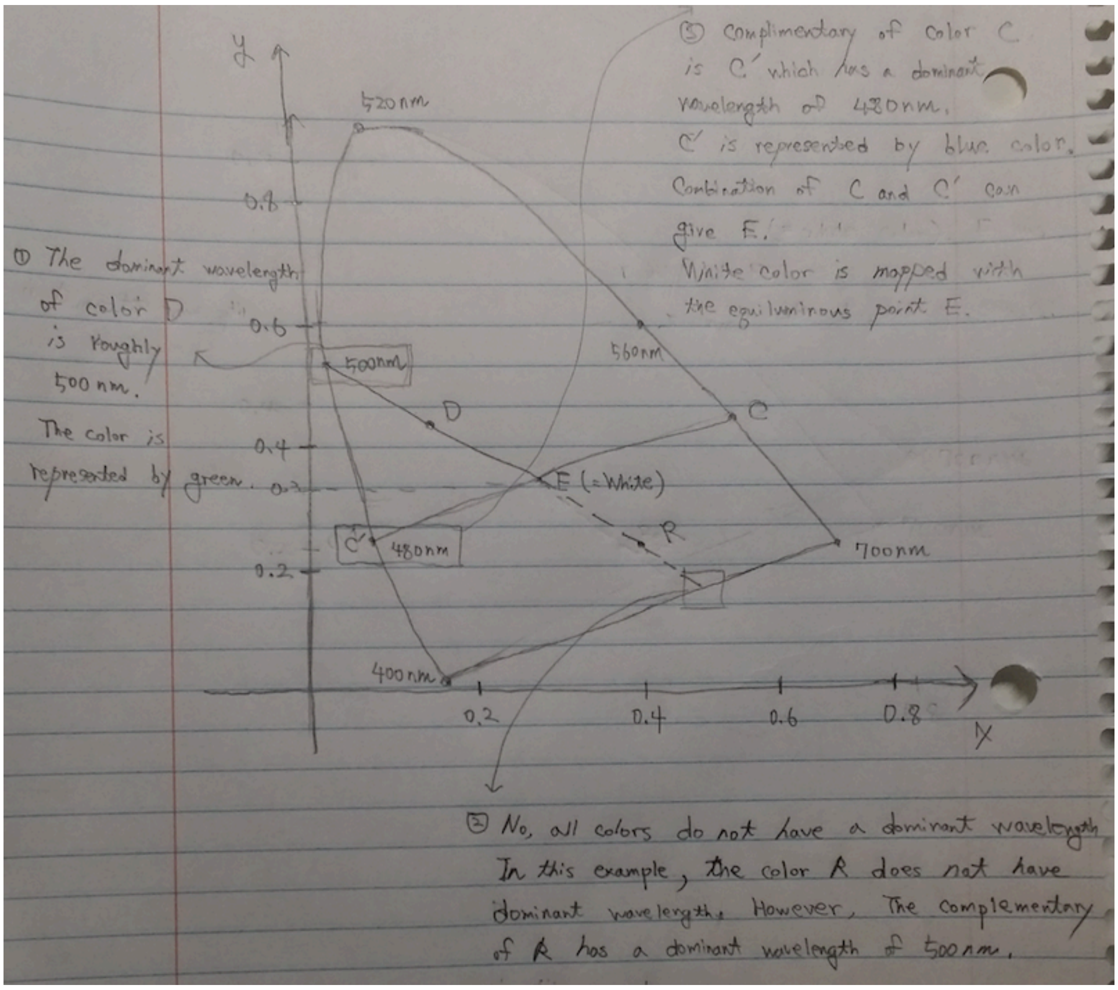
1. Color Theory

(1) In the image alongside find the dominant wavelength of color D. Show this wavelength.

(2) Do all colors have a dominant wavelength? Explain your reasoning.

(3) Find the color which is complimentary to the color C and plot its location.
What colors in the three dimensional RGB color space map to the equiluminous point E upon projection into the 2D chromaticity space.



