Stevens Institute of Technology
Department of Electrical and Computer Engineering

CpE 462 Introduction to Image Processing and Coding

Spring Semester 2022 Final Exam, May 12, 6:00 – 10:00 PM

Instructions:

- Please provide necessary **intermediate steps** in your work. You will get zero credit if you only provide the final result without necessary steps.
- **All** calculations are to be done by **hand**, with the help of a calculator. Computer is only allowed for viewing lecture notes and course materials.
- Sign the following statement

I pledge on my honor that I have abided by the Stevens honor code

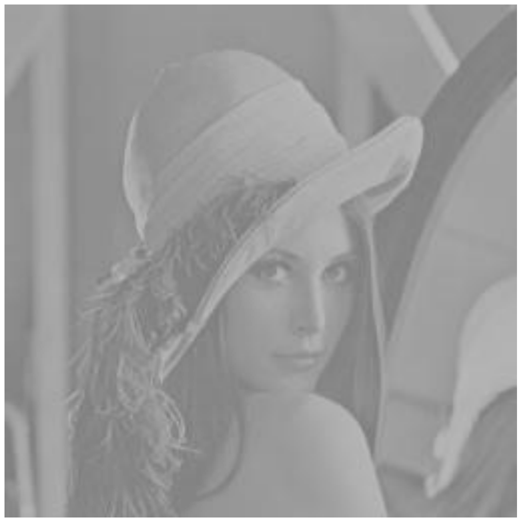
[illegible]

Name (print): Alex Gaskins

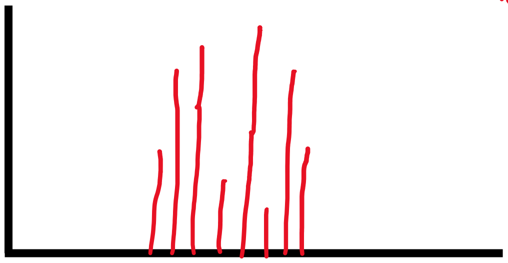
Last four digits of your Student ID: 5143

Problem 1: (20 points) A slightly distorted 256 256 LENA image is shown below.
1) Sketch the histogram of this distorted image
2) Explain whether the nth power or the nth root function can enhance this image.
3) Sketch a contrast stretching function that can enhance this image

original image

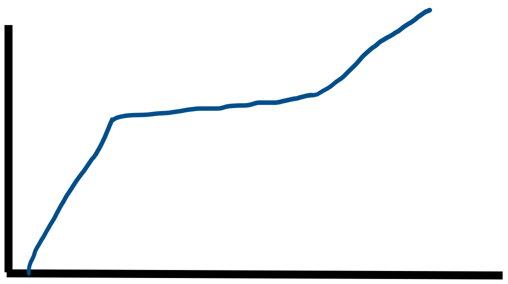

distorted image

1) Low Contrast Image



2) The image is brighter, so the lower amplitudes should be mapped. Thus, the nth power function should be used to enhance visibility

3)



