נושאים נבחרים בעיבוד תמונה -
תרגיל בית 2

מרצה:

ד"ר עופר לוי

מגישים:

יותם אילון 305407330

נדב לרר 302170378

ירדן זוהר 304972656

תאריך הגשה: **04/01/18**

1. זיהוי קווי מתאר בעזרת פילטרים

א. בסעיף זה אנו נדרשים לבנות שתי תמונות (מטריצות בינאריות) ללא שימוש בלולאות. התמונה הראשונה היא תמונה שבמרכזה עיגול לבן, והתמונה השנייה היא תמונה שבמרכזה מתומן. להלן ניתן לראות את ההגדרה המתמטית של שתי הצורות:

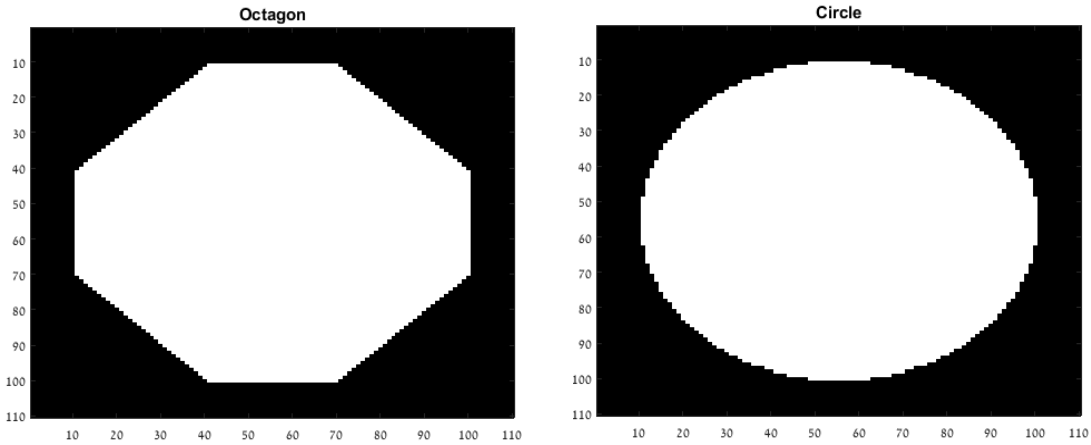
עיגול:

$$\mathrm{Circle}(r,c)=\begin{cases}1 & if \quad \sqrt{(r-55.5)^2+(c-55.5)^2}<45\\ 0 & otherwise\end{cases}$$

מתומן:

$$\mathrm{Octagon}(r,c)=\begin{cases}1 & if \quad 11\le r\le 40 \quad and \quad 52-r\le c\le 59+r\\ 1 & if \quad 41\le r\le 70\\ 1 & if \quad 71\le r\le 100 \quad and \quad r-59\le c\le 170-r\\ 0 & otherwise\end{cases}$$

את המטריצות בנינו באמצעות הפקודה **meshgrid** אשר מרשתת את המטריצות בשני הצירים האורכי והאנכי בספרות מ-1 ועד 110 (סה"כ מטריצה 110X110). שתי המטריצות המתקבלות הן מגדירות גריד דו ממדי, אשר באמצעותו ניתן לממש את הפונקציות המוצגות לצור יצירת הצורות באמצעות תנאים בוליאניים. באיור 1.1 ניתן לראות את הצורות שהתקבלו.

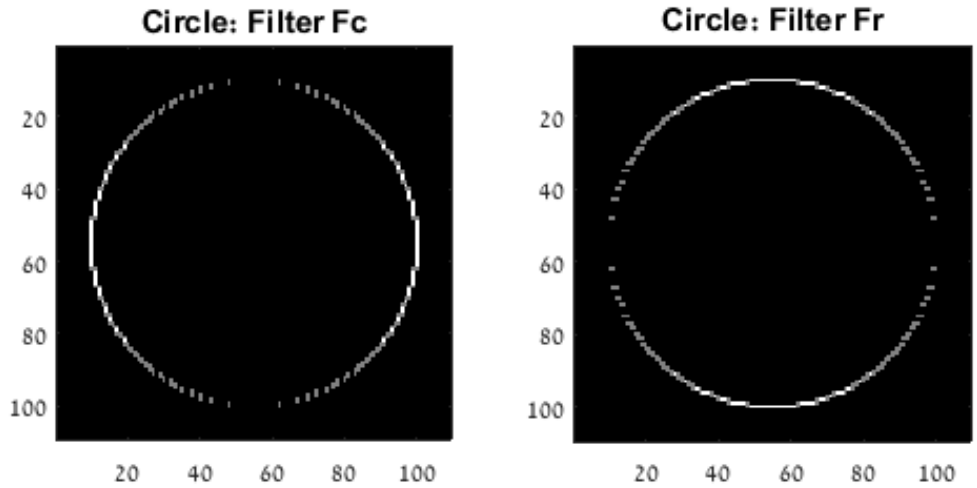


איור 1.1: תמונת העיגול (מימין) והמתומן (משמאל).

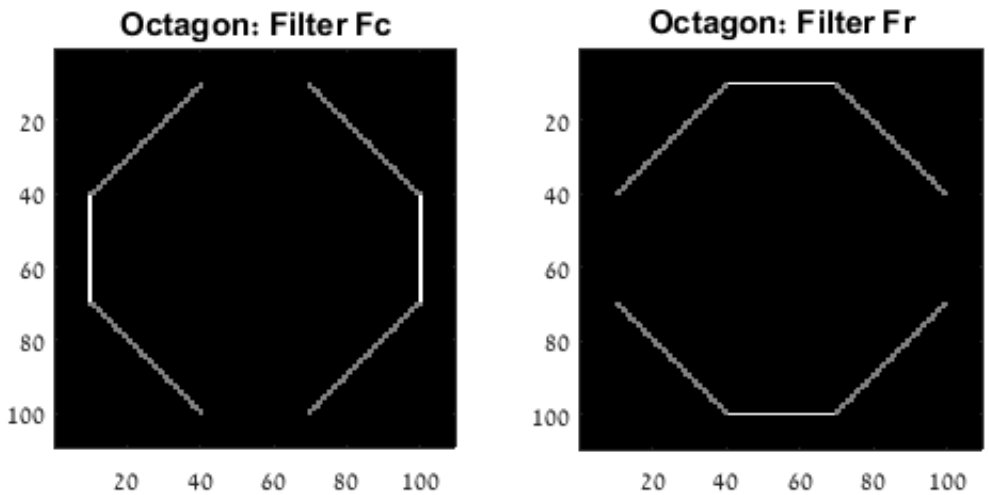
ב. בסעיף זה נדרשנו להעביר על שתי התמונות את הפילטרים הבאים:

$$F_r = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix},\ F_c = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

הפילטרים האלו משמשים לזיהוי קווי מתאר (בפועל הם מזהים שינוי חד בערכי הפיקסלים). הפילטר F_c מזהה קווים אנכיים בעוד שהפילטר F_r מזהה קווים אופקיים. את הפילטרים הפעלנו על האיורים מסעיף קודם באמצעות הפונקציה **conv2** שהיא מבצעת את אותה הפעולה כמו הפונקציה **filter2** פרט לאופן הסריקה שלהן. באיורים 1.2 ו-1.3 ניתן לראות את התמונות המתקבלות מן לאחר הפעלת כל פילטר על כל תמונה. על הערכים שהתקבלו לאחר הרצת הפילטרים הפעלנו את הפונקציה **abs** על מנת לקבל את גודלם מכיוון שהסימן נקבע מכיוון המעבר – משחור ללבן או להיפך. בפועל מעניין אותנו השינוי בצבע פיקסלים בלבד ולכן נדרש ערך מוחלט.

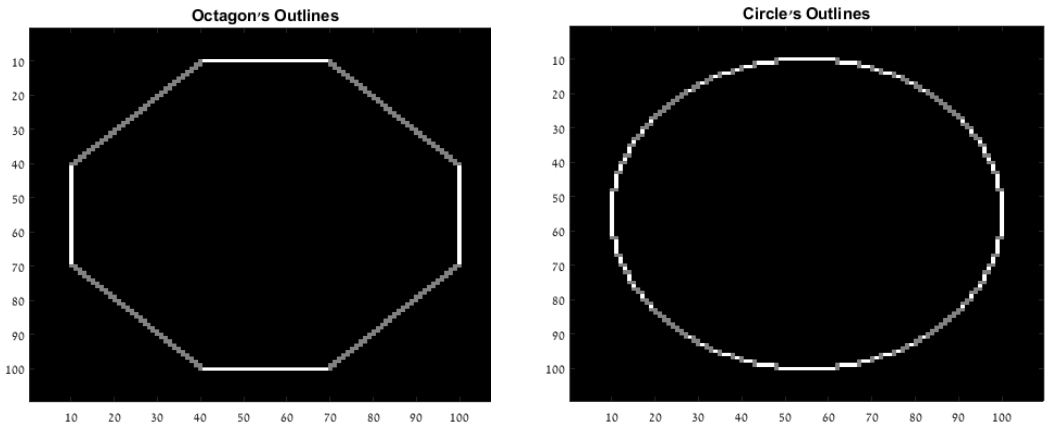


<u>איור 1.2:</u> הפעלת הפילטרים על המעגל. **Fr** מימין ו-**Fc** משמאל.



