מעבדה בעיבוד אותות פזיולוגיים  
הנדסה ביורפואית

מגישים:

דן טורצקי   
סול אמארה

תאריך:

20.12.2022

תוכן עניינים:

[1 תקציר: 3](#_Toc122271261)

[2 ניסוים: 4](#_Toc122271262)

[1. ניסוי 1: 4](#_Toc122271263)

[2.1.1 היפותזה: 4](#_Toc122271264)

[2.1.2 מתודולוגיה: 4](#_Toc122271265)

[2.1.3 תוצאות: 4](#_Toc122271266)

[2.1.4 מסקנות: 13](#_Toc122271267)

[2. ניסוי 2: 14](#_Toc122271268)

[2.2.1 היפותזה: 14](#_Toc122271269)

[2.2.2 מתודולוגיה: 14](#_Toc122271270)

[2.2.3 תוצאות: 14](#_Toc122271271)

[2.2.4 מסקנות: 17](#_Toc122271272)

[3. ניסוי 3: 18](#_Toc122271273)

[2.3.1 היפותזה: 18](#_Toc122271274)

[2.3.2 מתודולוגיה: 18](#_Toc122271275)

[2.3.3 תוצאות: 18](#_Toc122271276)

[2.3.4 מסקנות: 25](#_Toc122271277)

[4. ניסוי 4: 26](#_Toc122271278)

[2.4.1 היפותזה: 26](#_Toc122271279)

[2.4.2 מתודולוגיה: 26](#_Toc122271280)

[2.4.3 תוצאות: 26](#_Toc122271281)

[2.4.4 מסקנות: 32](#_Toc122271282)

[3 מסקנות כלליות 33](#_Toc122271284)

[4 נספחים 34](#_Toc122271285)

# תקציר:

במעבדה זו למדנו להשתמש באלגוריתם של עיבוד תמונה ולבצע באמצעותם גם עיבוד מקדים בהתאם לצרכים שלנו וגם זיהוי אובייקטים. בניסוי הראשון חקרנו תמונה של טסיות דם הנלקחה מתוך המטלב וראינו כי ניתן לשפר את ביצועי זיהוי הטסיות באמצעות הפחתת הרקע ששערכנו על ידי open&close. בנוסף, למדנו כי ניתן לקבל תמונה המציגה רק את גבולות הטסיות באמצעות פעולות כמו לפלסיאן, שכן מסנן מסוג HPF ישאיר רק את התדרים הגבוהים המייצגים מעבר חד בתמונה- גבולות. מטרת הניסוי השני היתה סינון של רעש מסוג מלח פלפל הפוגע באיכות התמונה בעזרת מסנני חציון בגדלים שונים. ראינו כמצופה כי ככל שנגדיל את המסנן נקבל שיפור בסינון עד לגודל מסוים שהחל ממנו נקבל טשטוש של התמונה. תוצאה זו נובעת מכך שכאשר המסנן גדול מידי, ערכי פיקסלים רחוקים שונה ולכן ישפיעו על החציון באופן ניכר. לעומת זאת, עבור מסננים יותר קטנים בהם ערכי הפיקסלים קרוב למעט הרעש, נקבל כי הרעש כמעט איננו משפיע ותוצאת הסינון טובה. בניסוי השלישי, מצאנו את מיקומי האישונים מתמונות שנלקחו במעבדה ובעזרתם מצאנו את כיווני ההסתכלות. לפי התאוריה, קיבלנו כי ניתן למצוא עצמים בתמונות על ידי קורלציה של התמונה עם תמונת רפרנס של העצם מתוך ההנחה כי הקורלציה המקסימלית (אם העצם לבן כלומר 1), תהיה מרכז העצם שחיפשנו בתמונה. בנוסף, בניסוי הרביעי ראינו כיצד ניתן לשערך מהירות מתוך סרטון באמצעות חישובי המרחקים האופקיים שעוברים המרקרים ביחידות של פיקסל והמרתם ליחידות אורך כמו ס"מ. כמו כן, חישבנו זוויות של חלקי הגפה באמצעות חישובים גאומטריים.

# ניסוים:

## ניסוי 1:

### היפותזה:

בניסוי זה ננסה למצוא סף שיפריד בין הטסיות לרקע התמונה. נצפה כי לאחר שערוך הרקע והורדה שלו מהתמונה המקורית נוכל לבצע זאת בצורה מיטבית יותר שכן לאחר ההחסרה הרקע יהיה אחיד. בשל כך, במידה והסף יפריד בצורה טובה בין הטסיות לרקע נצפה להצליח לשערך את כמות הטסיות שכן כעת מיקומן בתמונה ברור. בשל כך, נצפה גם שמציאת הטסיות הנוטות לכיוון מסוים יצליח.