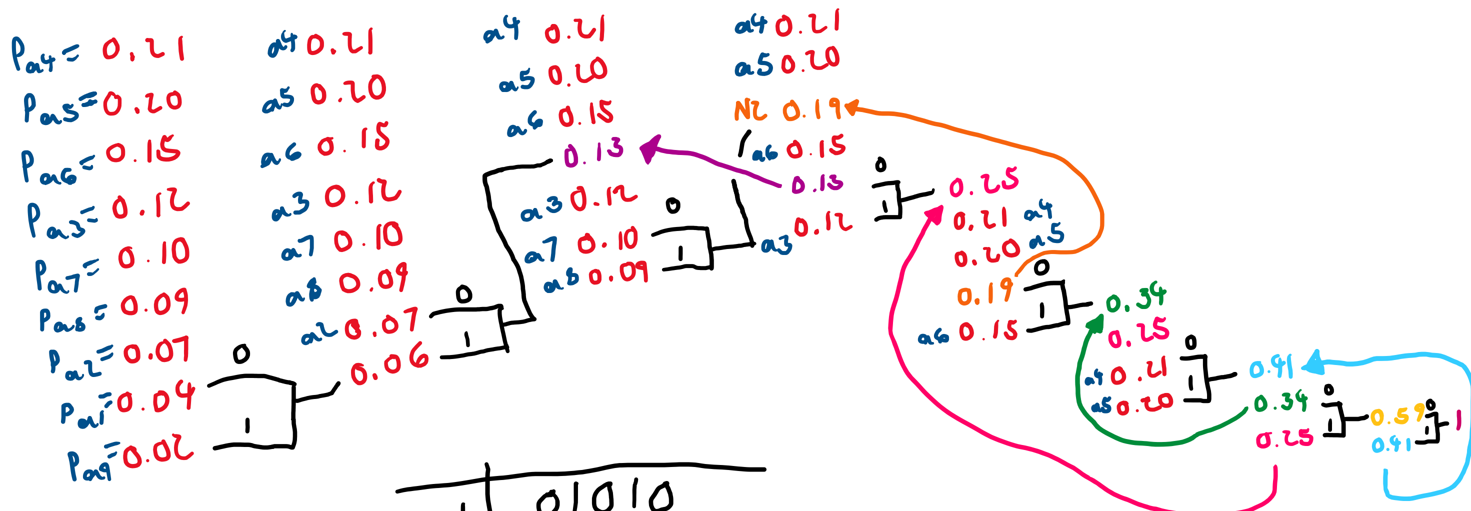
Problem 2 (20 points) Given an alphabet A={a$_1$, a$_2$, a$_3$, a$_4$, a$_5$, a$_6$, a$_7$, a$_8$, a$_9$} with probabilities P(a$_1$)=0.04, P(a$_2$)=0.07, P(a$_3$)=0.12, P(a$_4$)=0.21, P(a$_5$)=0.20, P(a$_6$)=0.15, P(a$_7$)=0.10, P(a$_8$)=0.09, P(a$_9$)=0.02.

1) Compute the entropy of this data source **(in Log Base 2)**;

$$\sum\left[-\log_2(P(x))\right]P(x) = \left(\left[-\log_2(0.04)\right]0.04\right) + \left(\left[-\log_2(0.07)\right]0.07\right) + \left(\left[-\log_2(0.12)\right](0.12)\right) + \left(\left[-\log_2(0.21)\right]0.21\right) + \left(-\log_2(0.20)\left[0.20\right]\right) + \left(\left[-\log_2(0.15)\right]0.15\right) + \left(\left[-\log_2(0.10)\right]0.10\right) + \left(-\log_2(0.09)\left[0.09\right]\right) + \left(-\log_2(0.02)\left[0.0\right.\right.$$

