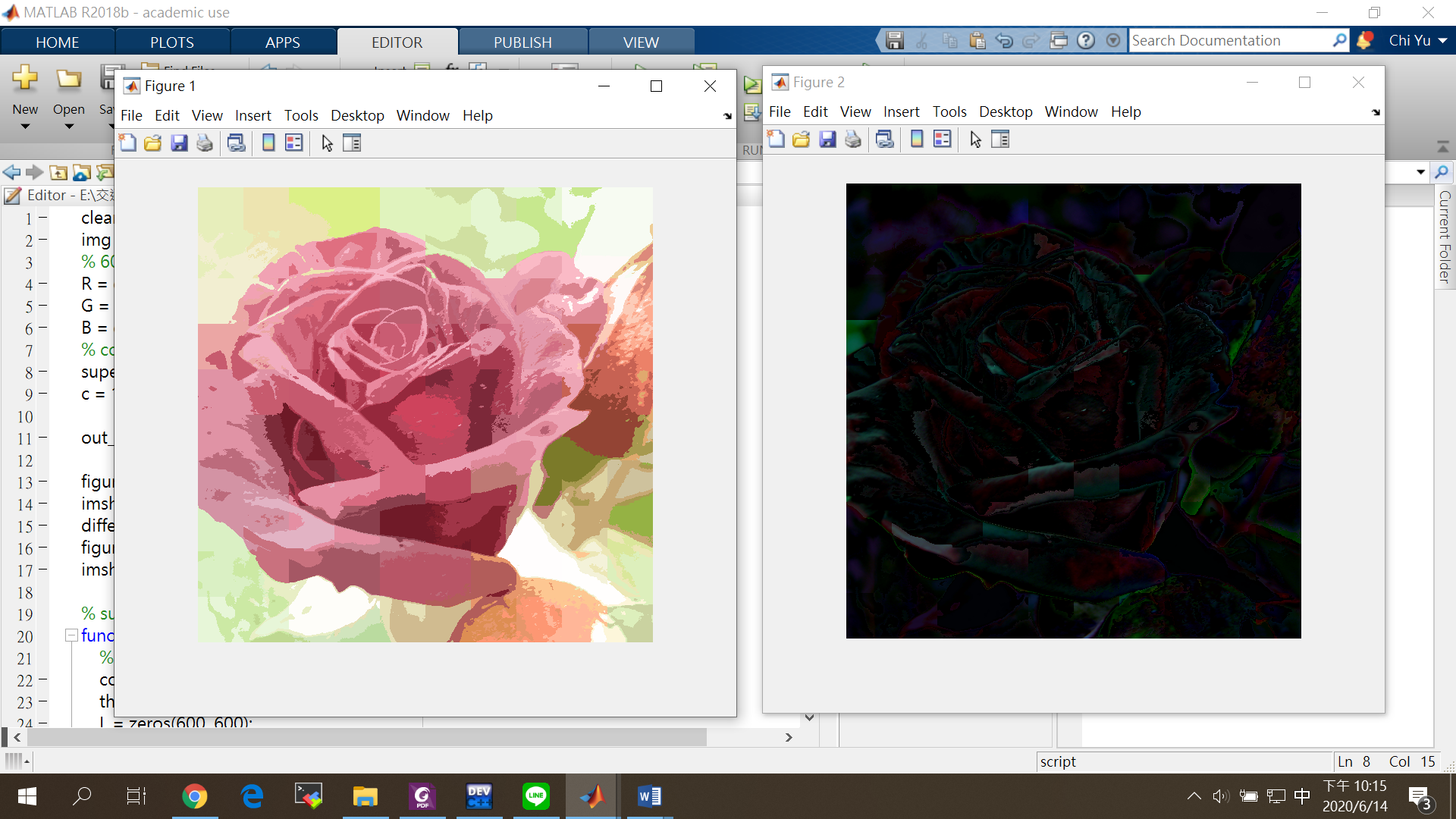


C=10,400 superpixel



1. Figures of 100-superpixel images for *c*=1 and *c*=10 (20%)

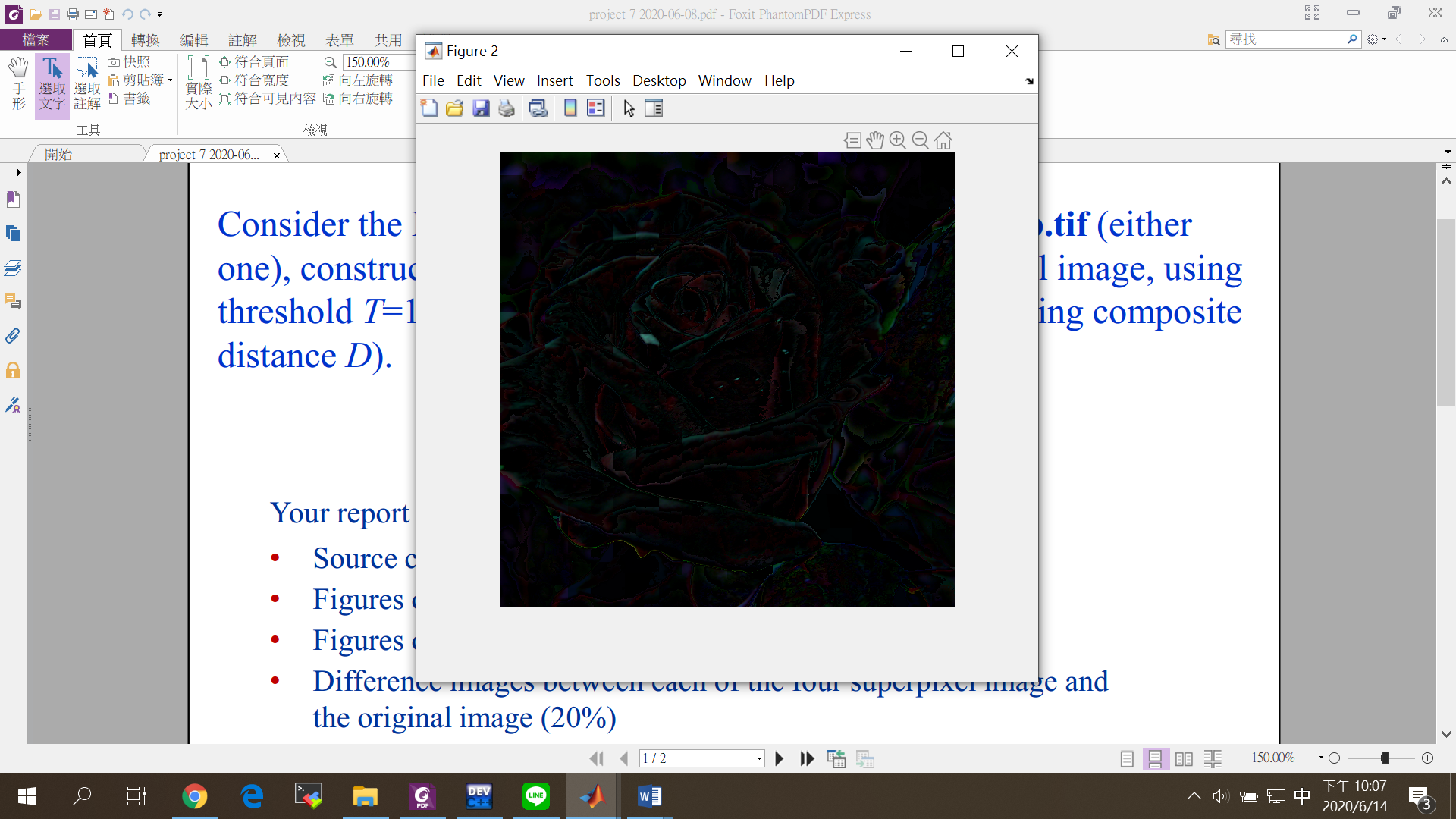
C=1,100 superpixel



