Ismail Akram 8834

I. Writing Assignments (10×4 = 40 points)

(1). How does an image change (e.g., objects' sizes in the image, field of view, etc.) when you change the zoom factors of your pinhole camera (i.e., the focal length of a pinhole camera is changed)?

With perspective projection through a pinhole camera:

- When the zoom factor increases (i.e. as focal length, f, increases), the objects' sizes in the image increases and the field of view narrows.
- When the zoom factor decreases (i.e. as focal length, f, decreases), the objects' sizes in the image decreases and the field of view expands.

(2). Give an intuitive explanation why a pinhole camera has an infinite depth of field, i.e., the images of objects are always sharp regardless their distances from the camera.

We know (from slide 13 of lecture 1) that "each point in (an) image corresponds to a particular direction defined by a ray from that point through the pinhole".

For any camera, the depth of field increases as the lens aperture gets smaller. This is because a smaller aperture has sufficient focus to resolve all objects. Less light rays to deal with when they come in as they're more organized as a point of focus.

Only one ray per unit point creates a projection, so it's sharp in the image plane.

So, a pinhole camera (i.e. smallest hole as possible) would have infinite depth of field.

(3). In the thin lens model, $1/o + 1/i = 1/f$, there are three variables, the focal length f, the object distance o and the image distance i (please refer to the Slides of the Image Formation lecture). If we define $Z = o-f$, and $z = i-f$, please write a few words to describe the physical meanings of Z and z, and then prove that $Z*z = f*f$ given $1/o + 1/i = 1/f$.

"Z" physically means "object distance (distance between lens and object) subtracted by focal length".

"z" physically means "image distance subtracted (distance between image plane and lens) by focal length".

Proof:

We know that Z = o-f and z = i-f, and that they're positive (physical lengths)

Given 1/o + 1/i = 1/f

i*f + o*f = o*i … multiplying across by o*i*f

Finding Z:

i*f + o*f - o*i = 0
-i(o-f) + o*f = 0
-i*z + o*f = 0
Z = (o*f)/i

Finding z:

i*f + o*f - o*i = 0

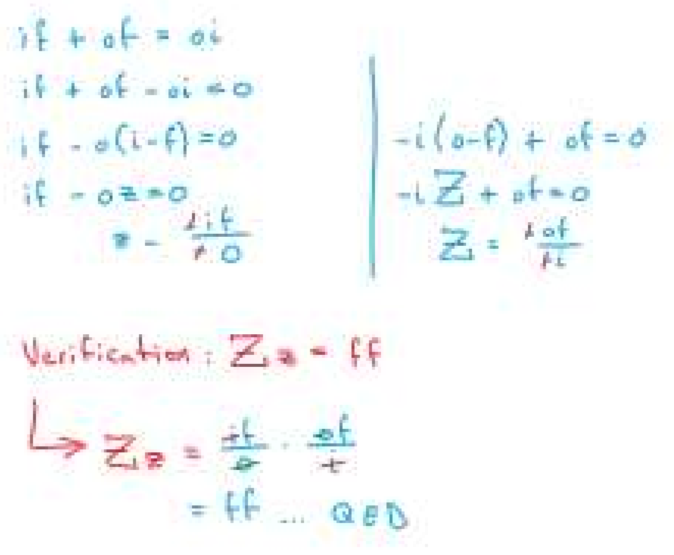
i*f – o(i-f) = 0

i*f – o*z = 0

z = (i*f)/o

Verifying if Z*z = f*f

Z*z = [(i*f)/o] * [(o*f)/i]
(i*f) * (f/i) … o's cancel
f*f … i's cancel
Z*z = f*f … QED



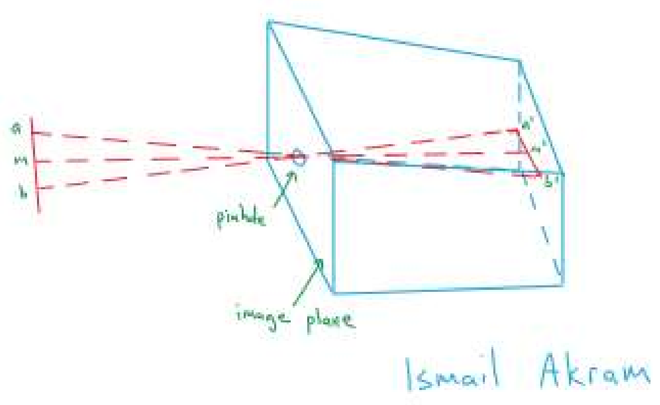
$if + of = oi$
$if + of - oi = 0$
$if - o(i-f) = 0$
$if - oz = 0$
$z = \frac{if}{o}$