### מתודולוגיה:

מכשור וציוד: תוכנת MATLAB

מהלך הניסוי: בניסוי זה נבצע אלגוריתמים של עיבוד תמונה על תמונת אורז שנלקחה מתוך המטלאב. ניסינו על ידי ניסוי וטעיה למצוא רף שיפריד בין הטסיות לרקע. לאחר מכן, יצרנו מבנה דיסק על מנת להעלים את הטסיות ולקבל את הרקע באמצעות . לאחר מכן הפחתנו את הרקע מהתמונה המקורית. לתמונה זו ביצענו שוב את מציאת הסף. השתמשנו בתמונה הבינארית לאחר החסרת הרקע למציאת גבולות הטסיות. באמצעות תמונת הגבולות של הטסיות מצאנו את מספר הטסיות בתמונה. למציאת הטסיות הנוטות שמאלה בדקנו עבור כל גבול ייחודי (כלומר היקף טסייה מסוימת) האם הפיקסל הנמוך ביותר מבין הפיסקלים בקצה השמאלי של הטסייה גבוהים מסנטרואיד הטסייה.

### תוצאות:

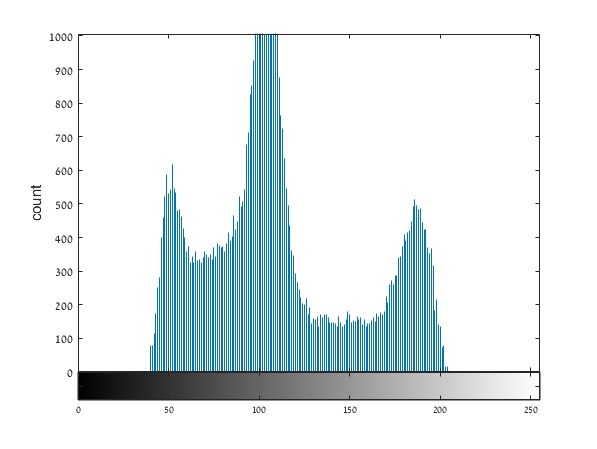
תמונה שמכילה טקסט, בד

התיאור נוצר באופן אוטומטי

איור 1: טסיות הדם. תמונה שמורה של המטלאב "rice"

**שלב 1:**

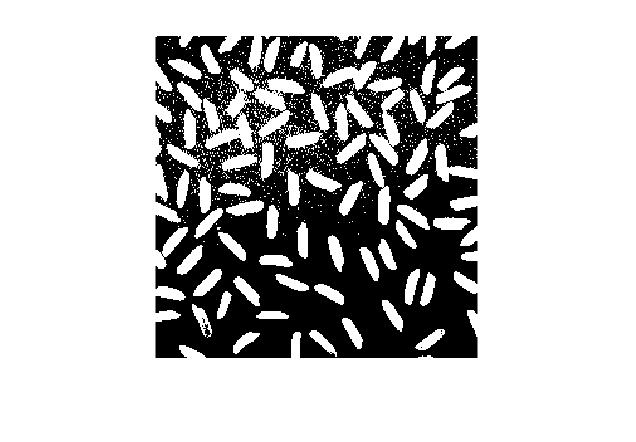
1.2



איור 2: היסטוגרמת תמונת הטסיות

בהיסטוגרמה ניתן לראות כי יש שלושה פיקים עיקריים. אנו משערים כי הפיק הנמוך ביותר שייך לרקע, הפיק השני שייך לרקע הבהיר יותר שבמרכז התמונה והפיק השלישי שייך לטסיות הבהירות שבמרכז התמונה. אם כך, נחפש סף יחיד הנמצא בין הפיק השני לשלישי כך שנוכל להפריד בין הרקע לטסיות בלי לאבד מידע על הטסיות עם סינון של כמה שיותר מהרקע.

1.3



איור 3: תמונה בינארית של הטסיות עם בחירת סף של 115

ניתן לראות כי עבור סף זה מתחילה להיות פגיעה מסוימת בטסיות וכי לא סיננו את כל הרקע. זאת כיוון שיש חפיפה בין הפיקסלים של הרקע לטסיות, כך שלא ניתן להפריד בין הרקע לטסיות באופן מוחלט באמצעות סף לתמונה בינארית.