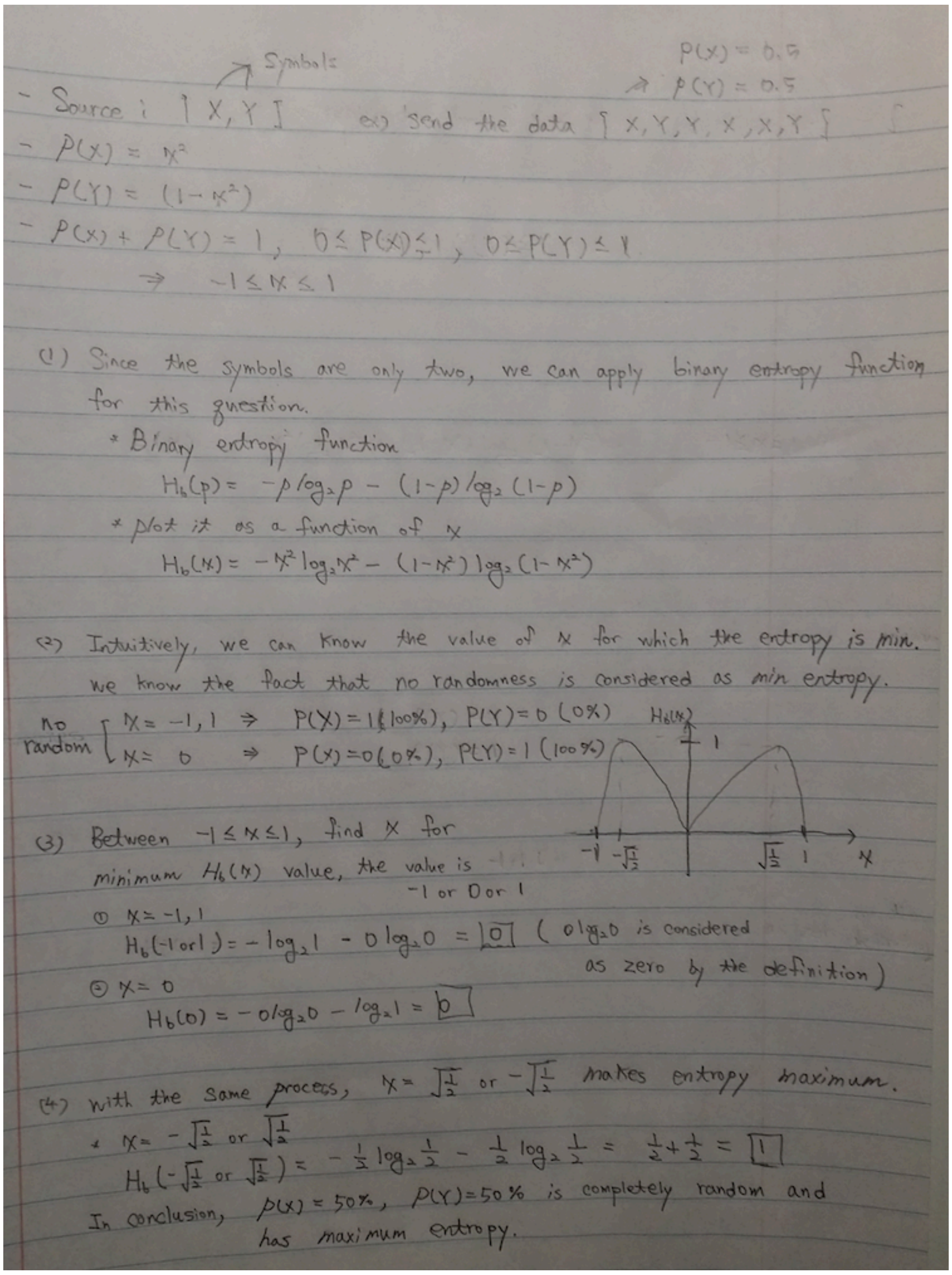
③ Complimentary of color C is C′ which has a dominant wavelength of 480nm.
C′ is represented by blue color.
Combination of C and C′ can give E.
White color is mapped with the equiluminous point E.

① The dominant wavelength of color D is roughly 500 nm.
The color is represented by green.