$-i(o-f) + of = 0$
$-iZ + of = 0$
$Z = \frac{of}{i}$

Verification : $Zz = ff$

$\hookrightarrow Zz = \frac{if}{o} \cdot \frac{of}{i}$
$= ff$ … QED

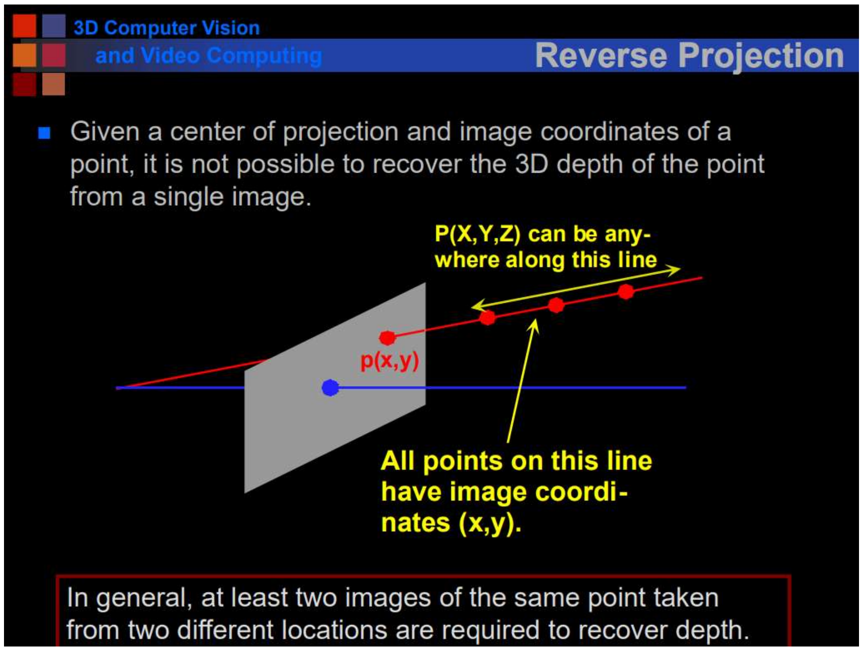
(4). Prove that, in the pinhole camera model, three collinear points in the world (i.e., they lie on a line in 3D space) are imaged into three collinear points on the image plane. You may either use geometric reasoning (with line drawings) or algebra deduction (using equations).



In the crude diagram I drew (based on the lecture slides (slide 24 “Reverse Projection”)), we can see that the points: (a, m, b), form a straight line. And when they are imaged through the pinhole, they also form a straight line (a’, m’, b’).

Similarly, all points along these projection lines are colinear (red dotted lines), and when they’re imaged, they’re still colinear. Any points along these lines will still be colinear when projected through the image plane.

Slide 24 for reference:


3D Computer Vision
and Video Computing

Reverse Projection

- Given a center of projection and image coordinates of a point, it is not possible to recover the 3D depth of the point from a single image.

P(X,Y,Z) can be anywhere along this line
p(x,y)
All points on this line have image coordinates (x,y).

In general, at least two images of the same point taken from two different locations are required to recover depth.

II. Programming Assignments

1. Read in a color image C1(x,y) = (R(x,y), G(x,y), B(x,y)) in Windows BMP format, and display it.
   Already done for us (Fig 1).