1.4



איור 4: תמונת גרגרי האורז עם קוטר גרגיר אורז מסומן



איור 5: הרקע שחילצנו באמצעות הדיסקה ושיטת Open->close

כפי שניתן לראות, קיבלנו גווני אפור הדומים לגווני הרקע של התמונה. החלק העיקרי במציאת הרקע הינו פעולת ה – erosion שבוצעה תחילה עם הדיסק. לשם מציאת הרקע עלינו להסיר את הטסיות. כיוון שהטסיות בהירות יותר מהרקע, עלינו לבחור דיסק ברדיוס הגדול שווה לאורך הטסייה, על מנת שהדיסק יחיל במלואו את הטסיות. בצורה זו, כאשר נבצע הטסיות יוחלפו בפיקסל כהה. פעולת ה - שמבוצעת אחרי מחליקה את הרקע *לגוונים הדומים לגווניו המקוריים.*

כעת, נחסר רקע זה מהתמונה המקורית:

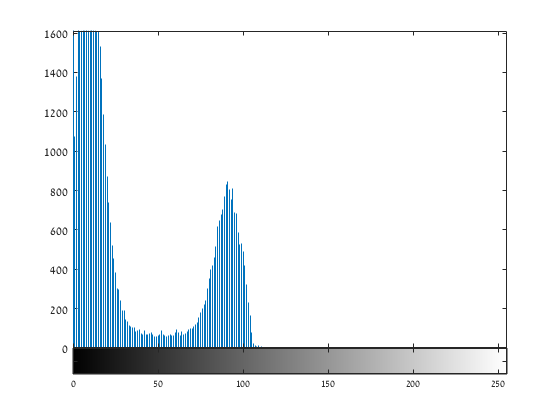
תמונה שמכילה טקסט, בד

התיאור נוצר באופן אוטומטי

איור 6: תמונת האורז לאחר הפחתת הרקע

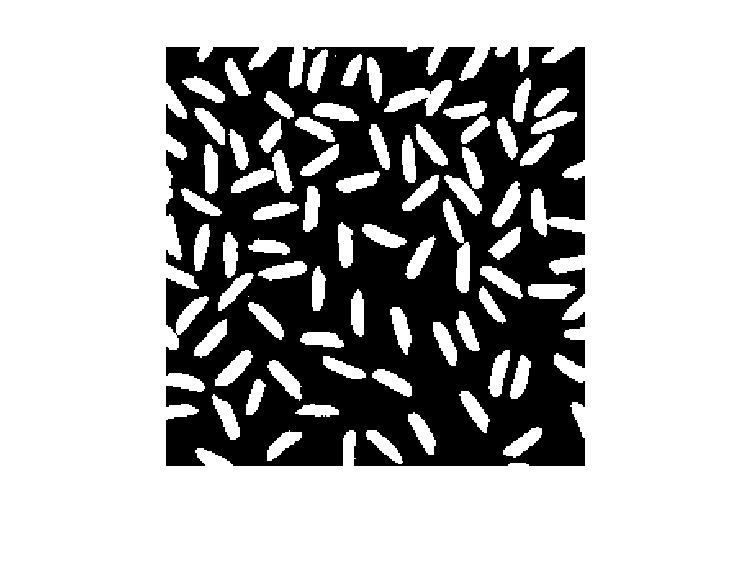
ניתן לראות כי החסרנו את הרקע מהתמונה. כיוון שבאזורים בהם יש טסיות לא יכולנו לשים פיקסל עם ערך אפס, הטסיות הוכהו מעט, אך הרקע אחיד יותר והטסיות מובדלות מהרקע בבירור.

1.5.



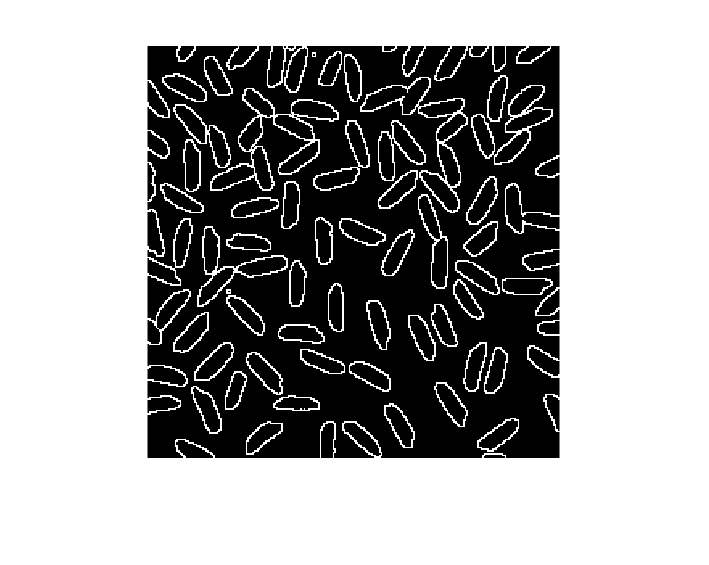
איור 7 – היסטוגרמת תמונת גרגרי האורז לאחר החסרת הרקע

בהיסטגורמה של התמונה המקורית (איור 2) ראינו כי ישנם יותר פיקים שמתארים את התמונה בגלל שהגוון של הרקע והטסיות לא היה אחיד. באיור הנתון לעיל ניתן לראות כי ישנו פיק בגוונים הנמוכים השייך לרקע הכהה שיצרנו, ופיק נוסף השייך לטסיות. כמו כן, ניתן לראות בהיסטוגרמה כי הבהירות של הטסיות ירדה ביחס לבהירויות המקוריות של הטסיות.