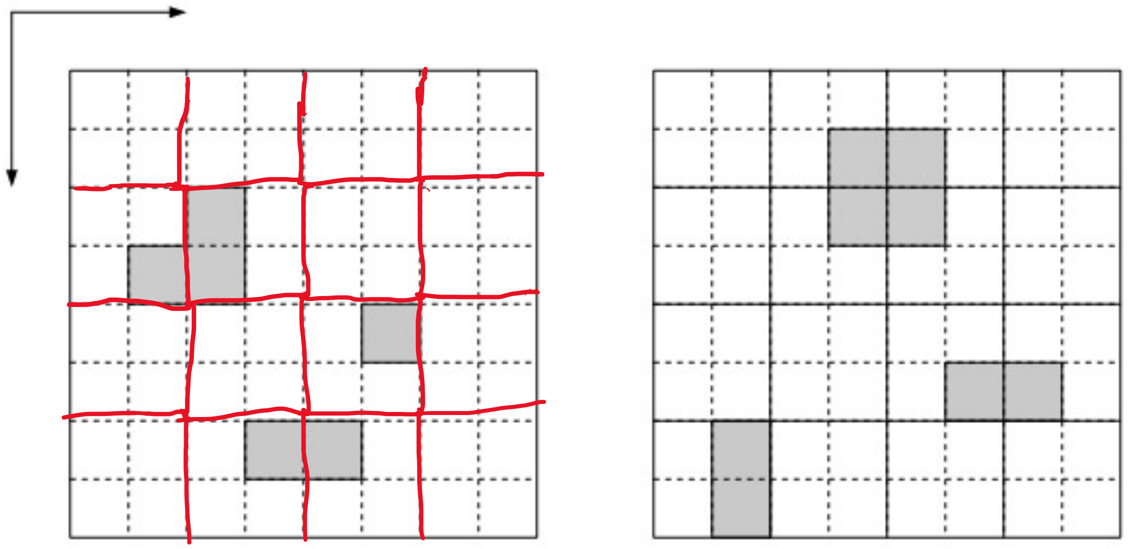
$$= \boxed{2.93}$$

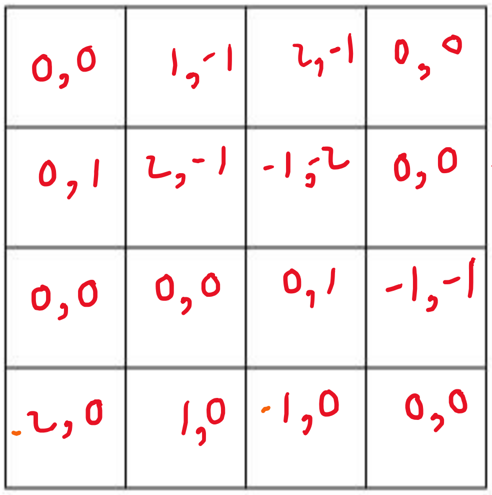
2) Design a Huffman code for this data source. **(Show your steps)**



a1	01010
a2	0100
a3	011
a4	10
a5	11
a6	001
a7	0000
a8	0001
a9	01011

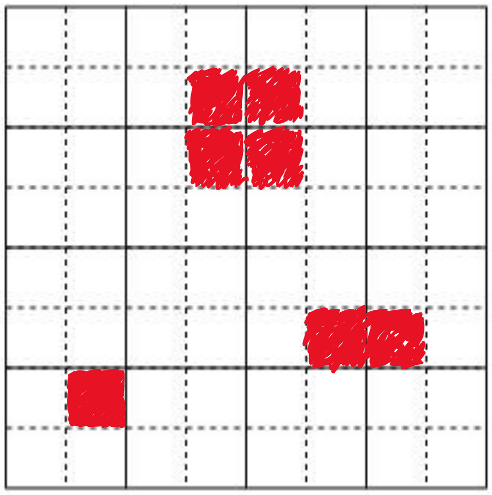
Problem 3 (20 points) Based on the motion compensated prediction used in MPEG, generate the prediction frame, the motion vectors, and the difference frame for the current frame as shown. Assume each box represents a pixel, each macro-block is of 2□2 pixels, the white boxes have value of zero (**0**), and the gray boxes have value of one (**1**). (Note: motion vectors should point from the location of the best match block in the reference frame to the location of the current macro-block in the current frame.)


