תרגיל בית מס' 2

אלכסנדר שנדר

328626114

שאלה 2

א. אני מציע את השיטה הבאה. נגדיר פילטר 3X3, הנראה באופן הבא:

$$filter = \begin{bmatrix} \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & -\frac{1}{2} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -4 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

המשתמד צריך להגדיר סף (נניח, 5 ערכי רמת אפור). לפני שימוש בפילטר נרפד את התמונה בערכי רמות אפור של הפיקסלים שנמצאים בקצוות. לאחר שימוש בפילטר, אנו נעביר את הסף על כל התמונה, וכל הערכים שהיו מעל הסף, יקבלו ערך 1, ומתחת לסף – 0 (בעצם, בינריזציה של התמונה). כל הפיקסלים השחורים – אלה שכן ערכם הוכפל פי שתיים. הרעיון של הפילטר הוא שהוא מחסיר את חצי של הערך של הפיקסל מהממוצע של ערכי הפיקסלים שמסביב. אם הפיקסל הוא תקול, נקבל מספר שקרוב ל-0. אם הפיקסל אינו תקול – נקבל בערך חצי מהערך של הפיקסל. ואז בעזרת פעולת סף נעביר רק את הפיקסלים שקרובים ל-0.

סף גדול – יעביר גם את הפיקסלים שלא תקולים, אז שערכם קטן מדי כדי ליצור הפרש גדול (לאחר פילטר).
סף נמוך מדי – יכול לפספס את הפיקסלים התלוחים. אני מניחים שיש רעש בתמונה כמובן. דוגמא (קוד מטלב בסוף המסמך):

1. נניח שהתמונה הנתונה הינה:

10x10 uint8

	1	2	3	4	5	6	7	8
1	65	63	58	63	61	63	60	60
2	65	64	65	66	60	65	62	64
3	64	62	63	60	59	57	66	60
4	65	57	60	59	63	65	61	59
5	58	65	58	65	122	61	66	66
6	61	57	59	64	64	62	58	63
7	62	61	64	61	58	59	60	60
8	65	57	56	57	63	57	126	57
9	63	60	63	62	57	65	57	63
10	60	64	62	58	64	57	63	62

ניתן לשים לב שיש לתמונה רעש, וכמו כן שני פיקסלים תקולים

2. נרפד את התמונה מכל הכיוונים לפני הפעלת פילטר. הערכים הם של הערך של התמונה בקצה.
3. נפעיל פילטר. נקבל את התמונה הבאה:

10x10 uint8

	1	2	3	4	5	6	7	8
1	32	31	34	30	32	30	32	30
2	32	31	30	28	32	28	31	29
3	31	32	30	32	32	34	29	32
4	29	33	31	39	37	37	32	33
5	32	27	32	36	2	40	29	27
6	30	32	32	37	37	38	33	28
7	30	30	27	30	32	39	38	38
8	29	33	33	32	28	40	0	40
9	30	31	28	29	32	36	40	38
10	32	30	30	32	28	33	29	31

נשים לב שהערכים של התאים התקולים הם נמוכים וקרובים ל-0.

4. נגדיר סף שגודלו 5. כל התאים שהם מתחת ל-5 יהיו 0, כל התאים מעל – יהפכו להיות 1. בעצם, בינריזציה. נקבל:

10x10 logical

	1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1
5	1	1	1	1	0	1	1	1
6	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	0	1
9	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1

ניתן לראות שאכן קיבנו אלגוריתם שיודע למצא פיקסלים תקולים **מתוך תמונה בודדת**.

ב. עבור מקרה בו יש לנו שתי תמונות, יותר קל לנו למצוא את הפיקסלים התקולים. בהנחה שאין תזוזה של עצמים בין שתי תמונות כמובן. האלגוריתם יחשב את ההפרשים של הפיקסלים בין שתי תמונות. לאחר מכן יעביר סף (מוגדר ע"י משתמש) ויעשה בינריזציה לתמונה. חשוב – כאשר מחסירים שתי תמונות, עלינו להעביר את התמונה ל-double, אחרת החסרת פיקסל עם רמת אפור גבוהה מפיקסל עם רמת אפור נמוכה יתן לנו 0, מה שיגרום לתוצאות שגויות. נגדים. כל התהליך בוצע במטלב בסוף המסמך.