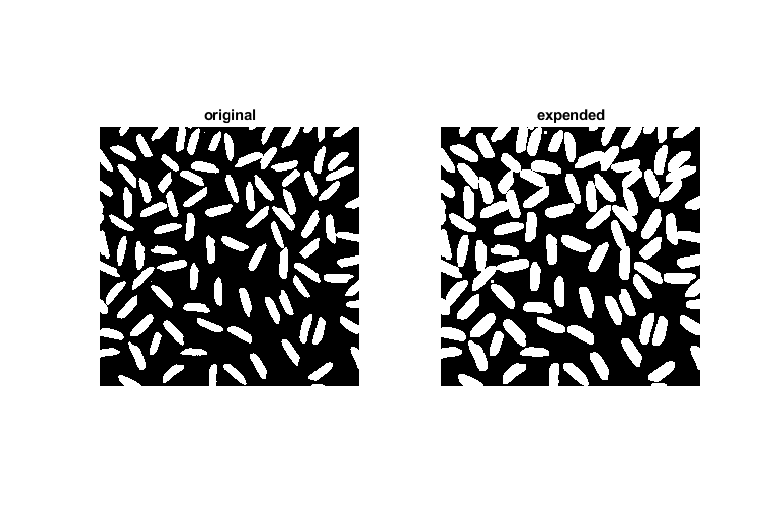
איור 8 – תמונה בינארית של תמונת האורז לאחר החסרת הרקע

ניתן לראות כי כעת ההפרדה בין הרקע לטסיות הדם הינה מוצלחת הרבה יותר. בחרנו את הסף להיות 50 כך שיהיה בין שני הפיקים בהיסטוגרמה. נבצע זיהוי גבולות בשתי דרכים. ראשית, נמצא את הגבולות באמצעות מסנן לפלסיאן המזהה נגזרות גדולות, כך שנזהה את אזורי המעבר מטסיות לרקע בהם יש שינוי גדול בין הפיקסלים:



איור 9: זיהוי הגבולות באמצעות מסנן לפלסיאן

כפי שניתן לראות, באמצעות מסנן לפלסיאן זוהו בהצלחה קצוות הטסיות. כעת, נבצע זיהוי גבולות באמצעות הרחבת התמונה ע"י dilation והחסרת התמונה המקורית מהתמונה המתקבלת:



איור 10 – התמונה הבינארית המקורית (משמאל) והתמונה לאחר הרחבתה באמצעות dilation (מימין)

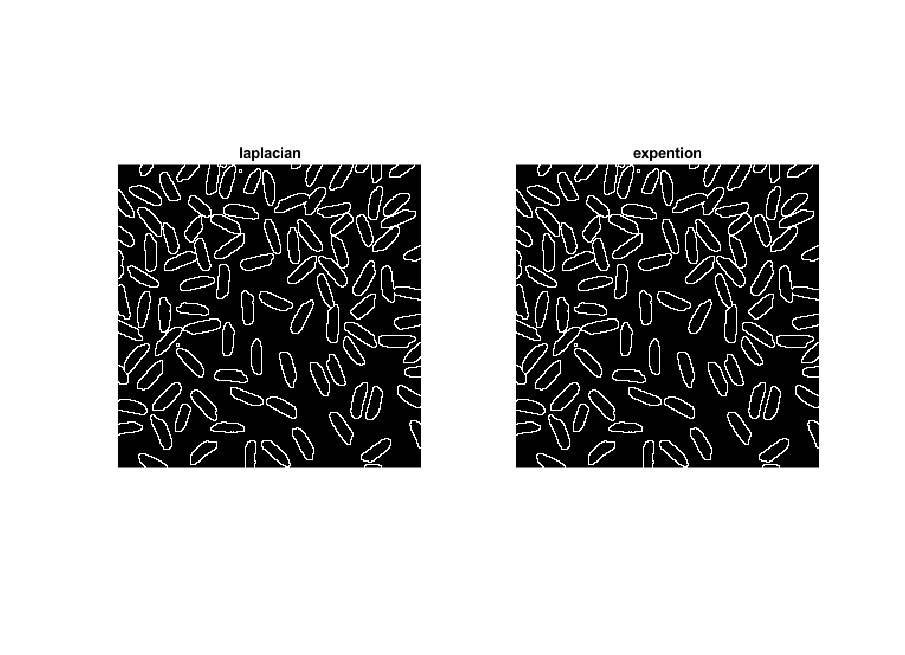
הפחתת התמונה המורחבת מהתמונה המקורית נותנת:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