reference frame current frame

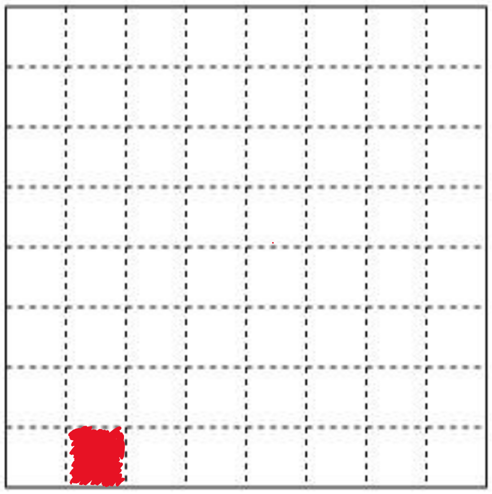


0,0	1,-1	2,-1	0,0
0,1	2,-1	-1,-2	0,0
0,0	0,0	0,1	-1,-1
-2,0	1,0	-1,0	0,0

motion vector field



prediction frame



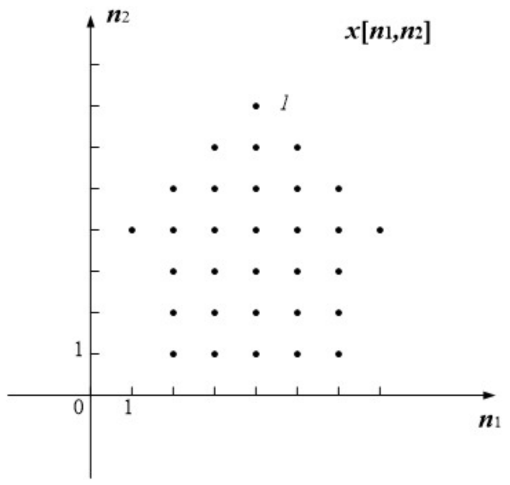
difference frame

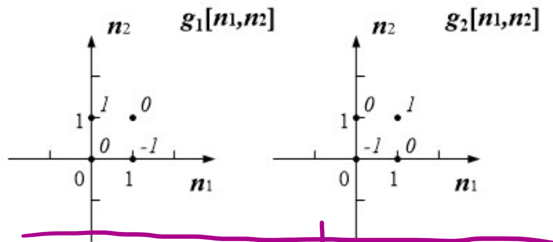
Problem 4 (20 points) Given a 2-D signal $x[n_1, n_2]$ and the Roberts edge detector $g_1[n_1, n_2]$ and $g_2[n_1, n_2]$ as shown, all dark pixels shown in $x[n_1, n_2]$ have value of **1.**

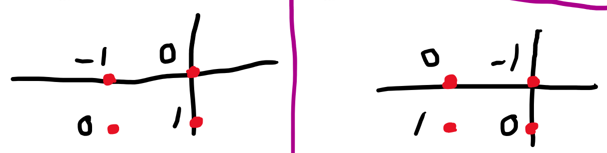
1) Calculate the 2-D convolution of $y_1[n_1, n_2] = x[n_1, n_2] ** g_1[n_1, n_2]$,

2) Calculate the 2-D convolution of $y_2[n_1, n_2] = x[n_1, n_2] ** g_2[n_1, n_2]$,

3) Sum up the absolution values of these two outputs as $y[n_1, n_2] = |y_1[n_1, n_2]| + |y_1[n_1, n_2]|$, which should become the detected edge image. **(Important: show your steps in all calculations.)**

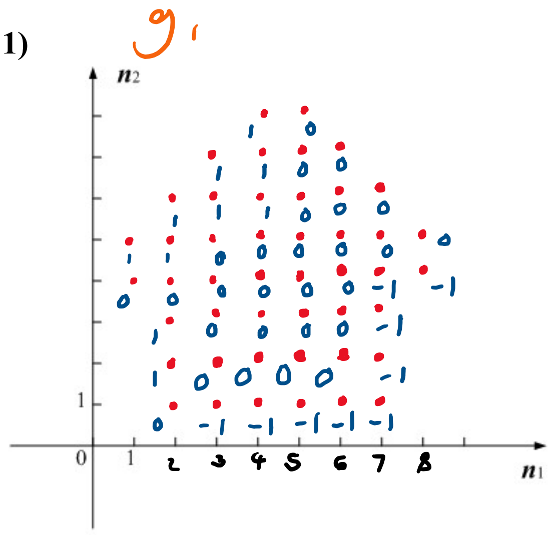






g_1 mask $\qquad g_2$ mask

1)