② No, all colors do not have a dominant wavelength
In this example, the color R does not have dominant wavelength. However, The complementary of R has a dominant wavelength of 500nm.

2. Generic Compression

(1) Write down the entropy function and plot it as a function of *x*.
(2) From your plot, for what value of *x* does the Entropy become a minimum?
(3) Although the plot visually gives you the value of *x* for which the entropy in minimum, can you now mathematically find out the value(s) for which the entropy is a minimum?
(4) Can you do the same for the maximum, that is can you find out value(s) of *x* for which the value is a maximum?



Symbols

$P(X) = 0.5$

$P(Y) = 0.5$

- Source : { X, Y } ex) Send the data { X, Y, Y, X, X, Y }
- $P(X) = x^2$
- $P(Y) = (1 - x^2)$
- $P(X) + P(Y) = 1$, $0 \le P(X) \le 1$, $0 \le P(Y) \le 1$

$\Rightarrow -1 \le x \le 1$

(1) Since the symbols are only two, we can apply binary entropy function for this question.

* Binary entropy function

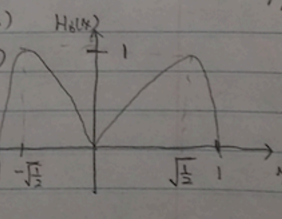
$$H_b(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$

* plot it as a function of x

$$H_b(x) = -x^2 \log_2 x^2 - (1-x^2) \log_2 (1-x^2)$$

(2) Intuitively, we can know the value of x for which the entropy is min. we know the fact that no randomness is considered as min entropy.

no random $\begin{cases} x = -1, 1 \Rightarrow P(X) = 1 (100\%),\ P(Y) = 0\ (0\%) \\ x = 0 \Rightarrow P(X) = 0 (0\%),\ P(Y) = 1\ (100\%) \end{cases}$



(3) Between $-1 \le x \le 1$, find x for minimum $H_b(x)$ value, the value is -1 or 0 or 1

① $x = -1, 1$

$H_b(-1 \text{ or } 1) = -\log_2 1 - 0 \log_2 0 = \boxed{0}$ ($0 \log_2 0$ is considered as zero by the definition)

② $x = 0$

$H_b(0) = -0 \log_2 0 - \log_2 1 = \boxed{0}$

(4) with the same process, $x = \sqrt{\frac{1}{2}}$ or $-\sqrt{\frac{1}{2}}$ makes entropy maximum.

* $x = -\sqrt{\frac{1}{2}}$ or $\sqrt{\frac{1}{2}}$

$H_b(-\sqrt{\frac{1}{2}} \text{ or } \sqrt{\frac{1}{2}}) = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = \frac{1}{2} + \frac{1}{2} = \boxed{1}$

In conclusion, $P(X) = 50\%$, $P(Y) = 50\%$ is completely random and has maximum entropy.

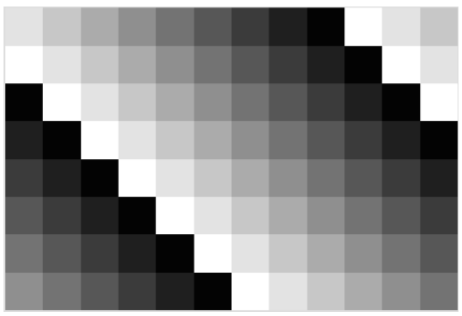
3. Image Dithering

(1) Plot the image and make sure that you submit a “zoomed” in image which properly shows all image gray values.

(a) Original Image should be represented as 8 bits per pixel. In the 8 bits gray scale color space, 0 should be represented as 255(=white); and **9 should be represented as 0(=black).**

Original_image

1	2	3	4	5	6	7	8	9	0	1	2
0	1	2	3	4	5	6	7	8	9	0	1
9	0	1	2	3	4	5	6	7	8	9	0
8	9	0	1	2	3	4	5	6	7	8	9
7	8	9	0	1	2	3	4	5	6	7	8
6	7	8	9	0	1	2	3	4	5	6	7
5	6	7	8	9	0	1	2	3	4	5	6
4	5	6	7	8	9	0	1	2	3	4	5