<u>איור 1.3:</u> הפעלת הפילטרים על המתומן. **Fr** מימין ו-**Fc** משמאל.

ג. בסעיף זה השתמשנו עבור כל צורה בשתי התוצאות שהתקבלו מהשימוש בפילטרים **Fr** ו-**Fc** על מנת לקבל את קווי המתאר של הצורה כולה. אופן המימוש היה לקיחת הערך המקסימלי המתקבל בכל פיקסל מכל פילטר. באופן זה מתקבלים הקווים האורכיים והרוחביים של הצורות. ניתן לראות כי קווי שהיו בזווית של 45 מעלות, התקבלו בערך נמוך יותר באופן יחסית משמעותי, מכיוון שהם נפלו הכי רחוק שניתן מהכיוונים של שני הפילטרים. זה מראה לנו על כך שהשימוש בפילטר זה הוא מוגבל ולא מאפשר הבחנה בקווי מתאר בתמונות מורכבות יותר. באיור 1.4 ניתן לראות את שילוב הפילטרים עבור שתי הצורות.



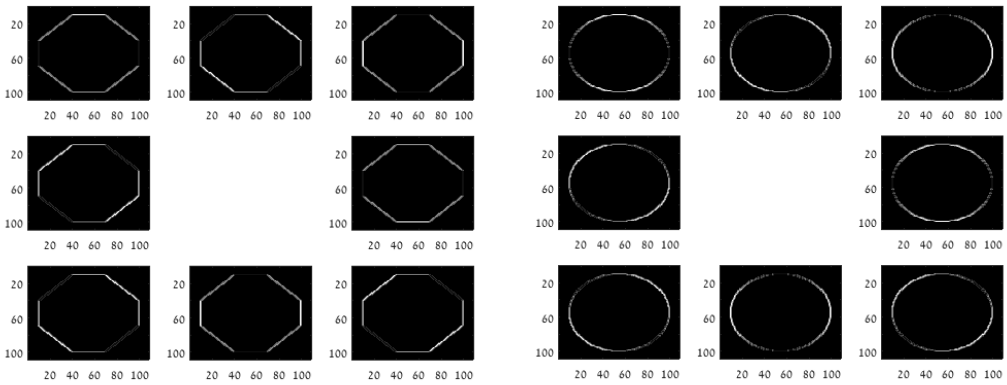
איור 1.4: שילוב הפילטרים עבור המעגל (מימין) והמתומן (משמאל).

ד. בסעיף זה נדרשנו לחזור על סעיפים ב' ו-ג', אך הפעם בשימוש במסכות המצפן של קירש, במקום בפילטרים הפשוטים מסעיף קודם. מסכות המצפן של קירש הן הפילטרים הבאים:

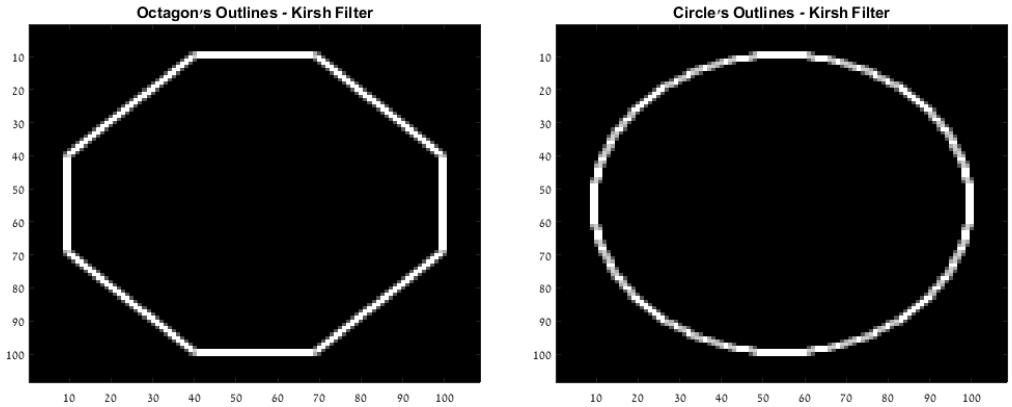
$$G_1=\begin{bmatrix}5&5&5\\-3&0&-3\\-3&-3&-3\end{bmatrix}\quad G_2=\begin{bmatrix}-3&5&5\\-3&0&5\\-3&-3&-3\end{bmatrix}\quad G_3=\begin{bmatrix}-3&-3&5\\-3&0&5\\-3&-3&5\end{bmatrix}\quad G_4=\begin{bmatrix}-3&-3&-3\\-3&0&5\\-3&5&5\end{bmatrix}$$

$$G_5=\begin{bmatrix}-3&-3&-3\\-3&0&-3\\5&5&5\end{bmatrix}\quad G_6=\begin{bmatrix}-3&-3&-3\\5&0&-3\\5&5&-3\end{bmatrix}\quad G_7=\begin{bmatrix}5&-3&-3\\5&0&-3\\5&-3&-3\end{bmatrix}\quad G_8=\begin{bmatrix}5&5&-3\\5&0&-3\\-3&-3&-3\end{bmatrix}$$

באופן דומה העברנו את שמונת הפילטרים על שתי התמונות ולקחנו את הערך המקסימלי מכל פילטר לקבל קווי המתאר. באיור 1.5 ניתן לראות את תוצאות שמונת הפילטרים על כל אחת מן הצורות ובאיור 1.6 ניתן לראות את קווי המתאר לאחר לקיחת הערך המקסימלי.



איור 1.5: תמונות המעגל (מימין) והמתומן (משמאל) לאחר הפעלת מסכות המצפן של קירש.



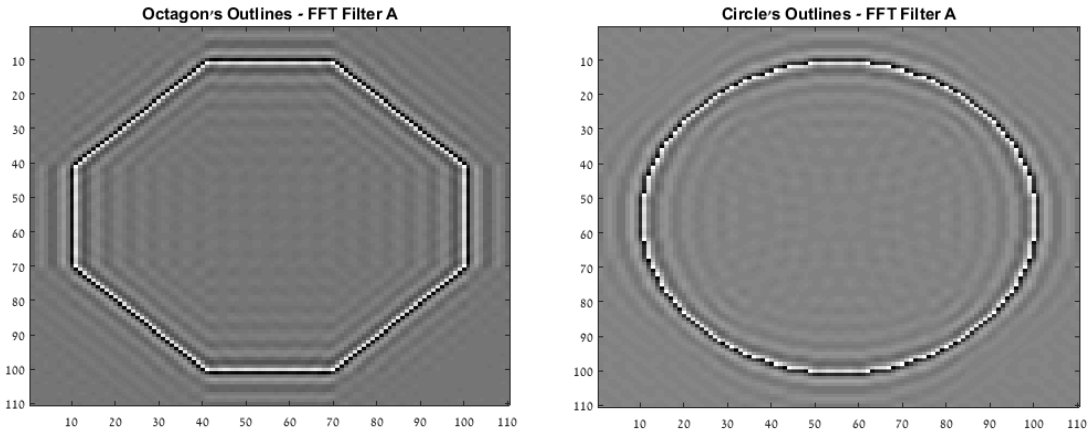
איור 1.6: ערכי המקסימום לאחר הפעלת מסכות המצפן על המעגל (מימין) והמתומן (משמאל).

בהשוואה לפילטרים בסעיפים קודמים, ניתן לראות שמסכות המצפן של קירש מכסות כיוונים נוספים שהפילטרים הפשוטים לא כיסו. עבור המתומן הוא פעל באופן כמעט מושלם מכיוון שצלעות המתומן הן בדיוק בכיווני המסכות, בעוד שעבור המעגל ניתן עדיין לראות חלקים שהם מעט אפורים (ולא לבנים לחלוטין), מכיוון שמעגל מכסה את הכיוונים האפשריים ביחס לרזולוציית התמונה.