(2,1) = 1×0 = 0

(3,1) → (7,1) = −1×1 = −1

(2,2) = 1×1 = 1

(3,2) → (6,2) = −1×1 + 1×1 = 0

(7,2) = −1×1 = −1

(1,4) = 1×0 = 0 $\quad$ (2,4) = −1×1 + 1×1 = 0

(3,4) → (6,4) = −1×1 + 1×1 = 0

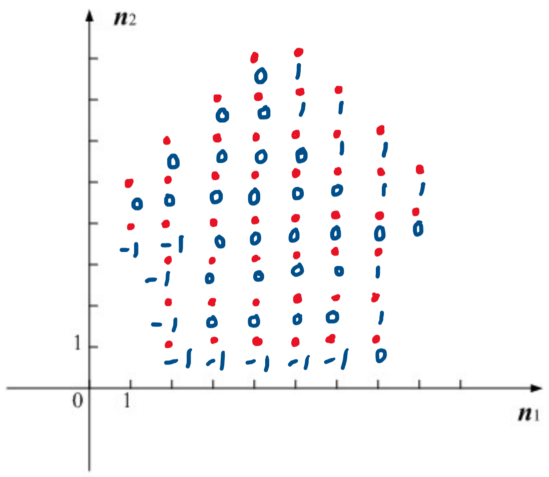
(3,5) → (7,5) = −1×1 + 1×1 = 0

(2,6) = 1×1 = 1

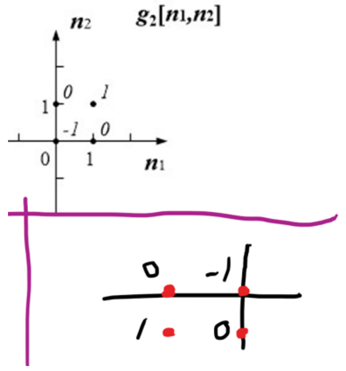
(5,6) → (7,6) = −1×1 + 1×1 = 0

g_2

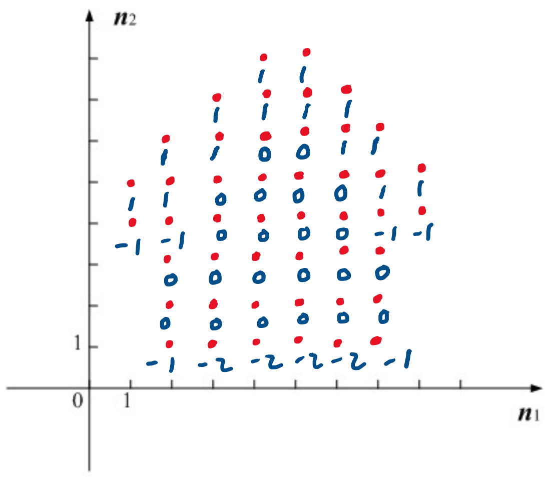
2)



(Flip g_1 result)