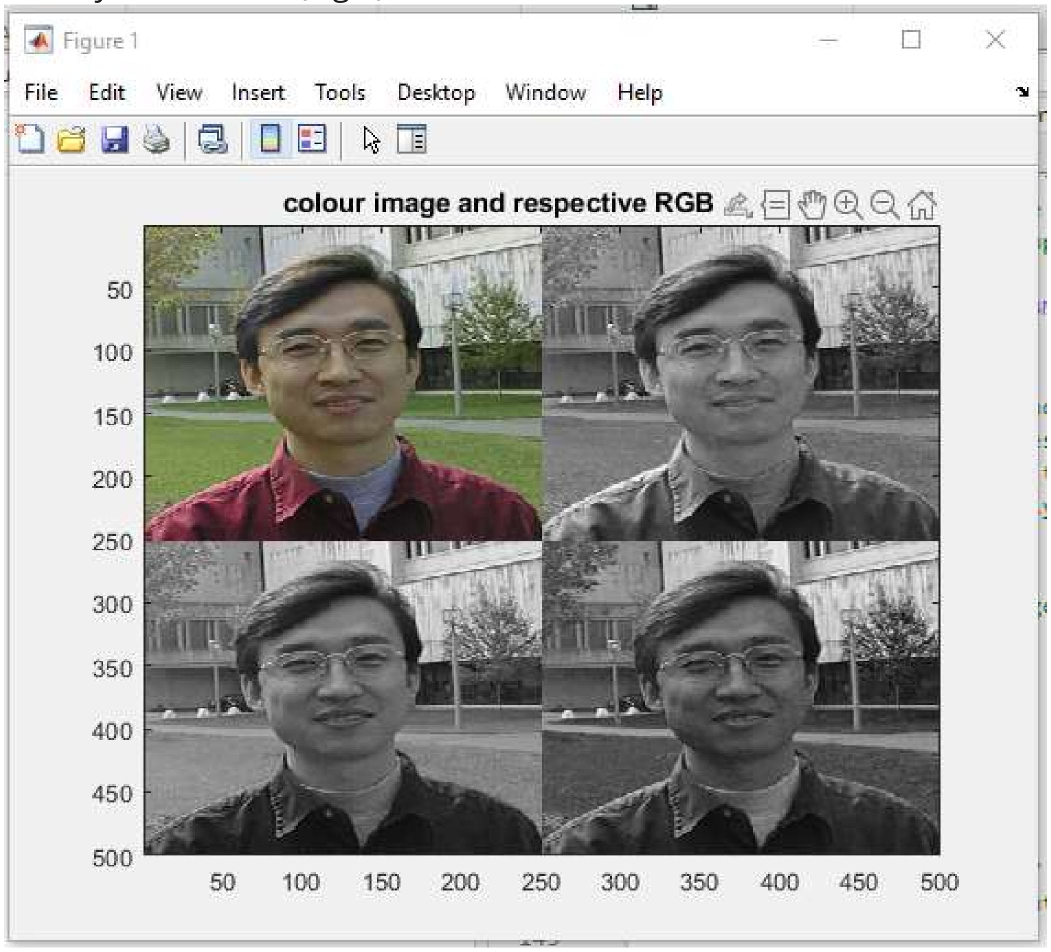


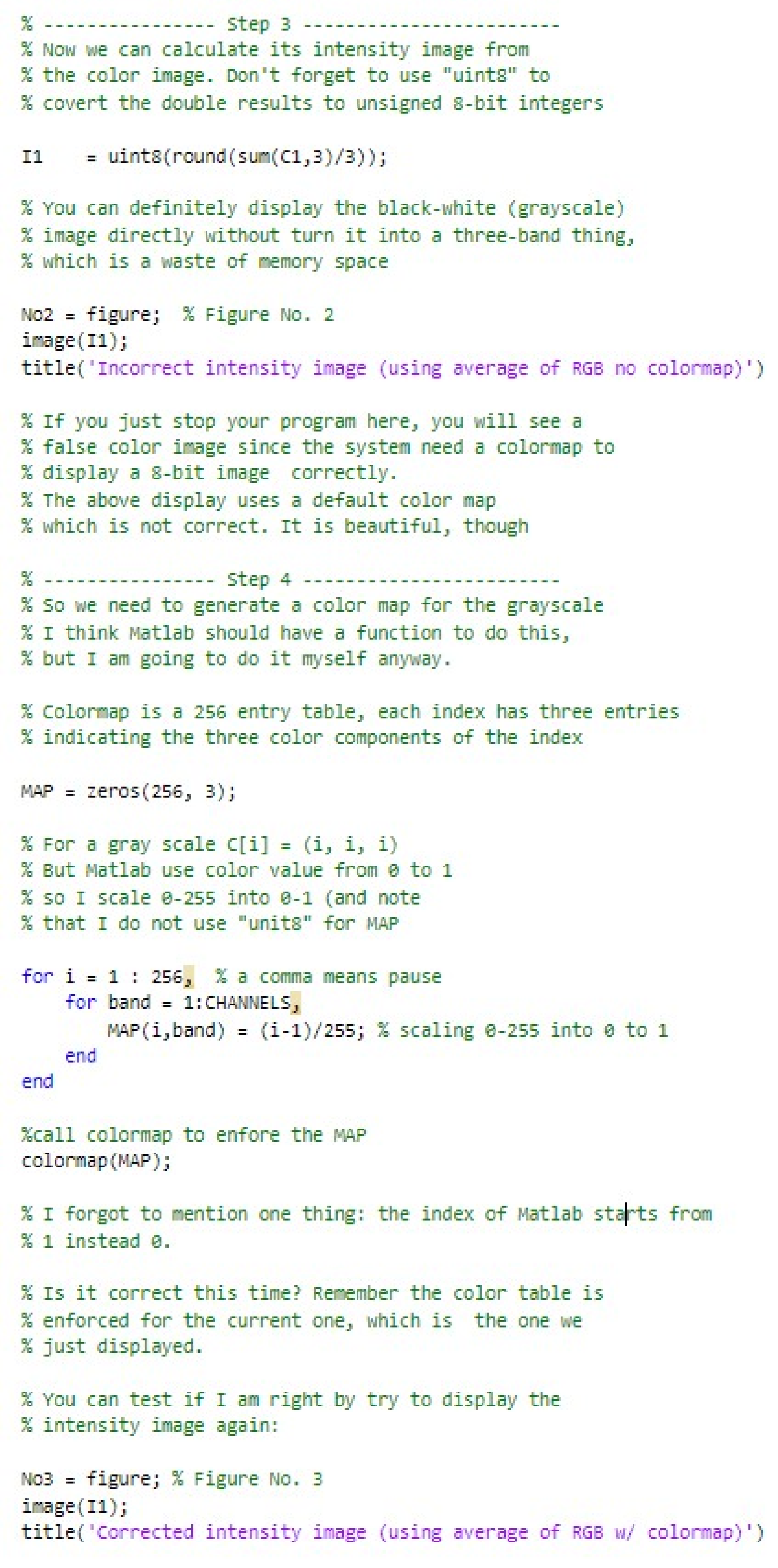
Figure 1
File Edit View Insert Tools Desktop Window Help
colour image and respective RGB
50
100
150
200
250
300
350
400
450
500
50 100 150 200 250 300 350 400 450 500

Fig 1. C1 and the respective RGB bands

2. Display the images of the three color components, R(x,y), G(x,y) and B(x,y), separately. You should display three black-white-like images.

   Already done for us (Fig 1).

3. Generate an intensity image I(x,y) and display it. You should use the equation I = 0.299R + 0.587G + 0.114B (the NTSC standard for luminance) and tell us what are the differences between the intensity image thus generated from the one generated using a simple average of the R, G and B components. Please use an algorithm to show the differences instead by just observing the images by your eyes.



```
% ---------------- Step 3 ------------------------
% Now we can calculate its intensity image from
% the color image. Don't forget to use "uint8" to
% covert the double results to unsigned 8-bit integers

I1    = uint8(round(sum(C1,3)/3));

% You can definitely display the black-white (grayscale)
% image directly without turn it into a three-band thing,
% which is a waste of memory space

No2 = figure;  % Figure No. 2
image(I1);
title('Incorrect intensity image (using average of RGB no colormap)')

% If you just stop your program here, you will see a
% false color image since the system need a colormap to
% display a 8-bit image  correctly.
% The above display uses a default color map
% which is not correct. It is beautiful, though

% ---------------- Step 4 ------------------------
% So we need to generate a color map for the grayscale
% I think Matlab should have a function to do this,
% but I am going to do it myself anyway.

% Colormap is a 256 entry table, each index has three entries
% indicating the three color components of the index

MAP = zeros(256, 3);

% For a gray scale C[i] = (i, i, i)
% But Matlab use color value from 0 to 1
% so I scale 0-255 into 0-1 (and note
% that I do not use "unit8" for MAP

for i = 1 : 256,  % a comma means pause
    for band = 1:CHANNELS,
        MAP(i,band) = (i-1)/255; % scaling 0-255 into 0 to 1
    end
end

%call colormap to enfore the MAP
colormap(MAP);

% I forgot to mention one thing: the index of Matlab starts from
% 1 instead 0.

% Is it correct this time? Remember the color table is
% enforced for the current one, which is  the one we
% just displayed.

% You can test if I am right by try to display the
% intensity image again:

No3 = figure; % Figure No. 3
image(I1);
title('Corrected intensity image (using average of RGB w/ colormap)')
```

Source code: first part of Q3 done for us.