ה. בסעיף זה נדרשנו לבנות פילטר שעובד על מקדמי פורייה של התמונה לאחר ביצוע התמרת פורייה לתמונה. את סעיף זה מימשנו בשתי דרכים אפשריות:

דרך א': ריבוע בינארי

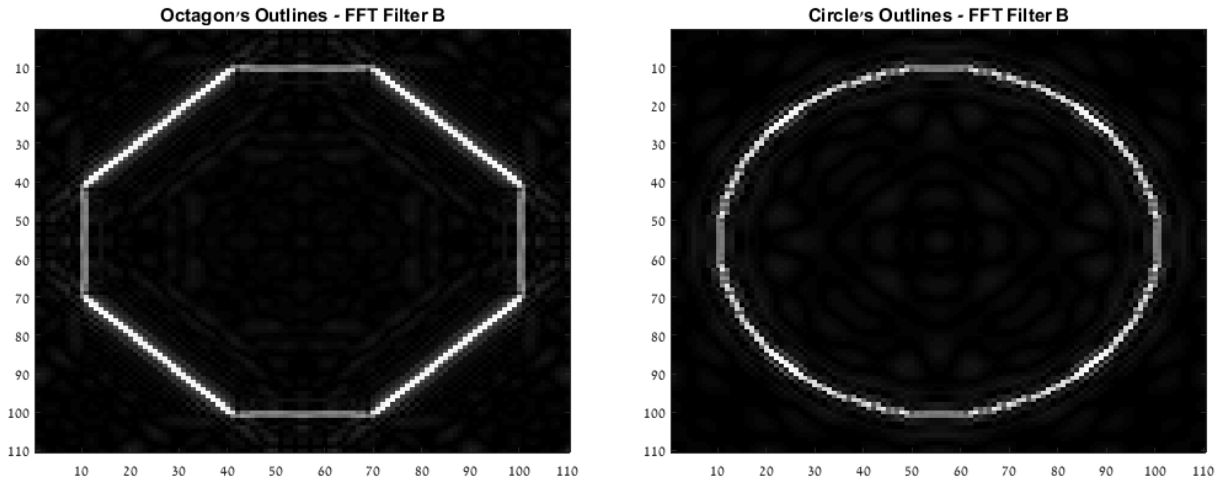
בשיטה זו בנינו פילטר שמתפקד כ- **High Pass Filter**. ידוע כי שינויים חדים בתמונה יבואו לידי ביטוי באופן בולט יותר בתדרים הגבוהים במרחב פורייה. לכן ביצענו לתמונות התמרת פורייה לתמונות של המעגל והמתומן, וביצענו הזזה (**fftshift**) כך שהתדרים הנמוכים יתקבלו במרכז. בנינו פילטר בינארי בממדי התמונות (110**X**110) כך שהמעטפת שלו מורכבת מהספרה '1' ומרכזו מהספרה '0'. האפסים במרכז מסודרים בריבוע סביב המרכז שאורך צלעותיו נמצא מביצוע מספר ניסיונות ולבסוף התקבל בערך $2 \cdot a = 42$. את הפילטר כפלנו בכפל איבר-איבר במטריצה מקדמי פורייה המוזזת, ובכך איפסנו את התדרים במרכז (הנמוכים) והשארנו את התדרים על המעטפת (הגבוהים). לאחר מוכן בוצע הזזה חזרה והתמרה הפוכה לקבלת התמונה המתוקנת. בגלל הפעלת הפילטר ההתמרה ההפוכה החזירה ערכים מרוכבים עם ערכים מדומים קטנים מאוד, ולכן על מנת להציג את התמונה לקחנו את גודל הערכים. באיור 1.7 ניתן לראות את התמונות שהתקבלו לאחר הפעלת הפילטר.



<u>איור 1.7:</u> פילטר א' במרחב פורייה עבור המעגל (מימין) והמתומן (משמאל).

דרך ב': פילטר גאוסיאני עם ערך סף

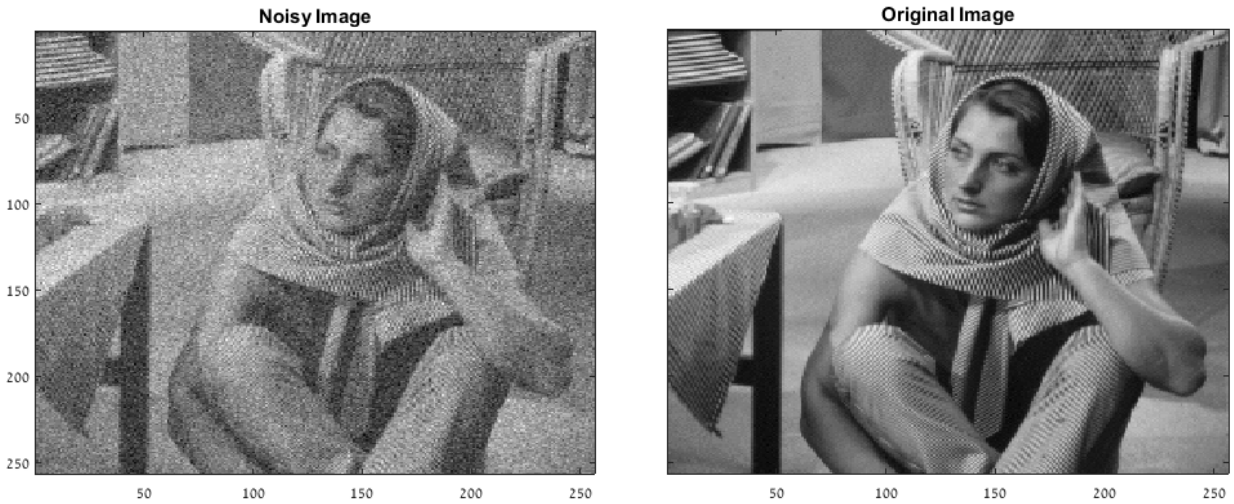
בשיטה זו קבענו ערך סף שתלוי בערך הממוצע של הערך המוחלט של התמרת פורייה של כל תמונה. באמצעות מטריצה בינארית, סיננו כל ערך אשר גבוה מערך הסף שנמצא מתוך ביצוע מספר הפעלות של הפילטר לקבל ערך אולטימטיבי $Threshold = 4.5 \cdot mean(:)$. על הערכים שנשארו הפעלנו פילטר גאוסיאני דו-ממדי (בפועל זהו הגאוסיאן הופכי מכיוון שהאקספוננט הוא חיובי), על מנת להגביר את התדרים הגבוהים לפי התפלגות גאוסיאנית. השונות של הגאוסיאן נקבעה גם היא מביצוע מספר הפעלות של הפילטר על התמונות, ועבור התמונות של המעגל והמתומן התקבלה שונות של $\sigma^2 = 35^2$. את התוצאות ניתן לראות באיור 1.8.



איור 1.8: פילטר ב' במרחב פורייה עבור המעגל (מימין) והמתומן (משמאל).

2. ניקוי רעשים מתמונה

בשלב הראשון נדרשנו לפתוח את תמונתה של ברברה ולהוסיף לה רעש לבן עם סטיית תקן של 20. באיור 2.1 ניתן לראות את התמונה לפני ואחרי הוספת הרעש.



איור 2.1: תמונתה המקורית (מימין) והרועשת (משמאל) של ברברה.

לאחר מכן נדרשנו לנקות את הרעש באמצעות ארבעה פילטרים שונים, כאשר את כל אחד מהם הפעלנו בחלונות בגדלים משתנים :

א. פילטר מיצוע פשוט

פילטר מיצוע פשוט, מבצע ממוצע לכל התאים בכל חלון. על מנת שזה התקיים דרוש שסכום כל התאים בחלון יהיה שווה ל-1 ולכן כל פילטר מוגדר באופן הבא :

$$h_{\mathrm{Re}}(k) = \frac{ones(k)}{k^2}$$

פילטר זה נוסה ב-9 גדלים שונים $(2X2-10X10)$ ובאיור 2.2 ניתן לראות את התוצאות שהתקבלו מהפעלת הפילטרים.

<u>איור 2.2:</u> פילטר מיצוע פשוט עבור חלונות בגדלים משתנים.

ניתן לראות כי ככל שמגדילים את גודל החלון הרעש הולך ונעלם, אך עם זאת משלמים במחיר של פגיעה בחדות התמונה, אשר הולכת ודועכת. לכן יש למצוא גודל חלון אופטימלי עבורו הפגיעה בחדות הינה מינימאלית בעוד שניקוי הרעשים הוא מקסימלי. עבור תמונה זו ניתן לראות שגודל החלון האופטימלי הוא בין 4 ל-6.

ב. פילטר מיצוע גאוסיאני

העיקרון של פילטר גאוסיאני הוא שערכי החלון הם בהתפלגות גאוסיאנית סביב מרכז החלון, בניגוד לפילטר מיצוע פשוט שבו ההתפלגות הינה אחידה. בדומה לפילטר המיצוע הפשוט, גם כאן סכום על איברי החלון הוא 1. לצורך בניית הפילטרים, נבנה גריד דו-ממדי בגודל החלון (כאשר יש לעשות הפרדה בין חלון בגודל זוגי לבין אי-זוגי), ומתוך הגריד שהתקבל חושבה ההתפלגות הגאוסיאנית, כאשר ערכה של סטיית התקן היה שווה לשמינית מרוחב החלון, לפי דרישת התרגיל. חישוב ההתפלגות בוצע באופן הבא:

$$h_{Ga}(k) = e^{-\left(\frac{r^2+c^2}{2\cdot\sigma^2}\right)}$$

כאשר r ו-c הם הגריד שנפרש בהתאם לרוחב החלון k. השונות בכל חלון הוגדרה כך $\sigma = k/8$.

במטלב ישנה פונקציה שבונה מטריצות בגדלים שונים בהתאם להתפלגות הרצויה. בוצעה השוואה בין החלונות שבנינו לבין אלו שמתקבלים מהפונקציה של המטלב ונמצאה התאמה מלאה! הפונקציה במטלב היא: `h = fspecial('gaussian', hsize, sigma)`. באיור 2.3 ניתן לראות

את התוצאות עבור הפעלת הפילטר בחלונות בגדלים משתנים.