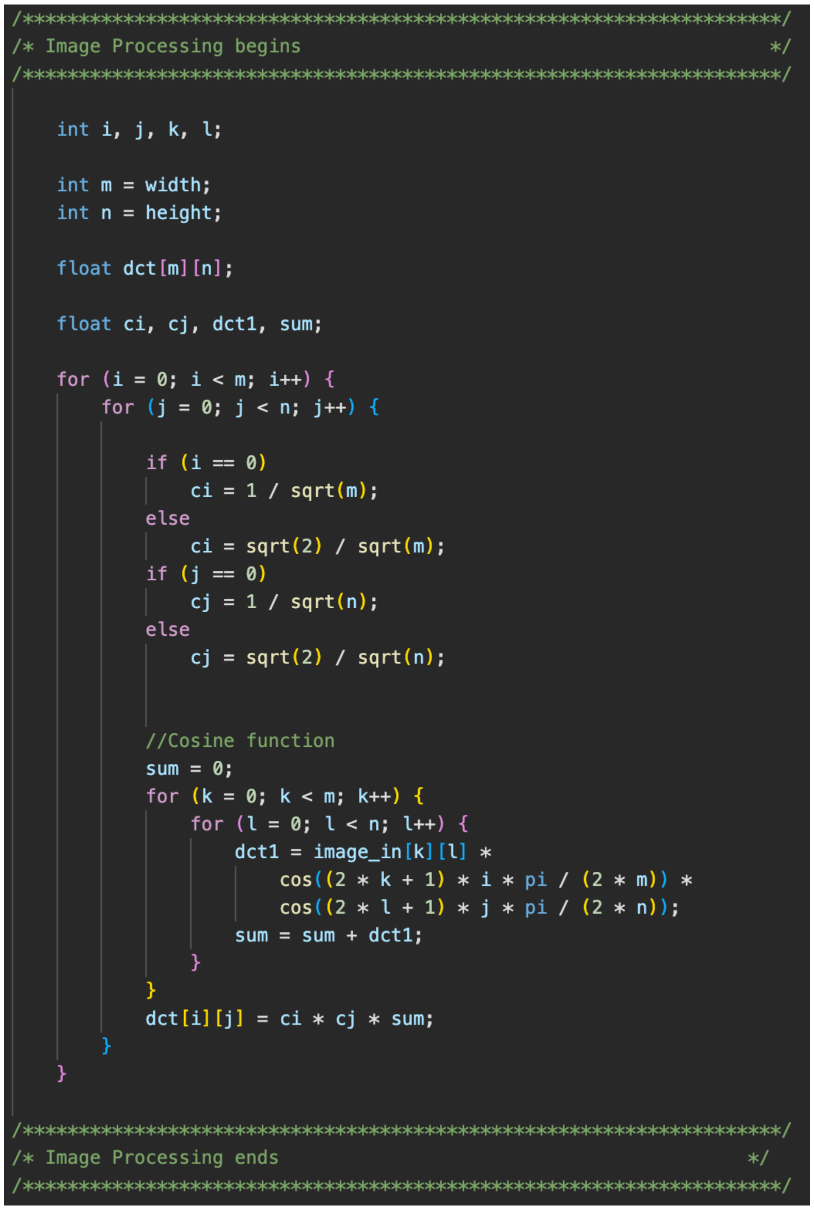


3)



Problem 5 (20 points) Write a segment of C/C++ routine to calculate the 2□2 2D DCT of an input image. The 2□2 2D DCT is calculated on every non-overlapping 2□2 image block inside the image. Assume you are using the "**`imageproc.cpp`**" code structure, and your work should be placed between the comments "**`image processing begins`**" and "**`image processing ends`**". Assume the input image is stored in the 2-D array **`image_in[][]`**, the output image should be saved in the 2-D array **`image_out[][]`**.
Hint: 2D DCT transform coefficients can be pre-calculated, referring to Homework 4.
Notes:

1. Try to write your code as efficient as possible, which will be graded accordingly;
2. If you can not complete this code in C/C++, at least you may provide a pseudocode to outline your procedure, and receive some partial credits.



```
/***********************************************************************/
/* Image Processing begins                                             */
/***********************************************************************/

    int i, j, k, l;

    int m = width;
    int n = height;

    float dct[m][n];

    float ci, cj, dct1, sum;

    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {

            if (i == 0)
                ci = 1 / sqrt(m);
            else
                ci = sqrt(2) / sqrt(m);
            if (j == 0)
                cj = 1 / sqrt(n);
            else
                cj = sqrt(2) / sqrt(n);


            //Cosine function
            sum = 0;
            for (k = 0; k < m; k++) {
                for (l = 0; l < n; l++) {
                    dct1 = image_in[k][l] *
                        cos((2 * k + 1) * i * pi / (2 * m)) *
                        cos((2 * l + 1) * j * pi / (2 * n));
                    sum = sum + dct1;
                }
            }
            dct[i][j] = ci * cj * sum;
        }
    }

/***********************************************************************/
/* Image Processing ends                                              */
/***********************************************************************/
```