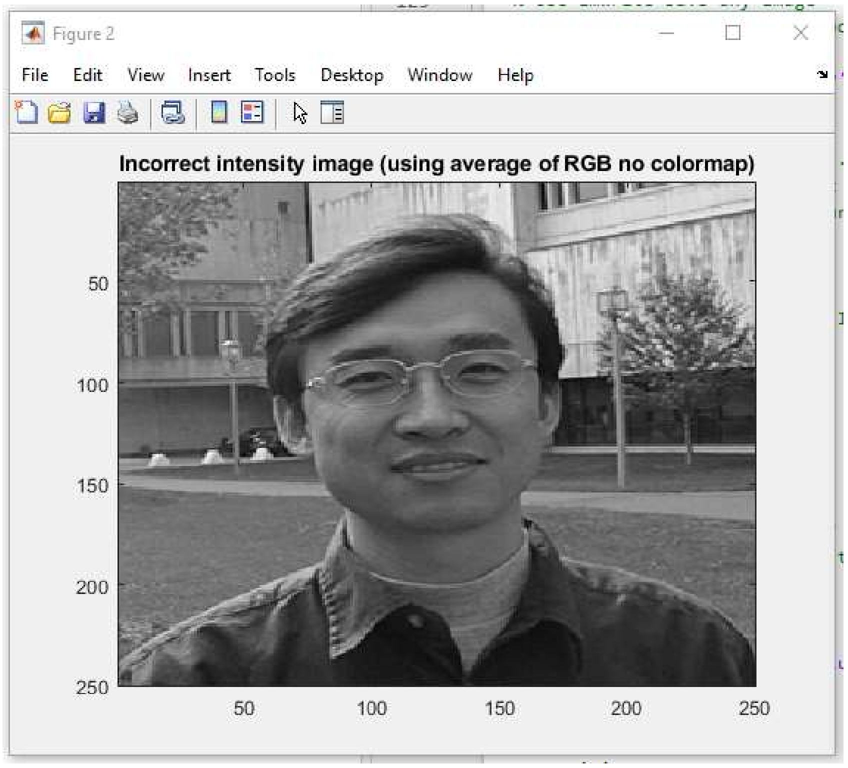


Figure 2
File Edit View Insert Tools Desktop Window Help
Incorrect intensity image (using average of RGB no colormap)
50
100
150
200
250
50
100
150
200
250

Fig 2. Grayscale image done by professor (using average)

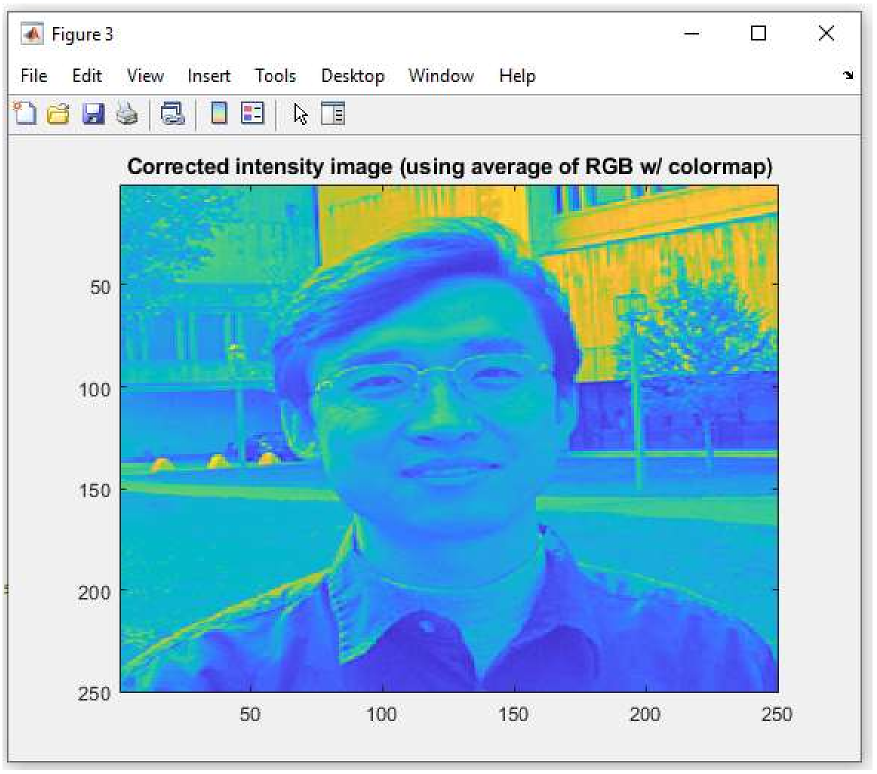
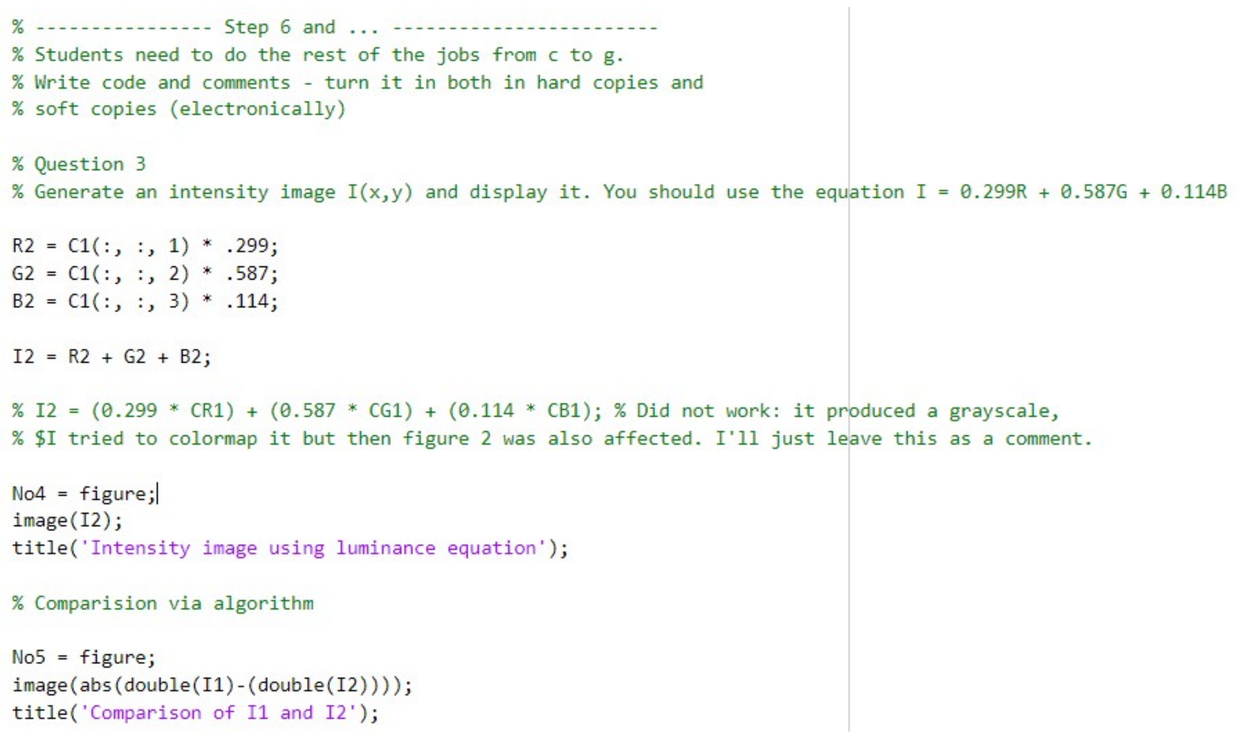