איור 2.3: פילטר גאוסיאני עבור חלונות בגדלים משתנים.

בהשוואה לפילטר המיצוע הפשוט ניתן לראות כי הפגיעה בחדות הינה נמוכה באופן משמעותי עבור הפילטר הגאוסיאני אך עם זאת גם ניקוי הרעשים הינו עדין יותר. ניתן לראות כי חלונות עבור חלונות בגודל של 9-10 מתקבלת התמונה האופטימלית.

ג. פילטר חציוני

לצורך מימוש פילטר חציוני יש להשתמש בפקודת המטלב **medfilt2** מכיוון שהפעולה של חישוב חציון היא איננה לינארית ולא ניתנת ליישום באמצעות הפונקציות **filter2** או **conv2** במטלב. את התוצאות עבור חלונות הגדלים משתנים בין 2-10 ניתן לראות באיור 2.4.

איור 2.4: פילטר חציוני עבור חלונות בגדלים משתנים.

ניתן לראות כי איבוד החדות הינו מהיר יותר בשימוש בפילטר זה בעוד שניקוי הרעש אינו יעיל יותר מאשר בפילטר המיצוע הפשוט.

ד. Low Pass Filter

בסעיף זה בנינו **Low Pass Filter** שפועל במרחב פורייה של התמונה. בנינו מטריצה בינארית שמרכזה מעגל (כמו בשאלה 1). למטריצה הזו ביצענו כפל איבר-איבר במטריצת התמרת פורייה של התמונה הרועשת וכך ביצענו השתקה של התדרים הרחוקים ממרכז ההתמרה (לאחר ביצוע הזזה). התדרים הרחוקים מן המרכז הם כמובן התדרים הגבוהים, ולכן הפילטר מעביר את התדרים הנמוכים מתדר סף כלשהו. הפעלנו את הפילטר ובמעגל בקטרים משתנים ובאיור 2.5 ניתן לראות את התמונות המתוקנות עבור מעגלים בגדלים משתנים.

<u>איור 2.5:</u> **Low Pass Filter** בגדלים משתנים.

3. דחיסת פורייה בבלוקים

בתרגיל זה נדרשנו לבנות שתי פונקציות שיבצעו דחיסה לתמונות וישמרו את המידע במרחב פורייה. לאחר מכן שתי פונקציות נוספות אשר יתרגמו את המידע ממרחב פורייה בחזרה למרחב הפיקסלים. הדחיסה נדרשה להתבצע בשני אופנים ולשם כך נדרשו שתי פונקציות, האחת אפטיבית והשנייה לא:

3.1. דחיסה לא אדפטיבית

פונקציית הדחיסה הלא אדפטיבית נראית באופן הבא:

```
function [ Fc ] = BFCNA(Im, nb, c)
```

היא מקבלת כקלט את התמונה (Im), את מספר הבלוקים בכל ממד (nb, כלומר סה"כ nb **X** nb בלוקים) ואת יחס הדחיסה (c), והיא מחזירה כפלט וקטור מקדמי פורייה דחוס (Fc). הפונקציה מחלקת תחילה מחשבת את כמות המידע הדרוש לשמור בכל בלוק בשימוש בממדי התמונה, מספר הבלוקים ויחס הדחיסה. לאחר מכן היא מחלקת את התמונה ל- nb^2 בלוקים ומבצעת לכל בלוק התמרת פורייה דו-ממדית. לאחר מכן היא מבצעת הזזה לערכים במרחב פורייה ושומרת את הערכים הנמצאים סביב מרכז הבלוק, בהתאם לכמות הערכים שדרוש לשמור בכל בלוק (החישוב לזה בוצע בהתחלה). בנוסף על כך, בגלל אופן הפעולה של התמרת פורייה דו-ממדית, ניתן לשמור חצי מהערכים ולשחזר את השאר מתוכם שחלוף וסיבוב בהתאם לרביע בו נמצאים (רביע הוא החלק המתקבל מחלוקת המטריצה ל-4 מטריצות קטנות). הפונקציה לוקחת בחשבון את העובדה שהערכים במרחב פורייה הם מרוכבים ודורשים כפול זיכרון מאשר פיקסל רגיל, והקיזוז בזיכרון מתבטא בכך שמראש היא שומרת רק חצי מהמקדמים ומשחזרת את האחרים מתוכם. לבסוף הפונקציה פורסת את כל הערכים לוקטור ארוך המכיל את כל המקדמים שנשמרו מכל הבלוקים. הסיבה שבגינה נבחר לשמור את הערכים בוקטור אחד ארוך במקום במטריצה, היא על מנת להימנע מאיזורים בלי מידע במטריצה שיושלמו ל-0 אוטומטית ע"י המטלב, ובכך נשמור זיכרון מיותר.

לאחר מכן נבנתה פונקציה נוספת אשר משחזרת את התמונה מתוך וקטור המקדמים השמור בזיכרון:

```
function [NewIm] = ReBFCNA( Fc, nb, FullImR, FullImC, c )
```

הפונקציה הזו מקבלת כקלט את וקטור המקדמים במרחב פורייה (Fc), מספר הבלוקים בכל ממד (nb), ממדי התמונה המקורית (FullImR, FullImC) ואת יחס הדחיסה (c), והיא מחזירה כפלט את התמונה המשוחזרת (NewIm). הפונקציה מבצעת את התהליך ההפוך לתהליך שבוצע בפונקציית הדחיסה ע"י כך שהיא מחלקת את הוקטור המקובץ למקטעים אשר מתאימים לכל

בלוק ומבצעת השמה בהתאם, לאחר מכן היא משחזרת את המספרים המרוכבים הניתנים לשחזור ומבצעת התמרה הפוכה.

3.2. דחיסה אדפטיבית

פונקציית הדחיסה האדפטיבית נראית באופן הבא:

```
function [ Fc, Ind ] = BFCA(Im, nb, c)
```

היא מקבלת כקלט את התמונה (Im), את מספר הבלוקים בכל ממד (nb, כלומר סה"כ nb X nb בלוקים) ואת יחס הדחיסה (c), והיא מחזירה כפלט וקטור מקדמי פורייה דחוס (Fc) ואת האינדקסים התואמים למקדמים (Ind). אופן הפעולה של פונקציה זו הוא פשוט יותר מכיוון שהיא מבצעת התמרה דו-ממדית לכל בלוק ושמה את כל המקדמים במטריצה בגודל אשר זהה לתמונה המקורית. לאחר מכן מבצעים סידור מהערך הגבוה לנמוך לפי הגודל של המקדמים (מספרים מרוכבים). לפי יחס הדחיסה בוחרים כמה משתנים יש לשמור, ושומרים את כל המקדמים הדרושים ואת האינדקס שלהם כאשר המטריצה פרוסה לוקטור עמודה. מכיוון שבפונקציה זו שומרים גם את האינדקסים, יש לשמור פחות מקדמי פורייה על מנת לקבץ את התמונה באותו היחס כמו בפונקציית הדחיסה הלא אדפטיבית, והפונקציה לוקחת את זה בחשבון.

לאחר מכן נבנתה פונקציה נוספת אשר משחזרת את התמונה מתוך וקטור המקדמים ווקטור האינדקסים השמורים בזיכרון:

```
function [NewIm] = ReBFCA( Fc, Ind, nb, FullImR, FullImC)
```

בדומה לפונקציית הדחיסה הלא אדפטיבית, גם במקרה זה הפונקציה מבצעת את הפעולה ההפוכה לפעולה שבוצעה בתהליך הדחיסה. תחילה היא בונה וקטור אפסים בגודל התמונה המלאה כאשר היא פרוסה לוקטור עמודה. אז היא מבצעת השמה בהתאם לאינדקסים השמורים, של ערכי מקדמי פורייה מתוך הוקטור. לאחר מכן היא שינוי צורה לוקטור והופכת אותו למטריצה בממדי התמונה המקורית. לבסוף היא עוברת על כל הבלוקים ומבצעת התמרה הפוכה, ומחזירה את הערכים הממשיים.