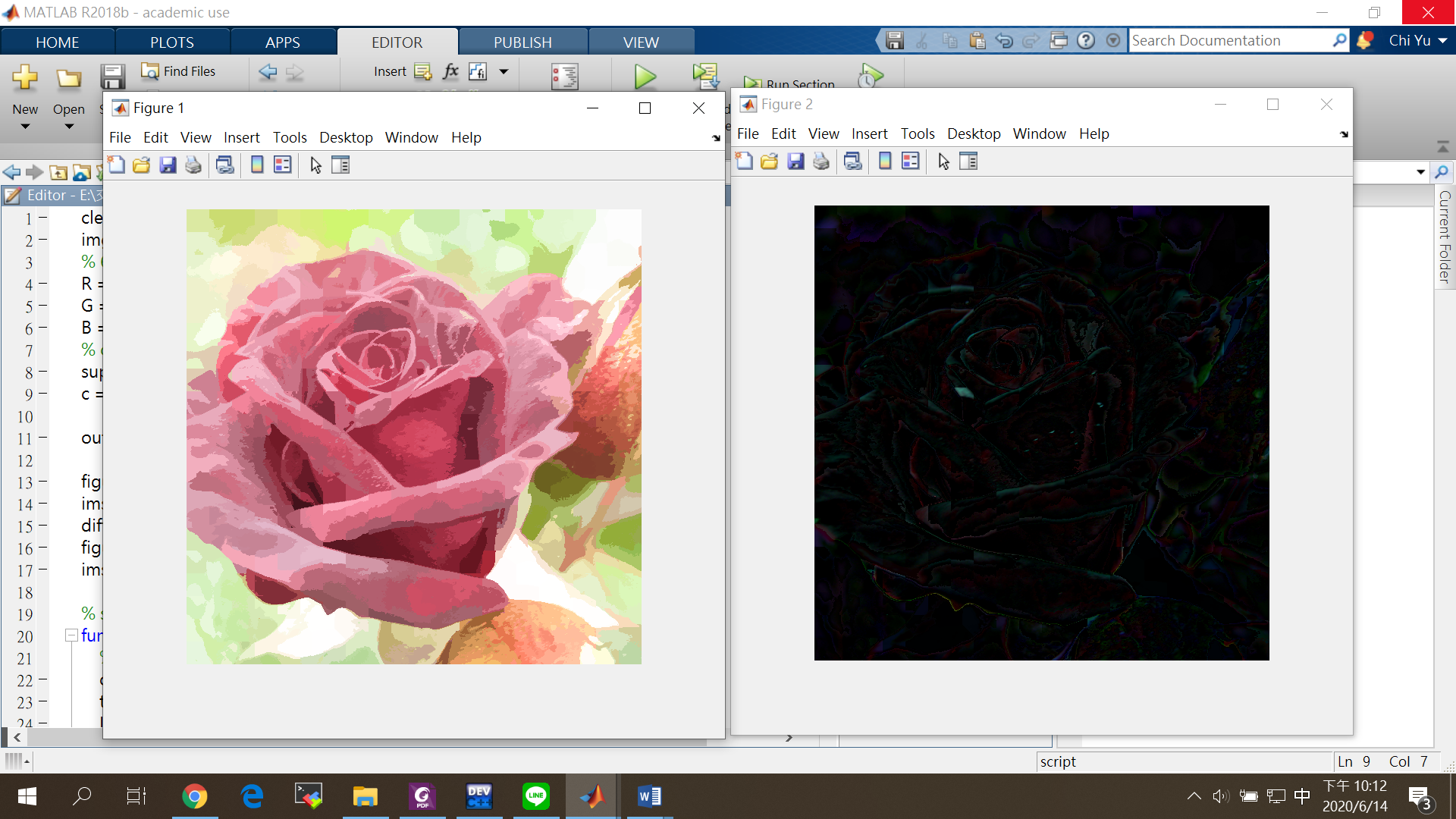
C=10,100 superpixel

1. Difference images between each of the four superpixel image and the original image (20%)

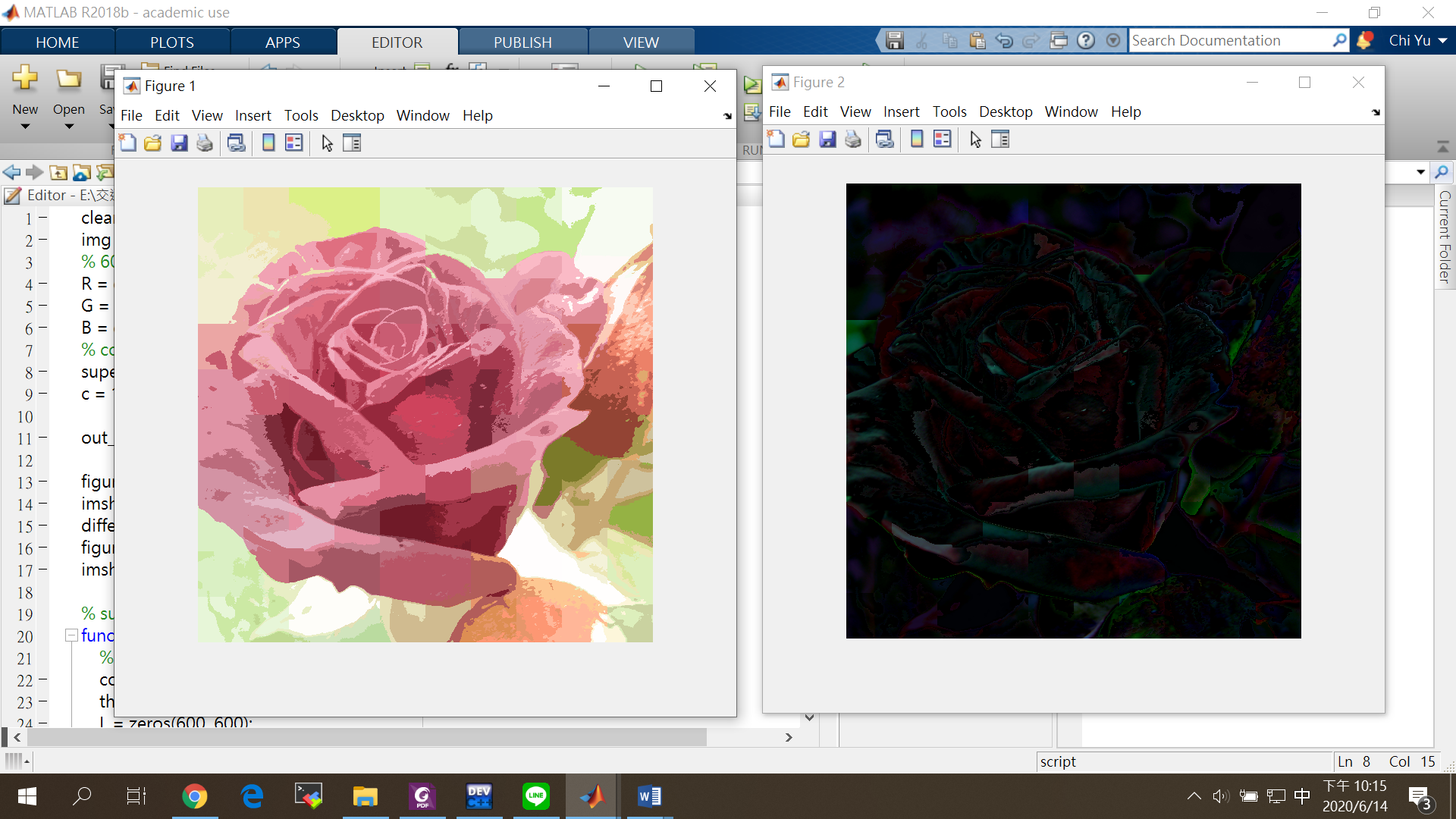