איור 11: זיהוי הגבולות באמצעות הרחבת התמונה והפחתת התמונה המקורית מהתמונה המורחבת

כפי שניתן לראות, גם בשיטה זו קיבלנו זיהוי גבולות מוצלח. כעת נציג את שתי תמונות זיהוי הגבולות שהתקבלו לשם השוואה:



איור 12: השוואה בין שתי הדרכים לזיהוי הגבולות

למראית עין התוצאות כמעט זהות, לכן במקרה זה אין העדפה ברורה לאחד מהתוצאות המתקבלות.

1.6.

תמונת הגבולות לאחר הסרת טסיות שנמצאות בקצה התמונה, כלומר חלקן נמצא מחוץ לתמונה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

איור 13: תמונת זיהוי הגבולות לאחר הסרת טסיות הנמצאות בקצה התמונה

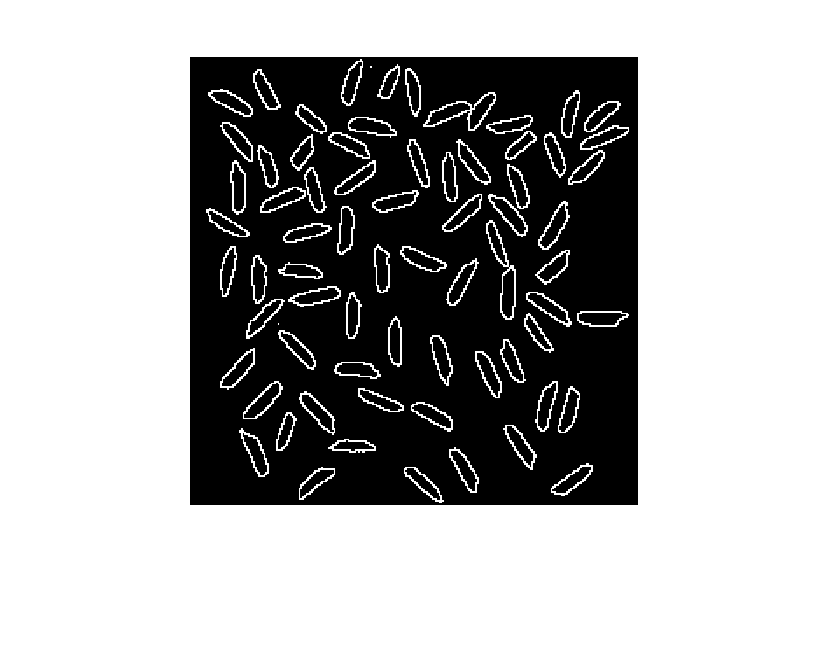
בספירה ידנית, ספרנו 68 טסיות בתמונה (באיור 12). באמצעות פקודת bwlabel במטלאב קיבלנו זיהוי של 53 טסיות, וייתכן שזאת מכך שחלק מגבולות הטסיות מחוברות אחת לשנייה. כדי למנוע מצב זה, ניקח את מסנן הלפלסיאן במינוס, וכך נתפוס את המעבר בין שחור ללבן מהכיוון הפנימי של הטסיה. נסביר זאת – בפועלת הלפלסיאן הרגילה הגבול יתקבל על המעטפת החיצונית של הטסייה היות באזורי הגבול אנו לוקחים פיקסל שחור שערכו אפס, מכפילים במינוס שמונה ומוסיפים אליו את הפיקסלים הלבנים השכנים – כלומר קו הגבול מתקבל בפועל איפה שהיו הפיקסלים השחורים שהקיפו את הטסיות. כעת, כאשר המרכז בפלוס והפיקסלים השכנים מוחסרים מהפיקסל במרכז המסנן, נקבל את קו הגבול במקום בו היו הפיקסלים הלבנים בהיקף הטסיות. תוצאת סינון זו הינה:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

איור 14: זיהוי הגבולות עם שימוש במסנן לפלסיאן המוכפל במינוס 1

ניתן לראות כי כעת ישנה הפרדה טובה יותר בין הטסיות. לאחר הסרת הטסיות הנמצאות חלקית מחוץ לתמונה:

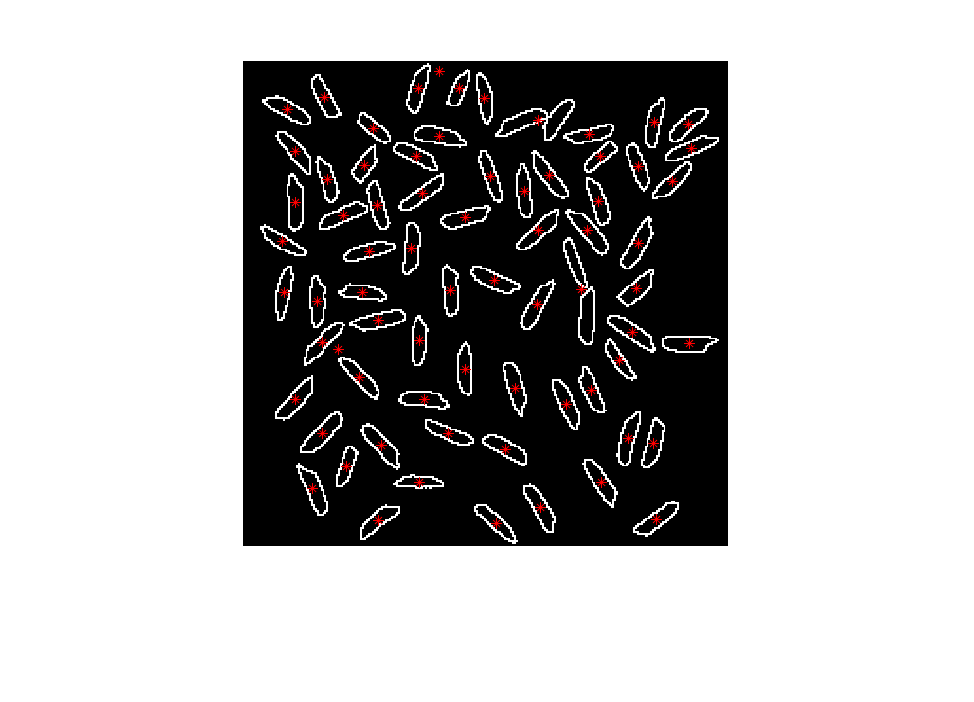


איור 15: תמונת זיהוי הגבולות (איור 14) לאחר הסרת טסיות הנמצאות בקצה התמונה

בתמונה זו זוהו 71 גופים נפרדים, כלומר זוהו 71 טסיות. אם כך, נעריך כי ישנם כ 71 טסיות ליחידת שטח.

**שלב 2:**

1.7. השתמשנו במטריצה המתאימה לאיור 15. באמצעות מצאנו את הסנטרואיד של כל טסייה:



איור 16: איור 15 לאחר הוספת סנטרואידים

ניתן לראות כי לרוב הטסיות נמצאו סנטרואידים מתאימים, אך ישנן כמה טסיות שנותרו מחוברות ולכן הסטנרואיד המשותף שלהן מספק מידע שגוי וכן לכלוכים בתמונה שנקלטו כצורות ולכן יש להן סנטרואידים. באמצעות פקודת עברנו טסייה טסייה ובדקנו האם הפיקסל הנמוך ביותר מבין הפיקסלים בקצה השמאלי של הטסייה גבוה יותר מהסנטרואיד. במידה וכן, טסייה זו הוגדרה כנוטה שמאלה וכל שטח הטסייה קיבל את הערך 1:

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

איור 17: מסכת הטסיות הנוטות שמאלה

ניתן לראות כי מרבית הטסיות הנוטות שמאלה זוהו ומס' הזיהויים השגויים נמוך.

1.8 התמונה המקורית עם טסיות הנוטות שמאלה בלבד:



איור 18: הטסיות הנוטות שמאלה בלבד

ניתן לראות כי אכן מוכלות טסיות הנוטות שמאלה מהתמונה המקורית במסכה.

### מסקנות:

בניסוי זה חקרנו תמונה של טסיות דם הנלקחה מתוך המטלב. ראינו כי כאשר ניסינו לחפש סף שיפריד ביניהן לרקע, קיבלנו פגיעה מסוימת בטסיות על חשבון הצלחת ההפרדה. לאחר מכן, שערכנו את הרקע באמצעות פעולת הerosion והחסרנו אותו מהתמונה המקורית. פעולה זו עזרה לנו לעשות הפרדה טובה יותר שפגעה בתמונה פחות. בשל כך, נסיק כי כאשר נרצה למצוא עצמים מסוימים בתמונות, דרך טובה שיכולה לעזור בהצלחת האלגוריתם הינה החסרת הרקע והפיכתו לאחיד בקירוב. לאחר מכן, ראינו כי ניתן למצוא את קצוות העצמים בתמונה באמצעות מסנן הלפלסיאן מתוך היותו מסנן HPF, כיוון שקצוות הינם בעלי תדר גבוה (שינויים חדים). דרך שניה למציאת הקצוות שראינו הינה הרחבת התמונה והפחתת התמונה המורחבת ממנה. בשני דרכים אלו ישנה בעיה בהפרדה בין עצמים במקרה של קצוות חופפים. ראינו בניסוי זה כי ניתן לפתור את הבעיה הזו באמצעות הכפלת מסנן הלפלסיאן במינוס, פעולה שמוצאת את המעבר מהכיוון הפנימי. יתר על כן, ראינו כיצד ניתן להפריד עצמים הנוטים לכיוון מסוים על ידי מציאת המרכז שלהם.