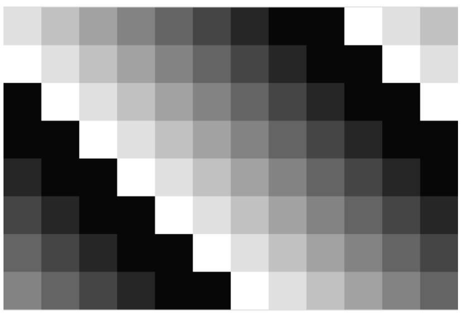
(b) **The normalization is not complete in this example.** Each value of the dithering matrix is composed of 0 ~ 8. However, the value of each pixel in the example image is composed of 0 ~ 9. Therefore, it should be reduced to 0 ~ 8 space in order to fit into the dithering matrix. It means 0 should be represented as 255(=white); **8 should be represented as 0(=black)** in gray scale color space.

Original image:	0.0~0.9,	1.0~1.9,	2.0~2.9,	3.0~3.9,	4.0~4.9,	5.0~5.9,	6.0~6.9,	7.0~7.9,	8.0~9.0.
Normalized image:	0	1	2	3	4	5	6	7	8

It means 8 and 9 values of original image should be mapped with 8 of the normalized image.

Normalized_image

1	2	3	4	5	6	7	8	8	0	1	2
0	1	2	3	4	5	6	7	8	8	0	1
8	0	1	2	3	4	5	6	7	8	8	0
8	8	0	1	2	3	4	5	6	7	8	8
7	8	8	0	1	2	3	4	5	6	7	8
6	7	8	8	0	1	2	3	4	5	6	7
5	6	7	8	8	0	1	2	3	4	5	6
4	5	6	7	8	8	0	1	2	3	4	5



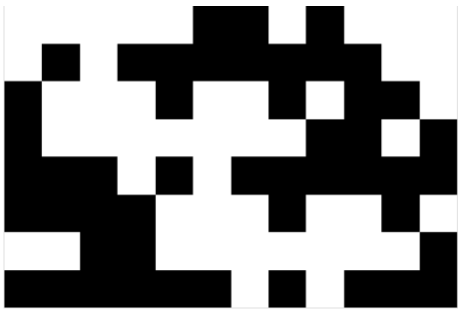
(2) Compute the output of a dithering operation using the dithering matrix D given below. Assume that the image top left coordinate indexes are [0, 0]. Show a graphical binary image plot of the dithered output.

- I wrote the code to generate output using Dithering algorithm for 3x3 matrix. According to the lecture, **when I(x,y) > D(i,j),** a pixel at point (x,y) should be intensified. I followed the lecture algorithm.

```
For all x pixels
   For all y pixels
      i = x mod 3
      j = y mod 3
   if Normalization_image[x,y] > Dithering_maxtrix[i,j] (not >= operation)
      Dithered_image[x,y] = 1;
      setRGB(x, y, dark_pix)
   else
      Dithered_image[x,y] = 0;
      setRGB(x, y, white_pix)
```

Dithered_image
(Starting with top left coordinate [0,0])

0	0	0	0	0	1	1	0	1	0	0	0
0	1	0	1	1	1	1	1	1	1	0	0
1	0	0	0	1	0	0	1	0	1	1	0
1	0	0	0	0	0	0	0	1	1	0	1
1	1	1	0	1	0	1	1	1	1	1	1
1	1	1	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	0	0	0	1
1	1	1	1	1	1	0	1	0	1	1	1

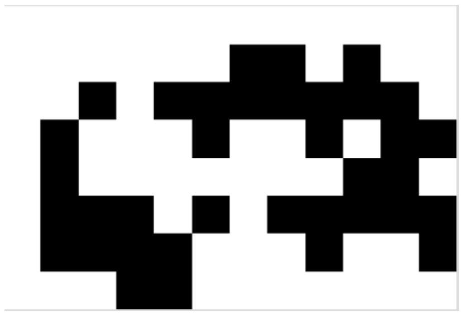


(3) What if the image block's top left coordinate indexes start with [1, 1]. Show a graphical binary image plot of the dithered output.