C=1,400 superpixel



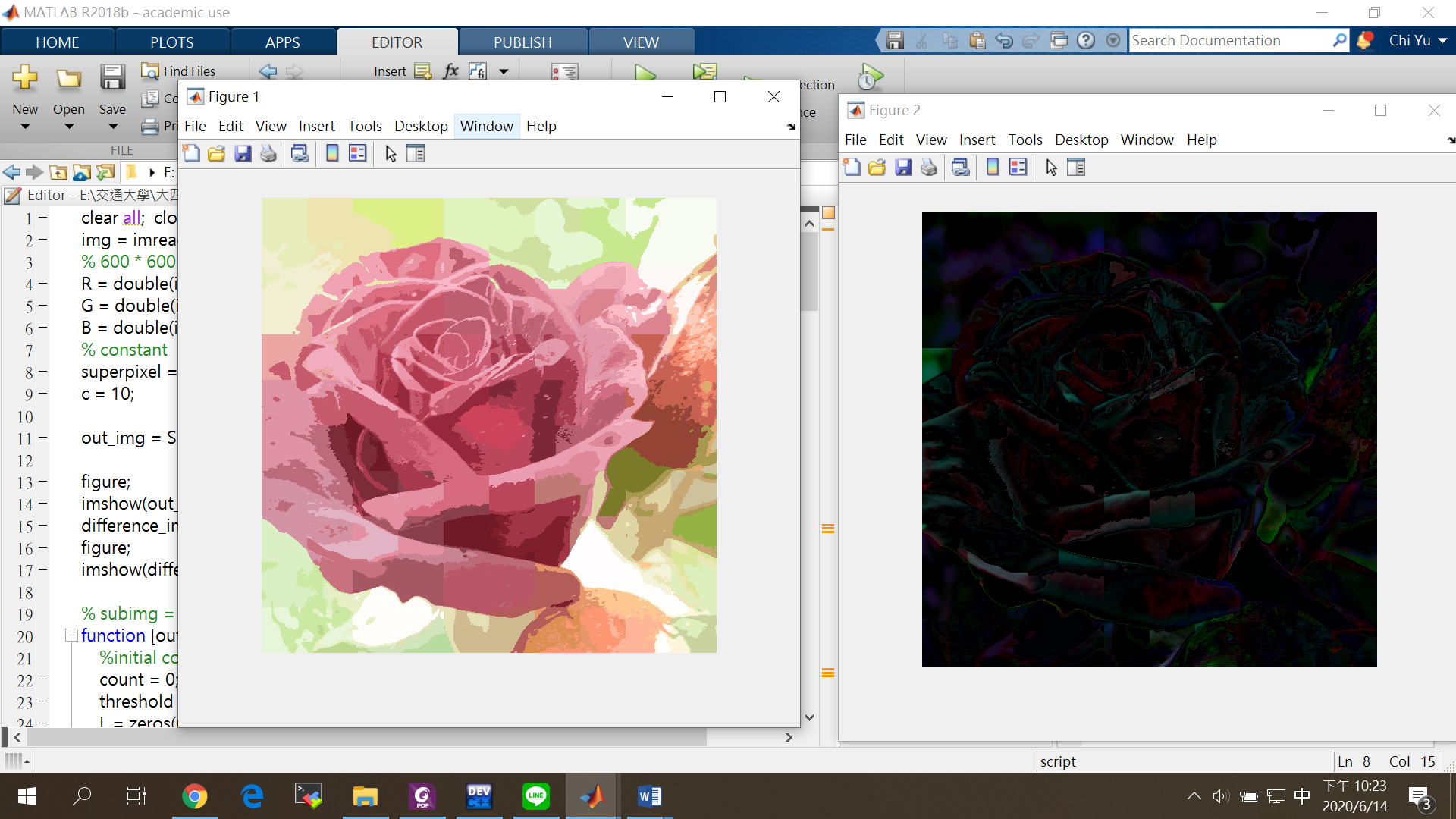
C=10,400 superpixel



C=1,100 superpixel



C=10,100 superpixel



Source codes

本次實驗使用Matlab 軟體分析(含註解)

clear all; close all; clc;

img = imread('image-pj7c.tif');

% 600 \* 600 \* 3

R = double(img(:,:,1));

G = double(img(:,:,2));

B = double(img(:,:,3));

% constant

superpixel = 100; % 100 & 400

c = 10;

out\_img = SLIC(superpixel, c, R, G, B);

figure;

imshow(out\_img./255, []);

difference\_img = double(img) - out\_img;

figure;

imshow(difference\_img./255, []);