1 – initial image A

	1	2	3	4	5
1	58	58	59	63	60
2	60	65	120	58	61
3	59	66	62	57	57
4	65	60	59	59	59
5	60	57	62	59	128

2 – initial image B

	1	2	3	4	5
1	112	58	58	60	64
2	65	61	61	66	57
3	63	66	62	56	59
4	61	61	63	65	59
5	62	61	60	65	63

3 – differentiate 2 matrixes in 'double' format, take absolute value

	1	2	3	4	5
1	0.2118	0	0.0039	0.0118	0.0157
2	0.0196	0.0157	0.2314	0.0314	0.0157
3	0.0157	0	0	0.0039	0.0078
4	0.0157	0.0039	0.0157	0.0235	0
5	0.0078	0.0157	0.0078	0.0235	0.2549

4 – binarize the image based on the threshold

	1	2	3	4	5
1	1	0	0	0	0
2	0	0	1	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	1

ג. מיקום הפיקסלים התקולים ידוע. התמונה היא בסידור של העמודה. f^{cs} הינה תמונה המעוותת, עלינו למצוא מצטירה A כך שנוכל לשחזר את התמונה המקורית g^{cs}:

$$g^{cs} = Af^{cs}$$

ידוע שעל מנת לתקן את הפיקסל התקול, עלינו להקטין את גודלו פי 2. הפתרון הינו לבנות מטירצה A אלכסונית לפי שלבים הבאים:

1. גודל תמונה מקורית היא X על Y.
2. נניח, שיש פיקסל תקום במיקום [x,y].
3. לאחר סידור של התמונה בעמודה (גודלה כעת $[X \cdot Y, 1] = [X_2, 1]$), מיקום של הפיקסל התקול נהיה:
$$[x, y] \rightarrow [(y-1) \cdot X + x, 1] = [x_2, 1]$$
4. נבנה מטריצה אלכסונית A, כלל הערכים באלכסון שלה הם 1
5. נקח כל פיקסל תקול, ונמיר את הערך במטריצה A האלכסונית עבור כל פיקסל כזה. נניח עבור פיקסל שלנו:
$$A[x_2, x_2] = 0.5$$
6. נפעל לפי הנוסחה המקורית, נהפוך את המטריצה שוב למימדים המקוריים, ונקבל g^{cs} מקורית.

דוגמא (קוד מטלב בסוף המסמך):

1 – initial image

	1	2	3	4
1	61	63	63	57
2	60	64	63	122
3	62	59	58	66

2 – make column – failed pixel at [11,1]

	1
1	61
2	60
3	62
4	63
5	64
6	59
7	63
8	63
9	58
10	57
11	122
12	66

4 – get fixed image

3x4 uint8

	1	2	3	4
1	61	63	63	57
2	60	64	63	61
3	62	59	58	66

3 – generate A matrix. Index [11,11] is 0.5

A

12x12 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	1	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	0	0	0	0	1	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	1	0	0
11	0	0	0	0	0	0	0	0	0	0	0.5000	0
12	0	0	0	0	0	0	0	0	0	0	0	1

ד. עלינו למצוא מטריצה שתמיר את תמונה בנסידור-שורה לסידור עמודה. כאן נשתמש בעובדה שלתמונה המקורית כמות שורות ועמודות היא זהה, אחרת יש אי וודאות לגבי סידור של העמודות הרצוי. ע"י דוגמא נגדים איך צריכה להיראות מטריצת פעולה שתעשה את העבודה. כל תוצאות שמוצגות כאן הושגו דרך *MATLAB* והפונקציות כתובות על ידי. הסבר קצר – מטריצה C הינה ריבועית, שבה כל הערכים אפסים, חוץ מערכים מסוימים שהם 1. תמיד מיקום [1,1] הינו 1. ואז בכל שורה המיקום של האיבר שהוא 1 זז בעמודות כמות תאים ששווה לאורך הצלע של התמונה. המקרה שלנו – 3. אם התא מגיע ועובר את המטריצה, הוא ממשיך בשורה הבאה עם האינדקס של עמודה של הערך של השארית של חלוקה באורך צלע בריבוע:

1 – initial image

	1	2	3
1	66	66	57
2	58	61	60
3	66	64	65

2 – make row-ordered image

	1
1	66
2	66
3	57
4	58
5	61
6	60
7	66
8	64
9	65

4 – get column ordered image

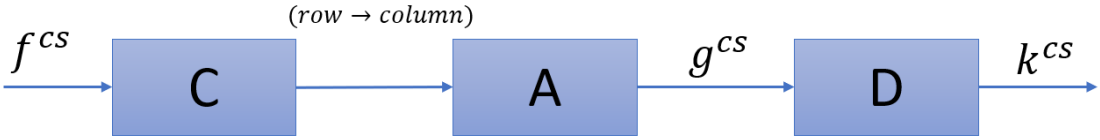
	1
1	66
2	58
3	66
4	66
5	61
6	64
7	57
8	60
9	65

3 – generate C matrix.

9x9 double

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

ה. ידוע כמטריצה D מבצעת התמרת פורייה על התמודה בסידור עמודה. לכן נפעל באופן הבא:



APPENDIX

1. MATLAB CODE

```
%% THIS CODE RELATES TO QUESTION #2

%% Algorithm 1 - single image

im1 = uint8(ones(10,10))+55;
noiseSignal = uint8(rand(10, 10).*10);
% add gauss noise
im1 = im1 + noiseSignal;

im1(5,5) = im1(5,5)*2;
im1(8,7) = im1(8,7)*2;


im2 = padarray(im1,[1 1],'replicate','both');

filter = (1/8)*[1 1 1; 1 -4 1; 1 1 1];

im_filtered = imfilter(im2,filter);
im_filtered_2 = im_filtered(2:end-1,2:end-1); % unpad

threshold = uint8(5);

% use the threshold to binalize the image
im_filtered_3 = imbinarize(im_filtered_2,im2double(threshold));

% plot to see results
imshow(im_filtered_3)


%% Algorithm 2 - two images

% set the threshold

threshold = uint8(20);
image_size = 5;

im1 = uint8(ones(image_size,image_size))+55;
noiseSignal = uint8(rand(image_size, image_size).*10);
% add gauss noise
im1 = im1 + noiseSignal;

% add failed pixels
im1(5,5) = im1(5,5)*2;
im1(2,3) = im1(2,3)*2;

% image 2 has other pixel which fails
im2 = uint8(ones(image_size,image_size))+55;
noiseSignal = uint8(rand(image_size, image_size).*10);
% add gauss noise
im2 = im2 + noiseSignal;

% add failed pixels
im2(1,1) = im2(1,1)*2;

difference_matrix = abs(im2double(im1) - im2double(im2));
difference_bin = imbinarize(difference_matrix,im2double(threshold));
```

```
%% Algorithm 3 - fix the image when the failed pixel coordinates are known

image_size_x = 3;
image_size_y = 3;
failed_pixels{1} = [2 3];

im3 = uint8(ones(image_size_x,image_size_y))+55;
noiseSignal = uint8(rand(image_size_x, image_size_y).*10);
% add gauss noise
im3 = im3 + noiseSignal;



for i=1:numel(failed_pixels)
    im3(failed_pixels{i}(1),failed_pixels{i}(2)) =...
        im3(failed_pixels{i}(1),failed_pixels{i}(2))*2;
end

im3_col_order = reshape(im3,[numel(im3) 1]);

A = create_fixing_matrix([image_size_x, image_size_y], failed_pixels);

im3_fixed_col_order = im2uint8(A*im2double(im3_col_order));
im3_fixed = reshape(im3_fixed_col_order,[image_size_x, image_size_y]);



%% Algorithm 4 - change the column ordered matrix to row ordered


image_size_x = 3;
image_size_y = 3;

im4 = uint8(ones(image_size_x,image_size_y))+55;
noiseSignal = uint8(rand(image_size_x, image_size_y).*10);
% add gauss noise
im4 = im4 + noiseSignal;

im4_row_order = reshape(im4',[numel(im4') 1]);
im4_col_order_check = reshape(im4,[numel(im4) 1]);

C = create_row_to_col_order_matrix(im4_row_order);

im4_col_order = im2uint8(C*im2double(im4_row_order));
```