Figure 3
File Edit View Insert Tools Desktop Window Help
Corrected intensity image (using average of RGB w/ colormap)
50
100
150
200
250
50
100
150
200
250

Fig 3. Corrected colormap done by professor (using average)



```
% ---------------- Step 6 and ... ------------------------
% Students need to do the rest of the jobs from c to g.
% Write code and comments - turn it in both in hard copies and
% soft copies (electronically)

% Question 3
% Generate an intensity image I(x,y) and display it. You should use the equation I = 0.299R + 0.587G + 0.114B

R2 = C1(:, :, 1) * .299;
G2 = C1(:, :, 2) * .587;
B2 = C1(:, :, 3) * .114;

I2 = R2 + G2 + B2;

% I2 = (0.299 * CR1) + (0.587 * CG1) + (0.114 * CB1); % Did not work: it produced a grayscale,
% $I tried to colormap it but then figure 2 was also affected. I'll just leave this as a comment.

No4 = figure;
image(I2);
title('Intensity image using luminance equation');

% Comparision via algorithm

No5 = figure;
image(abs(double(I1)-(double(I2))));
title('Comparison of I1 and I2');
```

Source code: using the luminance equation.

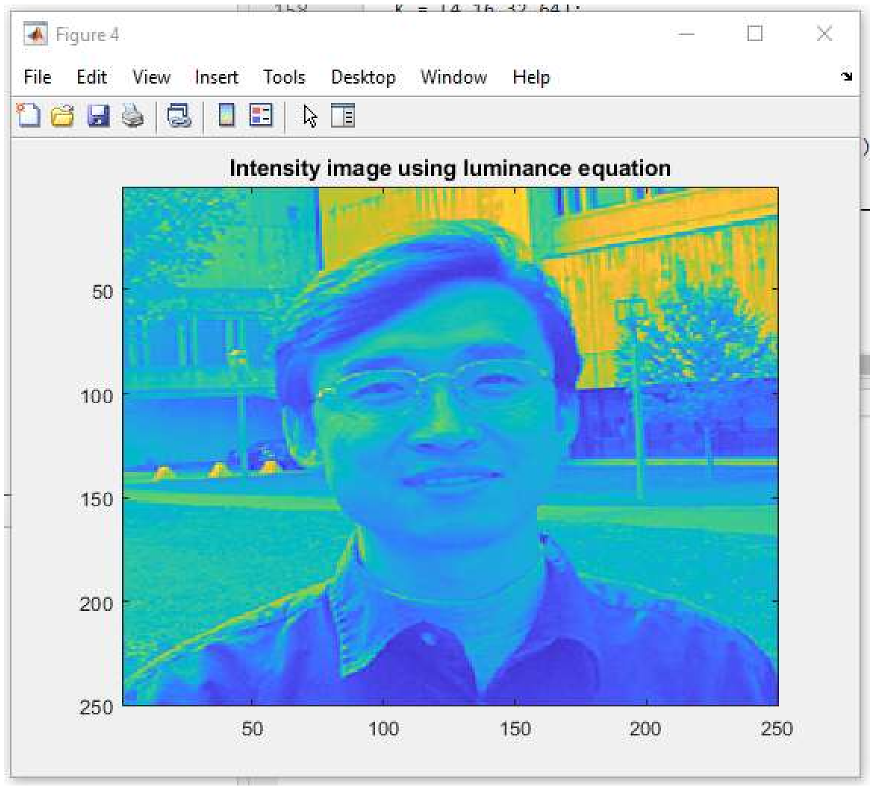


Fig 4. My colour intensity image generated from luminance equation.