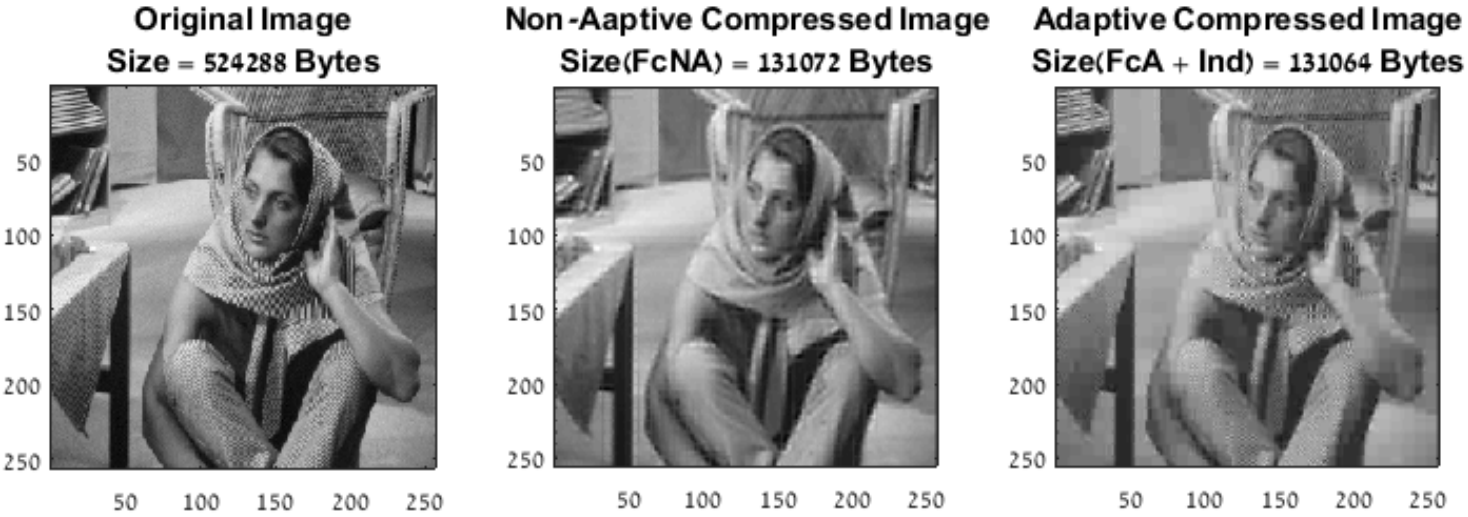
3.3. תיקון קווי ההפרדה

הפעולה של חלוקת התמונה לבלוקים וביצוע התמרה לכל בלוק בנפרד, גוררת שגיאה שנובעת מתוך ההנחה של התמרת פורייה כי הפונקציה המקורבת הינה מחזורית בתחומי ההתמרה. כתוצאה מכך לאחר ביצוע הדחיסה והשחזור, נשארים קווי אורך ורוחב שממוקמים על גבולות הבלוקים בתמונה המשוחזרת. על מנת לנקות את קווים אלו, כתבנו את הפונקציה הבאה:

```
function [ ClearIm ] = CompressClear( Im, nb )
```

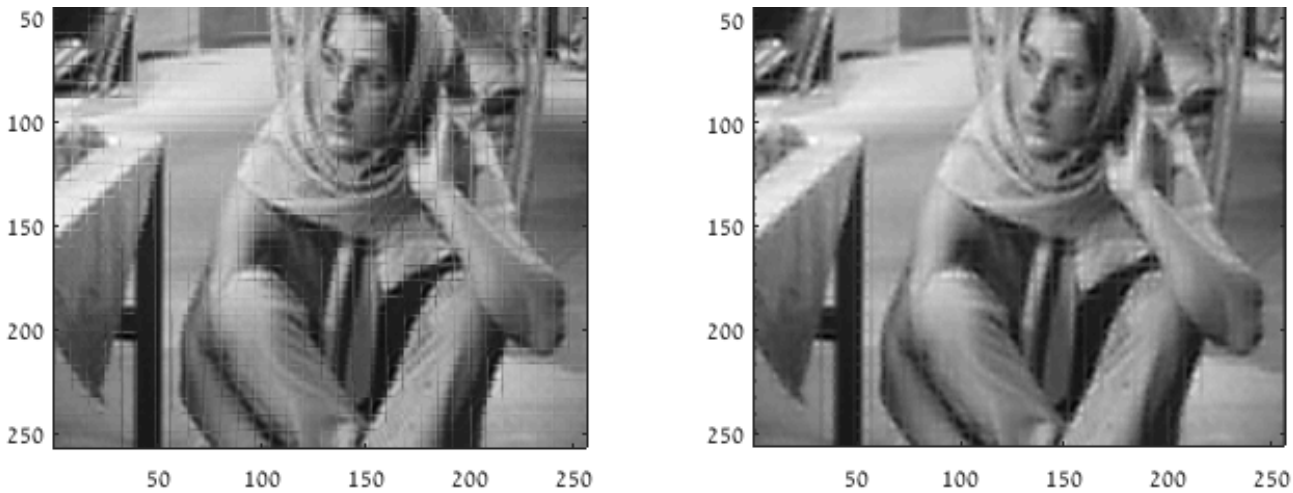
הפונקציה מקבלת כקלט את התמונה לאחר שחזור (Im) ואת מספר הבלוקים בכל ממד (nb), ומחזירה לבסוף את התמונה לאחר סינון הקווים (ClearIm). הפונקציה בונה פילטר גאוסיאני בגודל חלון של 3X3 כפי שבוצע בתרגיל מספר 2. לאחר מכן היא בונה גריד אשר מייצג את הפיקסלים בתמונה אשר נמצאים בסמוך לקווים המפרידים (הפיקסלים על גבולות הבלוקים), מעבירה את הפילטר הגאוסיאני על התמונה, כופלת איבר-איבר עם מטריצת הגריד של קווי ההפרדה, ולבסוף היא משלימה את המקומות הריקים בפיקסלים המקוריים מהתמונה.

מצורף קובץ סקריפט אשר מפעיל את הפונקציות שהוצגו להלן על תמונתה של ברברה, עבור ערכים של יחס דחיסה (c) ומספר הבלוקים בכל ממד (nb) שניתנים לשליטה. הקובץ מציג השוואה בין התמונה המקורית, לבין התמונות המשוחזרות לאחר דחיסה וניקוי קווי ההפרדה, בתוספת ציון הערך הנשמר בזיכרון עבור התמונות הדחוסות. כמו כן הוא מציג השוואה בין התמונות הדחוסות לפני ואחרי ניקוי הקווים. באיור 3.1 ניתן לראות את ההשוואה בין התמונה המקורית לתמונות המשוחזרות, ובאיורים 3.2 ו-3.3 ניתן לראות השוואה בין התמונות לפני ואחרי תיקון הקווים. כל האיורים הם עבור $nb = 32$ ו- $c = 4$.



איור 3.1: התמונה המקורית (משמאל), דחוסה לא אדפטיבית (במרכז) ודחוסה אדפטיבית (מימין).

איור 3.2: התמונה הדחוסה לא אדפטיבית המשוחזרת, לפני (משמאל) ואחרי (מימין) תיקון הקווים.



איור 3.3: התמונה הדחוסה אדפטיבית המשוחזרת, לפני (משמאל) ואחרי (מימין) תיקון הקווים.

4. FFT

א. בסעיף זה נדרש למצוא התמרת פורייה באופן ישיר עבור אותות חד ממדיים ותמונות דו ממדיות.

עבור אותות חד ממדיים

נוכל להשתמש בנוסחה הישירה להתמרת פורייה בדידה:

$$x_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j e^{-\frac{i2\pi kj}{n}}$$

כאשר $x \epsilon C^n$ ווקטור קומפלקסי המייצג את האות הנכנס.

בדרך זו נוכל לחשב בשתי לולאות את כל איברי ייצוג התדר של האות הבדיד x, ולקבל המחשה ברורה לסיבוכיות של $O(n^2)$ עבור החישוב (שתי לולאות באורך n כל אחת).

כמובן שבשימוש במטלב, צורת חישוב זו אינה המתאימה ביותר אלא נשאף לבצע פעולות בין ווקטורים ומטריצות, אי לכך נעזר בפיתוח של התמרת פורייה הבדידה כייצוג של ווקטור מקדמים הפורש את המרחב (להבדיל מהפיתוח שניתן לעשות כהתמרת פורייה רציפה לאות דגום ע"י רכבת הלמים וקבלת *DTFT* וקטימתו לאות סופי וקבלת *DFT*). בצורה זו, נוכל לפרוש את המרחב C^n ע"י קבוצת ווקטורים אורתונורמליים:

$$v_0, \dots, v_n \epsilon C^n$$

מהצורה:

$$v_k(j) = \frac{1}{\sqrt{n}} e^{\frac{i2\pi kj}{n}}$$

אי לכך נוכל להביע את ווקטור האות הנכנס כמטריצה B_{nxn} אשר עמודותיה הן הווקטורים הפורשים ומכפלה משמאל בווקטור המקדמים הפורש $\hat{x}$ כך ש:

$$\hat{x} = B^{-1}x$$

מכיוון ו-B אורתוגונלית וסימטרית מתקיים:

$$B^{-1} = B^{*} = \bar{B} \triangleq A$$

אי לכך חישובה פשוט ומקנה לנו דרך לחישוב התמרת פורייה של האות הנכנס בצורה מהירה ע״י הכפלת מטריצה בווקטור האות הדגום.