- The description does not mention how to deal with first row [0,0] ~ [0,11] and first column [0,0] ~ [7,0]. Therefore, I did not consider first row and first column (just keep white for the first row and first column).

Dithered_image
(Starting with top left coordinate [1,1])

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	0	0	0	0	0	1	1	0	1	0	0
0	0	1	0	1	1	1	1	1	1	1	0
0	1	0	0	1	0	0	1	0	1	1	1
0	1	0	0	0	0	0	0	0	1	1	0
0	1	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0	0	0	0	0



I found the interesting result of (3) which has a similar pattern with (2). In this example of the original image, I realized that the pattern is iterative. For example, each value of [0,0],[0,1],[0,2],[1,0][1,1][1,2],[2,0],[2,1],[2,2] is equal to [1,1][1,2][1,3],[2,1][2,2][2,3],[3,1][3,2][3,3] respectively.

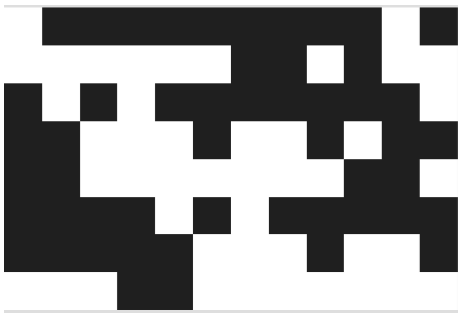
1	2	3	4	5	6	7	8	9	0	1	2
0	1	2	3	4	5	6	7	8	9	0	1
9	0	1	2	3	4	5	6	7	8	9	0
8	9	0	1	2	3	4	5	6	7	8	9
7	8	9	0	1	2	3	4	5	6	7	8
6	7	8	9	0	1	2	3	4	5	6	7
5	6	7	8	9	0	1	2	3	4	5	6
4	5	6	7	8	9	0	1	2	3	4	5

Even though there are no instructions for dealing with first row and first column, I try to come up with the idea of considering about them. Please refer to the image below. After retrieving the end of the row and column, first row and first column could be retrieved.

1	2	3	4	5	6	7	8	9	0	1	2
0	1	2	3	4	5	6	7	8	9	0	1
9	0	1	2	3	4	5	6	7	8	9	0
8	9	0	1	2	3	4	5	6	7	8	9
7	8	9	0	1	2	3	4	5	6	7	8
6	7	8	9	0	1	2	3	4	5	6	7
5	6	7	8	9	0	1	2	3	4	5	6
4	5	6	7	8	9	0	1	2	3	4	5

The result is

0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**0**	**1**
0	0	0	0	0	0	1	1	0	1	0	0
1	0	1	0	1	1	1	1	1	1	1	0
1	1	0	0	0	1	0	0	1	0	1	1
1	1	0	0	0	0	0	0	0	1	1	0
1	1	1	1	0	1	0	1	1	1	1	1
1	1	1	1	1	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0	0	0	0	0



4. Programming Assignment

I will briefly explain for implementation step by step.

(1) Break all channels into 8x8 blocks.
- I made 5-dimension array for breaking into the blocks.
- RGB_block[red or green or blue][image_height/block_height]
  [image_width/block_width][block_height][block_width]
- In our assignment, RGB_block[3][36][44][8][8]
- Time complexity is same as 1-dimension or 2-dimension array (352*288*3)

(2) Convert the RGB blocks into YPrPb blocks.
- Since each value of the following matrix is not integer, the type of RGB blocks is represented as “double.”

(3) DCT operation for YPrPb blocks.
- It takes some time to process it because it needs 4 iterative statement.

(4) Mimic quantization process by selecting part of the diagonal rows.
- The more it selects diagonal rows, the more the image accuracy is increased.
- (4) is the only part with loss.

(5) Inverse_DCT operation to recover YPrPb blocks.
- Same as (3). This is lossless transformation.

(6) Convert the YPrPb blocks into RGB blocks.
- Same as (2), This is inverse function of (2).

(7) Display the image.
- Casting double to byte
- Firstly, round the double values, and then make them byte type.