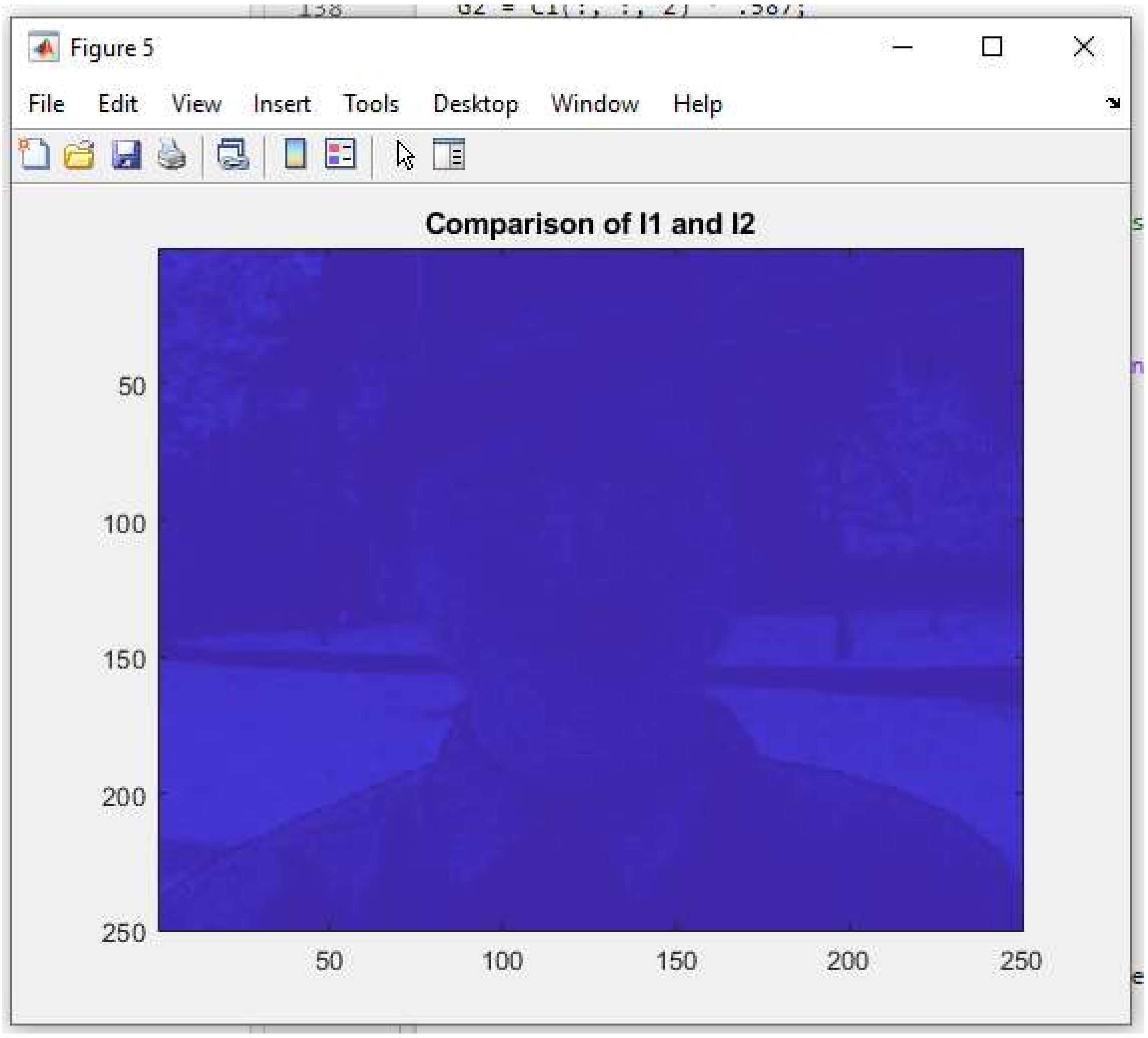


Fig 5. Comparison of figure 3 (average sum) and figure 4 (luminance equation) through algorithm.

I used a subtraction algorithm, <`image(abs(double(I1)-(double(I2))));`>, for this. The idea behind this is to subtract the absolute double values of I1 (your intensity image via averaging) and I2 (my intensity image via the equation).

4. The original intensity image should have 256 gray levels. Please uniformly quantize this image into K levels (with K=4, 16, 32, 64). As an example, when K=2, pixels whose values are below 128 are turned to 0, otherwise to 255. Display the four quantized images with four different K levels and tell us how the images still look like or different from the original ones, and where you cannot see any differences.

```
% Question 4
K = [4 16 32 64]; % respective
No6 = figure;

for i = 1:length(K)
    Img_thres = I1;
    intervals = round(linspace(1,256,K(i) + 1)); % +1 since we're using 1-256. To account for 0-255. In MATLAB, our first index is 1.
    for j = 1:length(intervals)-1
        Img_thres(I_thres > intervals(j) & Img_thres < intervals (j+1)) = intervals(j);
    end
    subplot(2, 2, i);
    imshow(Img_thres);
    title('I1 Quantize');
end
```