עבור מטריצה דו ממדית

נציג תחילה את התמרת פורייה הדו ממדית, אשר מוגדרת על מטריצה $x\epsilon R^{mxn}$ בצורה:

$$x_{k_1,k_2} = \frac{1}{\sqrt{mn}} \sum_{j_1=0}^{m-1} \sum_{j_2=0}^{n-1} x_{j_1,j_2} e^{-i2\pi\left(\frac{j_1k_1}{m}+\frac{j_2k_2}{n}\right)}$$

פעם נוספת, נוכל לבצע חישוב אשר ימחיש את הסיבוכיות $O(n^4)$ של החישוב הישיר, ולהציג כל איבר במטריצה המותמרת ע״י הסכום הכפול, כלומר אלגוריתם גס של 4 לולאות מקוננות. זו כמובן אינה הדרך האופטימלית לחישוב בעזרת מטלב, ונשאף לבצע את הפעולות בעזרת כפל מטריצה ווקטור. כיוון מחשבה ראשון יהיה לעשות זאת ע״י פריסת תמונה כווקטור והגדרת מטריצה מתאימה הפועלת על ווקטור זה (block diagonal), ולאחר מכן להחזיר את הווקטור המתקבל לכדי מטריצה (דרך לא ישימה כ״כ מכיוון והמטריצה הכופלת היא בגודל לא ישים מבחינת זיכרון, גם עבור תמונות קטנות יחסית) כיוון מחשבה שני, והוא זה בו השתמשנו, יהיה להגדיר את המטריצה הבאה:

$$\left\{A_{j_1j_2}\right\}_{j_1j_2} = e^{-i2\pi\left(\frac{j_1k_1}{m}+\frac{j_2k_2}{n}\right)}$$

כאשר:

$$j_1 = 0, \dots, m-1, \quad j_2 = 0, \dots, n-1$$

ולבצע כפל מערך (לא כפל מטריצה!) עם התמונה עבור כל:

$$k_1 = 0, \dots, m-1, \quad k_2 = 0, \dots, n-1$$

ובכך לצמצם את החישוב לשתי לולאות בלבד וכפל מערכים, אותו מטלב מבצע במהירות (הסיבוכיות אינה משתנה אך זמן ההרצה קטן משמעותית ביחס לחישוב הישיר לפי הנוסחה, מהיר יותר בערך פי 3).

הפונקציה שנכתבה Q4A(x, 'sep')

Q4A(x):

- פונקציה זו מקבלת מערך x בגודל m על n ובודקת את ממדיו.
- במידה ו-x הוא סקלר הפונקציה תחזיר את הסקלר.
- במידה והוא ווקטור, הפונקציה תבדוק אם הוא ווקטור שורה או עמודה ותמיר אותו לשורה במידה ואינו כזה, ותבצע את החישוב בעזרת כפל המטריצה שצוין מקודם (ללא הנרמול בשורש אורך המערך).
- במידה והמערך הנקלט הוא מטריצה, יבוצע החישוב בעזרת שתי לולאות וכפל מערכים.
- חשוב לציין שלא נעשות בדיקות לתקינותו של המערך הנכנס, ונקודת ההנחה היא כי הקלט הוא ווקטור\סקלר או מטריצה ותו לא, כל מערך בממד גדול מ-2, יגרור שגיאה. הפונקציה נכתבה על מנת לשרת את מטרות השאלה בלבד .

Q4A(x,'sep'):

- במידה והקלט הוא מערך דו ממדי (מטריצה), ניתן להוסיף אופציה אשר תחשב את ההתמרה הדו ממדית בצורה מהירה בהרבה (הסיבוכיות אינה משתנה אך בשימוש כפל מטריצות אשר אופטימלי עבור מטלב), בשימוש בתכונת הספרביליות.

לפי דרישות השאלה לא נוכל לחשב בצורה זו, אך האופציה קיימת ונעשה בה שימוש לצורך השוואה בהמשך השאלה.

ספרביליות

את התמרת פורייה הדו ממדית ניתן לפרק לשתי התמרות פורייה חד ממדיות ע״י פירוק האקספוננט בצורה הבאה:

$$x_{k_1,k_2} = \sum_{j_1=0}^{m-1}\sum_{j_2=0}^{n-1} x_{j_1,j_2} e^{-i2\pi\left(\frac{j_1k_1}{m}+\frac{j_2k_2}{n}\right)} = \sum_{j_2=0}^{n-1}\left(\sum_{j_1=0}^{m-1} x_{j_1,j_2} e^{-i2\pi\left(\frac{j_1k_1}{m}\right)}\right) e^{-i2\pi\left(\frac{j_2k_2}{n}\right)} = \sum_{j_2=0}^{m-1} x_{k_1 j_2}\, e^{-i2\pi\left(\frac{j_2k_2}{n}\right)}$$

עובדה זו מראה לנו כי נוכל לבצע את ההתמרה לאחד מממדי המטריצה ואז לשני, כלומר התמרת שורות ואז עמודות, או ההפך, לפי צורה זו נוכל להציג את התמרת פורייה הדו ממדית באופן מטריציוני בצורה הבאה:

$$\{X\}_{k_1k_2} = A_{mxm}\{X\}_{j_1j_2}A_{nxn}$$

כאשר המטריצות A הן מאותה הצורה של מטריצת ההתמרה החד ממדית, וכפל משמאל מסמן התמרת העמודות של $\{X\}$ ומימין את התמרת השורות.

מעבר לצורת החישוב היעילה בהרבה עבור מטלב בשימוש בתכונה זו, היא נותנת אינטואיציה טובה כאשר מוצגת כאוסף מכפלות חיצונית של וקטורי מטריצות ההתמרה, ומספקות הבנה כי הבסיס

האורתוגונלי הדו מימדי, מורכב מתמונות מחזוריות (מטריצות מחזוריות מרוכבות יותר נכון), בתדרים שונים וצימודים שונים לאורך שני הממדים של התמונה.

תוצאות סעיף א

נציג להלן את תוצאות זמן ההרצה של הפונקציה אותה כתבנו, ונורמת l_2 של ווקטור ההפרש של התוצאות של הפונקציה שכתבנו ושל הפונקציה המובנית של מטלב. נציג תוצאה עבור שני מערכים חד ממדיים באורכים שונים x_1, x_2, (אות סינוס באורך 100 ואות פרבולי באורך 1001 בהתאמה) ושתי תמונות Im_1, Im_2 בגדלים שונים כאשר הראשונה היא תמונת העיגול משאלה 1 (בגודל 110 על 110) והשנייה היא תמונת המתומן אשר הורחבה לגודל של 200 על 200.

מערך	זמן הרצה בפונקציה Q4a(x) $[sec]$	זמן הרצה בפונקציה fft2/fft $[sec]$	זמן הרצה בפונקציה Q4a(x,'sep') $[sec]$
x_1	0.00852	0.00003	-
x_2	0.05269	0.00007	-
Im_1	4.18521	0.00041	0.003
Im_2	44.1836	0.00038	0.007

טבלה 4.1: תוצאות זמני ההרצה של המערכים השונים

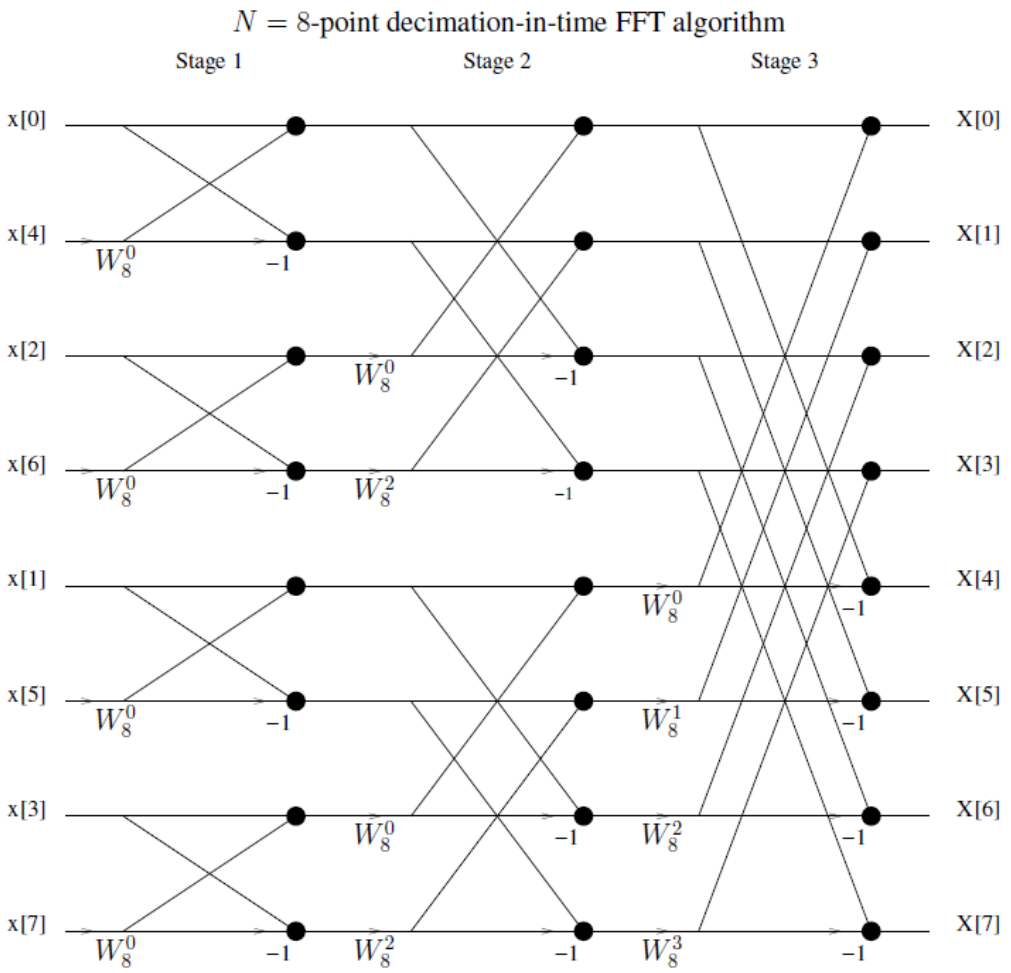
טבלה 4.2: נורמת l2 של וקטור ההפרש בין הפלט שלנו לזה של המטלב.