## ניסוי 2:

### היפותזה:

נצפה לקבל בניסוי זה שככל שגודל המסנן יגדל נקבל סינון טוב יותר כתוצאה מההנחה שבסביבת פיקסל מסוים כל הערכים סביבו קרובים עד לגודל מסוים החל ממנו נצפה לקבל טשטוש. הנחה זו נובעת מכך שלא יתכן כי מסננים גדולים מידי יתנו תוצאה טובה ונצפה לראות שהחל מנקודה מסוימת יש פגיעה בתוצאות כיוון שהנחה זו לא תקפה.

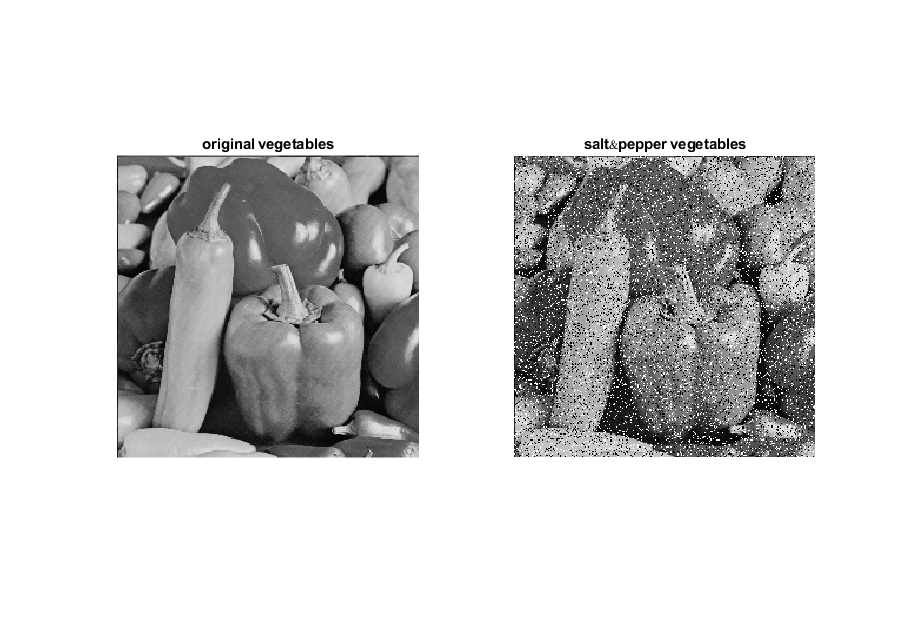
### מתודולוגיה:

מכשור וציוד: תוכנת MATLAB

מהלך הניסוי: בניסוי זה השתמשנו במסנן חציון בגדלים שונים על מנת לסנן את הרעש מהתמונה על ידי פילטרים בממדים שונים. לאחר מכן חישבנו את השגיאה בין התמונה המקורית למשוחזרת ובדקנו מובהקות סטטיסטית של התוצאות.

### תוצאות:

2.1.

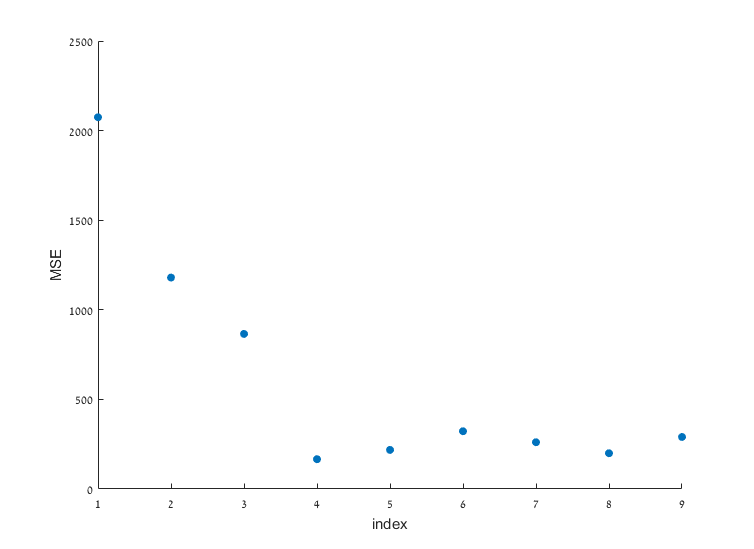


איור 19: תמונת הירקות המקורית (משמאל) ותמונה הירקות בתוספת תיבול של מלח ופלפל (מימין)