Source code: quantizing original intensity image into K levels

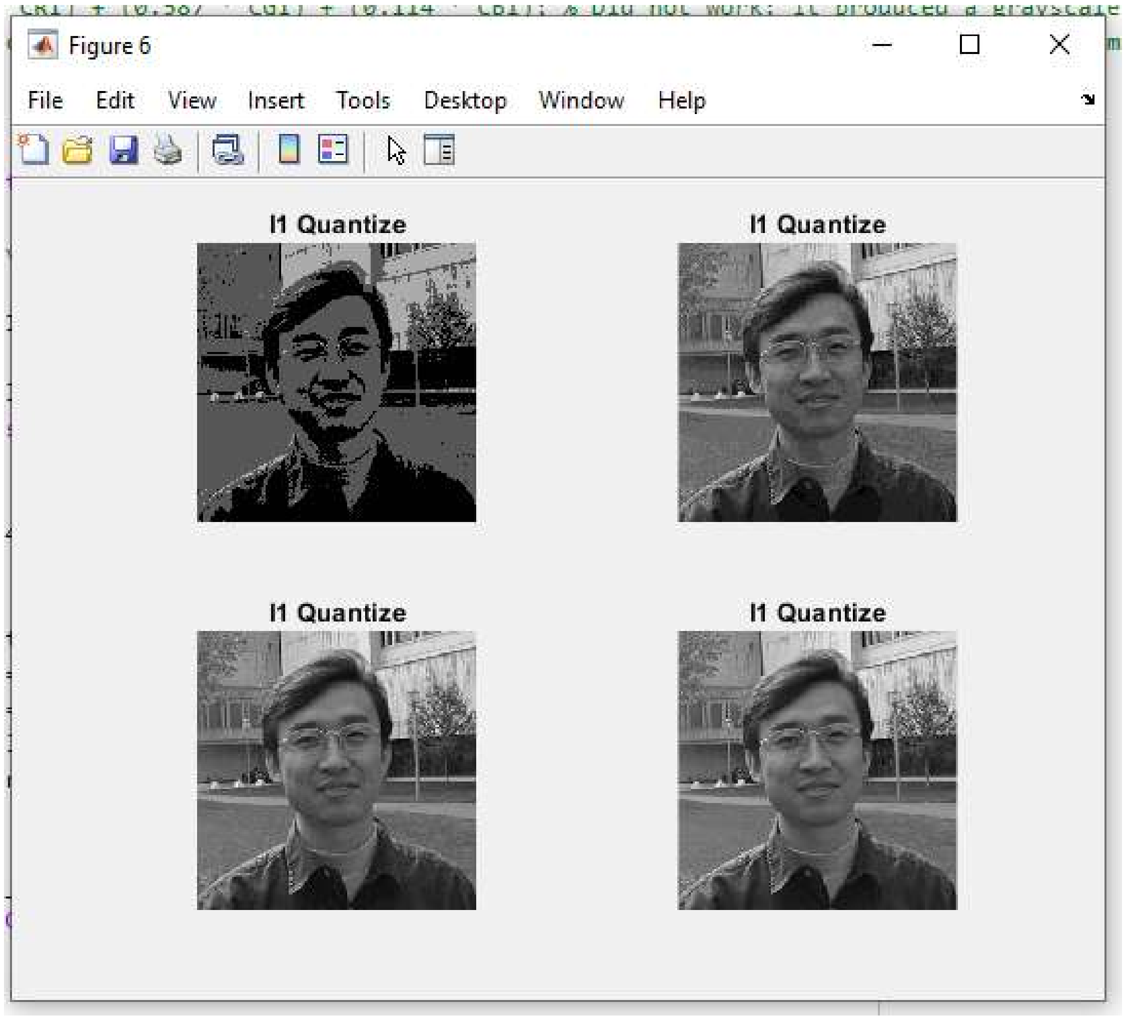


Fig 6. Quantization into K levels 4, 16, 32, 64.

I can tell lower resolution difference in the first 3 images (K = 4, 16, 32), but the last one (bottom right; K = 64) looks near identical to the original intensity image.

5. Quantize the original three-band color image C1(x,y) into K level color images CK(x,y)= (R'(x,y), G'(x,y), B'(x,y)) (with uniform intervals) , and display them. You may choose K=2 and 4 (for each band). Do they have any advantages in viewing and/or in computer processing (e.g. transmission or segmentation)?

```
% Question 5
K_c = [2 4]; % respective
No7 = figure;

for i = 1:length(K_c)
    Img_thres = C1;
    intervals = round(linspace(1,256,K_c(i) + 1)); % +1 since we're using 1-256. To account for 0-255. In MATLAB, our first index is 1.
    for j = 1:length(intervals)-1
        Img_thres(I_thres > intervals(j) & Img_thres < intervals (j+1)) = intervals(j);
    end
    subplot(1, 2, i);
    imshow(Img_thres);
    title('I1 Quantize');
end
```

Source code: Quantizing the original three-band color image into K levels 2 and 4