מערך	שגיאה בפונקציה Q4a(x) $\cdot 10^{-10}$	שגיאה בפונקציה Q4a(x,'sep') $\cdot 10^{-10}$
x_1	0.0135	-
x_2	119060	-
Im_1	4.683	2.699
Im_2	5.877	4.288

ניתן כאמור לראות כי ביצועי הפונקציות של מטלב טובות בכמה סדרי גודל מבחינת זמן הרצה. מעבר לכך ככל שהמערך אותו נכניס לפונקציה שלנו גדל, זמן ההרצה גדל (אמור להיות לפי חזקה שניה או רביעית בהתאם לממדי המערך) וכך גם השגיאה ביחס לפלט של הפונקציה של מטלב.

התוצאות המספקות היחידות נובעות משימוש בתכונת הספרביליות, אשר כלל אינה אמורה להיות חלק מהתרגיל והתווספה רק לשם השוואה. בשימוש בתכונה זו אנו רואים כי זמן ההרצה שונה רק בסדר גודל יחיד מזה של פונקציית *fft2* של מטלב, והשגיאות ביחס אליה נמוכות מהתוצאות האחרות.

ב. בסעיף זה נדרשנו לכתוב אלגוריתם *fft* וליישמו על אותות חד ממדיים באורך דיאדי. האלגוריתם אשר ניישם דומה לחלוטין לזה שד״ר עפר לוי יישם המבוסס תחילה על חלוקת ההתמרה הבדידה לסכום על האינדקסים הזוגיים וסכום על אלו האי זוגיים, וקבלת שתי התמרות למעשה. הראשונה היא התמרת האינדקסים הזוגיים והשנייה, היא קבוע כלשהו (קבוע עבור כל רכיב תדר) כפול ההתמרה של האינדקסים הזוגיים. בנוסף לכך נוכל לראות כי מכיוון וההתמרה בעלת אופי מחזורי, לאחר *n/2* צעדי תדר, האיברים יהיו הצמודים של אלו שחושבו עד לשלב זה. בצורה זו נוכל להמשיך לפרק כל התמרה לסדרת אינדקסים סוגיים ואי זוגיים ולחשב עד מחציתה. נצפה כאמור לקבל סיבוכיות התלוי בלוגריתם בבסיס 2 שכן כל פעם מחולק החישוב ואכן הסיבוכיות יורדת ל - $O(nlog_2(n))$. אלגוריתם זה מכונה *radix-2* (בסיס 2) זאת עקב החלוקה בחצי לסדרות אינדקסים שונים. האלגוריתם מכונה לעיתי גם כ-*butterfly* מכיוון וניתן להמחיש את אופן החלוקה בעצרת דיאגרמות של קווים מצטלבים שיוצרים דמיון (סובייקטיבי לחלוטין) לכנפי פרפר.



$$W_N = e^{-j\frac{2\pi}{N}}$$

<u>איור 4.1:</u> המחשה של האלגוריתם.

חשוב לציין כי יישום בצורה רקורסיבית אינו היעיל ביותר במטלב והפונקציה המובנית מהירה מהרבה. ייתכן והפונקציה עושה שימוש בעיבוד מקבילי וחלוקה לבסיס 4 (*radix 4*) המתאימה מאוד לעיבוד מקבילי, או מאידך כתובה באופן טוב בהרבה וחוסכת חישובים רבים בכך שהיא מנצלת את התבנית של האלגוריתם אשר מתארת אילו סדרות צריך לחבר לאילו. בכל אופן אלו אינן האופציות היחידות בהן ייתכן והפונקציה המובנית מבצעת את החישוב.

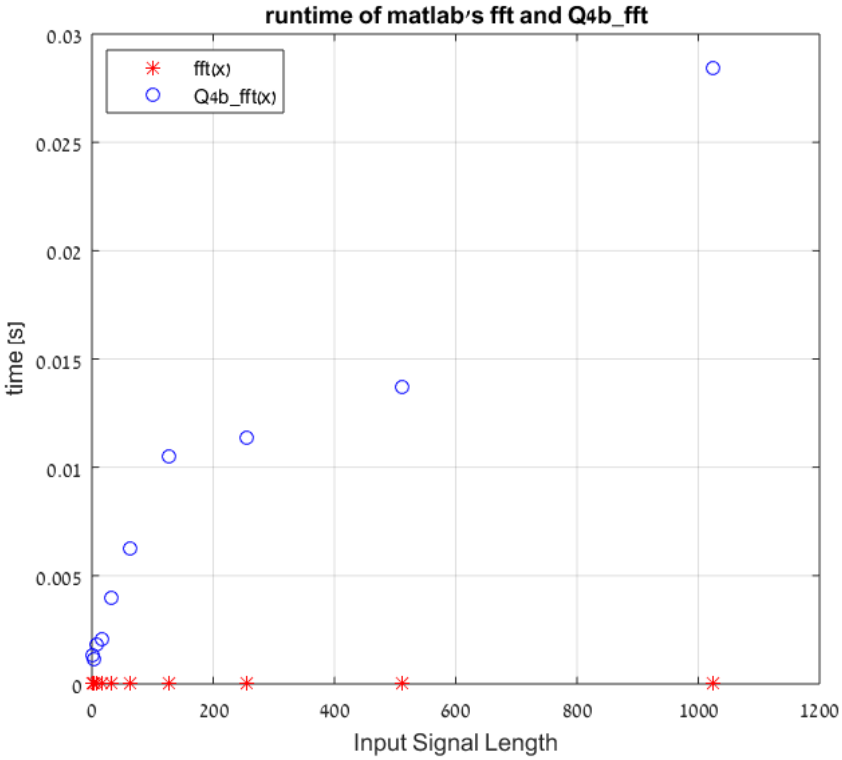
תוצאות סעיף א והפונקציה Q4b_fft(x)

להלן נציג את זמן ההרצה של הפונקציה הרקורסיבית Q4b_fft(x) אל מול זמן ההרצה של הפונקציה המובנית במטלב, עבור ווקטור באורך דיאדי משתנה:

$$length(x_n) = 2^n, \;\; n = 1, \dots, 10$$

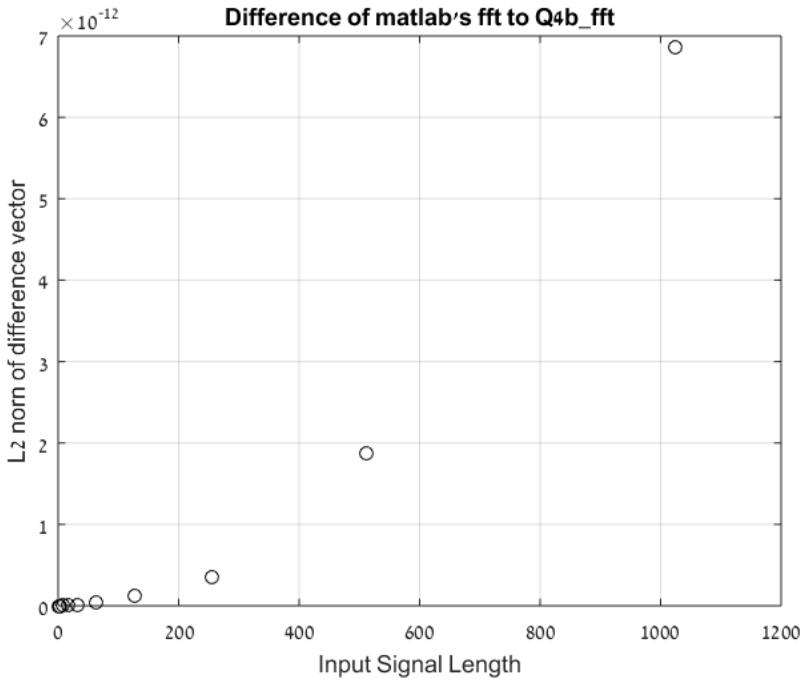
כאשר האות הדגום יהיה:

$$x = \sin(2\pi \cdot n \cdot t)$$



איור 4.1: גרף המתאר את זמני הריצה של שתי הפונקציות המושוות עבור ווקטור כניסה באורך דיאדי משתנה.