% subimg = img(49:51, 3:5, :);

function [output] = SLIC(superpixel, c, R, G, B)

%initial constant

count = 0;

threshold = 10;

L = zeros(600, 600);

D = zeros(600, 600);

D = D-1;

step = floor(sqrt(360000/superpixel)); % s = 60 for 100, s = 30 for 400

m\_seeds = init\_seeds(step, superpixel, R, G, B);

while true

mi\_index = 1;

for i = step:step:600

for j = step:step:600

for k = (i-step+1):(i+step)

for l = (j-step+1):(j+step)

if((k<601)&&(l<601))

d = calculate\_D(m\_seeds, mi\_index, k, l, R, G, B, step, c);

if (d < D(k, l)) || (D(k, l) == -1)

D(k, l) = d;

L(k, l) = mi\_index;

end

end

end

end

mi\_index = mi\_index + 1;

end

end

m = zeros(superpixel, 5);

for i = 1:600

for j = 1:600

m(L(i, j), 1) = m(L(i, j), 1) + R(i, j);

m(L(i, j), 2) = m(L(i, j), 2) + G(i, j);

m(L(i, j), 3) = m(L(i, j), 3) + B(i, j);

m(L(i, j), 4) = m(L(i, j), 4) + i;

m(L(i, j), 5) = m(L(i, j), 5) + j;

end

end

for i = 1:superpixel

times = length(find(L(:) == i));

m(i, 1) = m(i, 1)/times;

m(i, 2) = m(i, 2)/times;

m(i, 3) = m(i, 3)/times;

m(i, 4) = floor(m(i, 4)/times)+1;

m(i, 5) = floor(m(i, 5)/times)+1;

end

for i = 1:superpixel

L( floor(m(i, 4))+1, floor(m(i, 5))+1 ) = i;

end

% calculate error

error = 0;

E = m\_seeds - m;

for i = 1:superpixel

sum = sqrt(E(i, 1).^2 + E(i, 2).^2 + E(i, 3).^2 + E(i, 4).^2 + E(i, 5).^2);

error = error + sum;

end

count = count + 1;

fprintf("%d times, error %f\n", count, error);

if(error < threshold)

break

end

m\_seeds = m;

end

fprintf("done");

for i = 1:600

for j = 1:600

output(i, j, 1) = m(L(i, j), 1);

output(i, j, 2) = m(L(i, j), 2);

output(i, j, 3) = m(L(i, j), 3);

end

end

end

function [m] = init\_seeds(step, superpixel, R, G, B)

m = zeros(superpixel, 5);

step\_sqrt = sqrt(superpixel);

gradient = find\_gradient(R, G, B);

for i = 1:superpixel

x = floor(i/step\_sqrt)+1;

y = mod(i, step\_sqrt);

if (y == 0)

y = step\_sqrt;

end

if (mod(i, step\_sqrt) == 0)

x = x - 1;

end

X = x\*step;

Y = y\*step;

m(i, 1) = R(X, Y);

m(i, 2) = G(X, Y);

m(i, 3) = B(X, Y);

for k = -1:1

for l = -1:1

displace\_x = X+k;

displace\_y = Y+l;

if((displace\_x<601)&&(displace\_y<601)&&(gradient(X, Y)>gradient(displace\_x, displace\_y)))

m(i, 4) = displace\_x;

m(i, 5) = displace\_y;

else

m(i, 4) = X;

m(i, 5) = Y;

end

end

end

end

end

function [g] = find\_gradient(R, G, B)

[g\_r a]= imgradient(R);

[g\_g b]= imgradient(G);

[g\_b c]= imgradient(B);

g = sqrt(g\_r.^2+g\_g.^2+g\_b.^2);

end

function D = calculate\_D(m, mi\_index, x, y, R, G, B, s, c)

Dc = (R(x, y)-m(mi\_index, 1)).^2+(G(x, y)-m(mi\_index, 2)).^2+(B(x, y)-m(mi\_index, 3)).^2;

Ds = (m(mi\_index, 4)-x).^2+(m(mi\_index, 5)-y).^2;

D = sqrt(Dc/c/c+Ds/s/s);

end