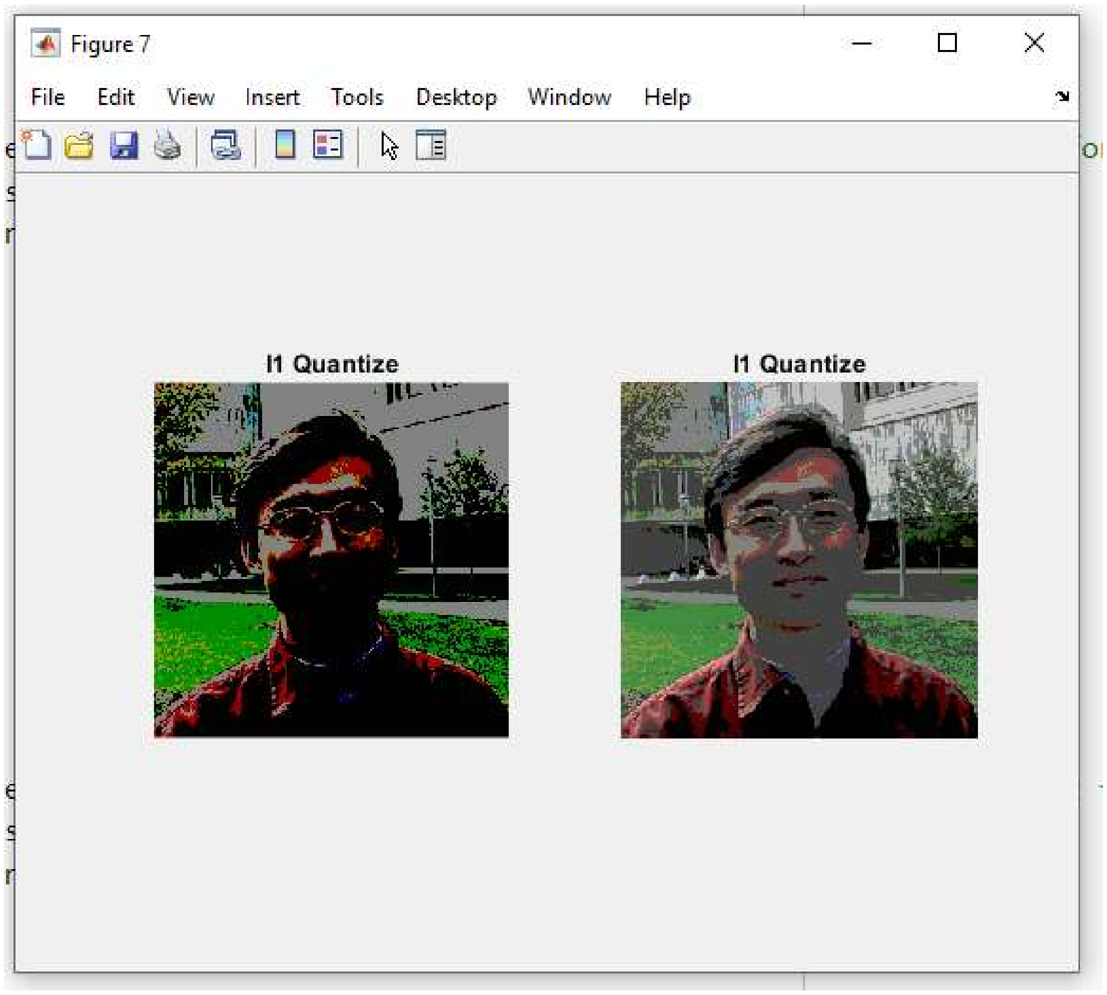


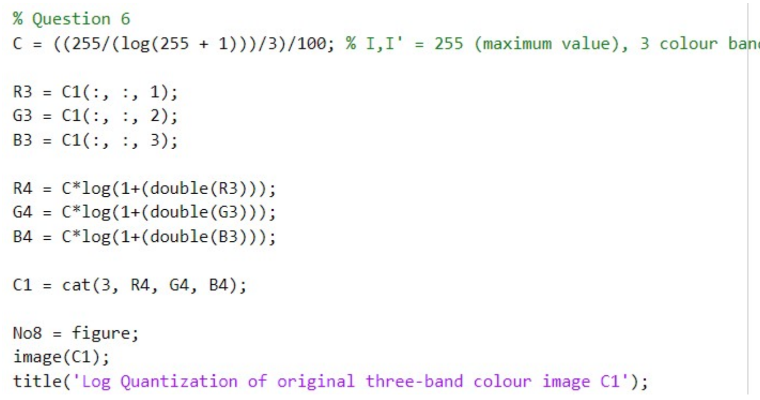
Fig 7. Quantizing the original three-band color image into K levels 2 and 4 output.

Uniform quantization divides the signal range into K equal sized intervals (in this case 2 and 4). All the signal values within an interval are represented by an associated integer which can define a mapping. In the case of log quantization (next question), we can enhance detail in low signal values at the expense of details in high signal values.

For K = 2 you only need 1 bit, for K = 4, you need 4 bits.

At a certain threshold (at about K =32 and beyond), there’s diminishing returns that is imperceivable to the human eye.

6. Quantize the original three-band color image C1(x,y) into a color image CL(x,y)= (R'(x,y), G'(x,y), B'(x,y)) (with a logarithmic function) , and display it. You may choose a function I' =C ln (I+1) (for each band), where I is the original value (0~255), I' is the quantized value, and C is a constant to scale I' into (0~255), and ln is the natural logarithmic function. Please describe how you find the best C value so for an input in the range of 0-255, the output range is still 0 – 255. Note that when I = 0, I' = 0 too.



```
% Question 6
C = ((255/(log(255 + 1)))/3)/100; % I,I' = 255 (maximum value), 3 colour bands, 100 is referring to opacity. So C ~= 0.15...

R3 = C1(:, :, 1);
G3 = C1(:, :, 2);
B3 = C1(:, :, 3);

R4 = C*log(1+(double(R3)));
G4 = C*log(1+(double(G3)));
B4 = C*log(1+(double(B3)));

C1 = cat(3, R4, G4, B4);

No8 = figure;
image(C1);
title('Log Quantization of original three-band colour image C1');
```

Source Code: log quantization of original three-band colour image

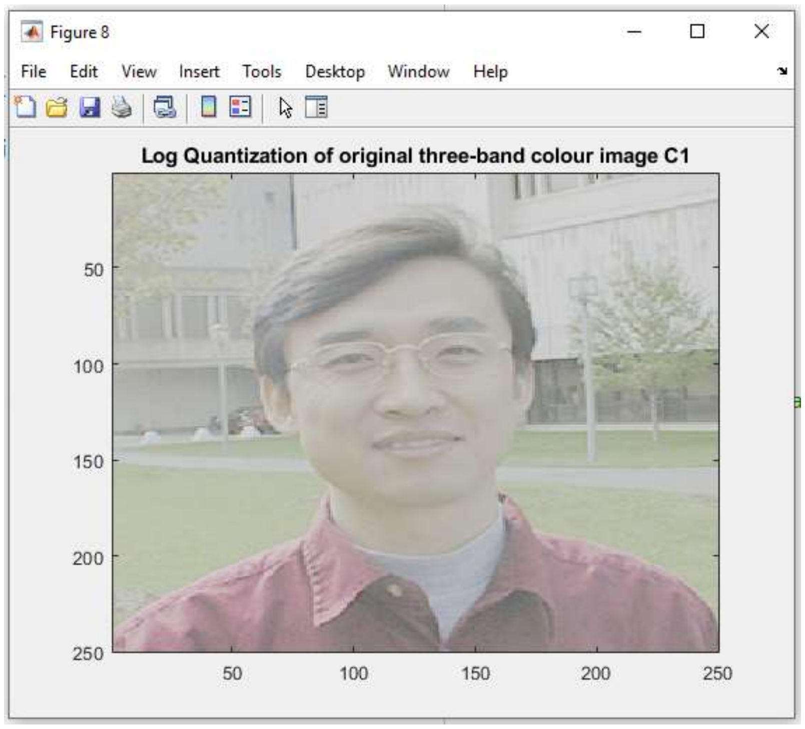


Figure 8
File Edit View Insert Tools Desktop Window Help
Log Quantization of original three-band colour image C1
50
100
150
200
250
50
100
150
200
250

Fig 8. log quantization of original three-band colour image output.

Determining the best value of C:

We know that I = 0 and I’ = 0 as well (given) and that I’ =C ln (I+1) (for each band).

So we used the maximum value of I and I’ and accounted for 3 colour bands while factoring in opacity and solved for C.

Thus we get the equation:

C = ([I’/ln(I+1)]/3)/100

C = ([(255)/ln(255+1)]/3)/100

C = 0.15