ניתן לראות בבירור כי זמן הריצה של הפונקציה המובנית אפסי לעומת זה של הפונקציה הרקורסיבית שנכתבה עבור סעיף זה, מעבר לכך זמני הריצה שמתקבלים בפונקציה זו ארוכים בהרבה מאלו אשר מתקבלים כאשר נכניס את האות לפונקציה שנכתבה בסעיף א בה החישוב נעשה בצורה מטריציונית, דבר המצביע בעיקר על העליונות של פעולות בין ווקטורים ומטריצות במטלב, לעומת הסתכלות רק על סיבוכיות הסכמה.



<u>איור 4.2:</u> נורמת ווקטור ההפרש בין הפלט בפונקציה של מטלב לבין זה שנכתב עבור סעיף זה.

בנוסף ניתן לראות כי ככל שווקטור הכניסה גדל, כך גדלה גם השגיאה אל מול הפלט של הפונקציה *fft()* של מטלב.

ג. לפי תכונת ההפרדה שנידונה בסעיף א', נוכל להשתמש באלגוריתם הרקורסיבי מהסעיף הקודם ולהפעיל אותו על שורות מערך דו ממדי (בעל ממדים דיאדים) ולאחר מכן על עמודותיו.

הפונקציה *Q4c_fft2(Im)*

הפונקציה מקבלת תמונה בעלת ממדים דיאדים ושולחת את עמודות התמונה לפונקציה Q4b_fft(x) מהסעיף הקודם, לאחר מכן היא שולחת את שורות המטריצה המורכבת מייצוג התדר של עמודות התמונה, אל Q4b_fft(x) פעם נוספת. בצורה זו (נניח לרגע כי התמונה ריבועית *nxn*) הפונקציה מבצעת אלגוריתם בסיבוכיות $O\left(nlog_2(n)\right)$, *2n* פעמים וכתוצאה הסיבוכיות עולה ל - $O\left(n^2log_2(n)\right)$.

להלן נשווה פעם נוספת את זמני הריצה ונורמת ההפרש

בקובץ ההרצה של שאלה 4, סעיף ג' מחולק לשני חלקים. בראשון מבוצעת ההתמרה עבור תמונת העיגול הבינארית משאלה 1 (מורחבת לגודל של 128 על 128) בעזרת כל הפונקציות שנכתבו עד כה. בטבלה הבאה נערוך השוואה בין זמני הריצה השונים :

טבלה 4.3: השוואה בין הפונקציות השונות עד כה.

Q4c_fft2(x)	Q4a(x, 'sep')	fft2	Q4a(x)	
0.750290	0.006037	0.000859	9.886034	**זמן הרצה** $[sec]$
1.395	19.64	0	24.60	$\boldsymbol{l_2\ norm\ of\ vectorized\ matrix \cdot 10^{-11}}$

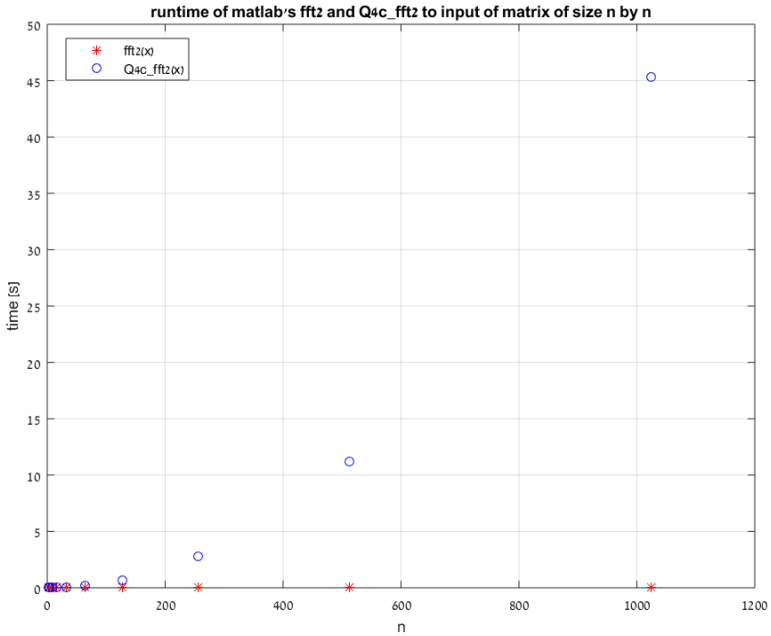
ניתן לראות כי עבור התמונה הקטנה בה השתמשנו קיבלנו זמן ריצה ארוך מאוד בשימוש בפונקציה הרקורסיבית ותכונת ההפרדה (*Q4c_fft2*) ביחס ל- *fft2* של מטלב, בעוד שימוש בתכונת ההפרדה וביצוע *DFT* בעזרת הפונקציה מסעיף א, הניב תוצאות טובות בהרבה.

לפי תוצאות הסעיפים הקודמים אין הדבר מפתיע, ויש לשער כי גם הפונקציה *fft2* של מטלב אינה עושה שימוש ברקורסיה בצורה בה אנחנו בצענו את התרגיל והיא מותאמת לעיבוד מקבילי (השערה בלבד) וכתובה בצורה יעילה בהרבה תוך היכרות טובה מאוד עם האלגוריתם בו היא עושה שימוש. יש לשער כי גם כי *fft2* משתמשת בתכונת ההפרדה וב-*fft* החד ממדי.

החלק השני של סעיף ג, בקובץ ההרצה, בונה סדרת מטריצות בגדלים דיאדים שונים של $2^n x 2^n$ כאשר $n = 2, \dots, 10$ המתארים את הפונקציה:

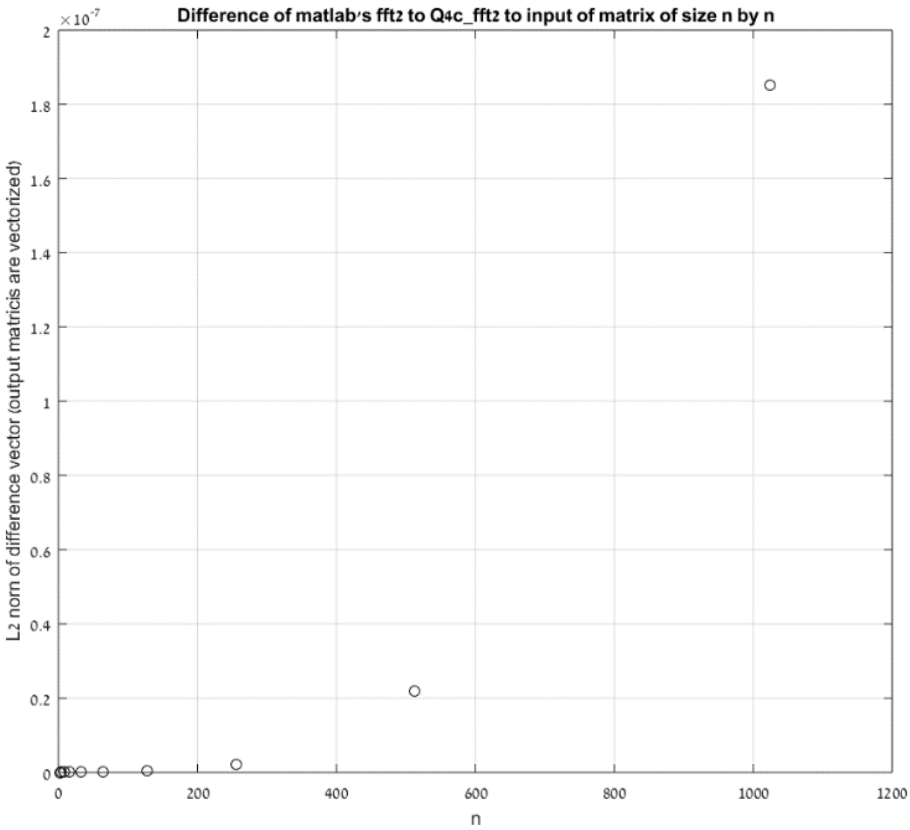
$$f(x,y) = \sin\big(2\pi(x+y)\big) + \mathrm{n} \cdot \sin\big(2\pi n(x+y)\big) + 9\sin\big(10\pi(x+y)\big) + 8\sin\big(18\pi(x+y)\big)$$

לאחר מכן מתבצעת השוואה של זמני הריצה והשגיאה ביחס לפלט של *fft2()*.



איור 4.3: זמן ההרצה של הפונקציה *fft2* והפונקציה הרקורסיבית העושה שימוש בתכונת ההפרדה.

כצפוי גם כאן נקבל הבדלים עצומים, כאשר כבר עבור תמונות בגודל של 512 על 512, זמן ההרצה לא מאפשר שימוש ישים בפונקציה.



איור 4.4: השגיאה של פלט הפונקציה *fft2* ביחס לפונקציה הרקורסיבית העושה שימוש בתכונת ההפרדה.

פעם נוספת גרף השגיאות מאמת את התוצאות שקיבלנו ומאפשר בחינה של השגיאה התקבלת ביחס לפונקציה המובנית, כאשר התמונה גדלה.