ניתן לראות את תמונת הירקות המקורית (משמאל באיור) ואת התמונה לאחר הוספת רעש (מימין באיור) שננסה לסנן כדי להתקרב ככל הניתן לתמונה המקורית. את הסינון ביצענו באמצעות מסנני חציון בממדים שונים. תוצאות הסינון:

גודל ממדי המסננים גדלים משמאל למעלה באיור עד לימין למטה. תחילה ככל שנגדיל את ממדי המסנן, הסינון של רעש המלח פלפל משתפר למראית עין וכן נראה כי איכות התמונה משתפרת. מהתבוננות בתוצאת הסינון עבור המסננים בממדים ו - נראה כי בשניהם רעש מלח פלפל סונן היטב, אך עבור המסנן בגודל נראה כי התמונה מטושטשת יותר. לכן, מהתבוננות בתמונות, לדעתנו מבחינה ויזואלית המסנן בגודל ביצע את הסינון המיטבי ביותר ביחס למסננים בהם השתמשנו. גרף שגיאת של תוצאות הסינון ביחס לתמונה המקורית:

איור 20: תוצאות הסינון של התמונה המורעשת באמצעות מסנני חציון בעלי ממדים שונים



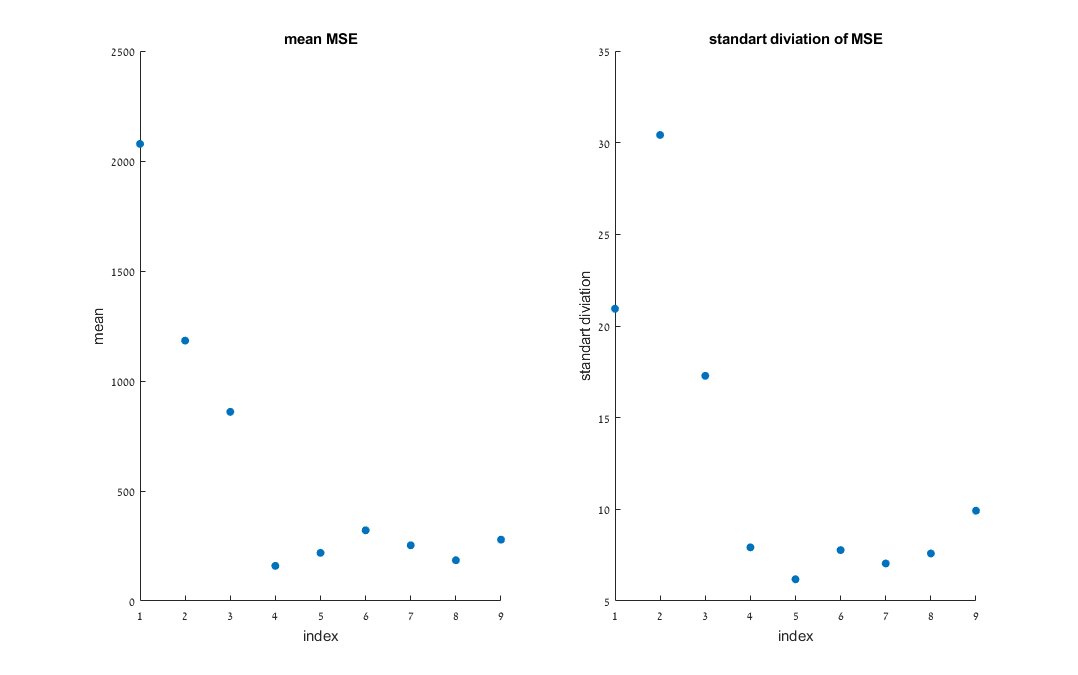
איור 21: MSE בין התמונה המקורית לתמונות המסוננות

באיור ניתן לראות את מדד ה – MSE בין התמונה המקורית לתמונות המסוננות, כאשר סדר האינדקסים תואם לסדר 1-9 כאשר 1 מתאים באיור 14 לפינה השמאלית למעלה ו – 9 מתאים באיור 14 לתמונה הימנית למטה. בניגוד לבחירה שלנו לסינון הטוב ביותר – מסנן בגודל (תמונה 8) - שנבעה מהתבוננות, התמונה בעלת ה – MSE הנמוך ביותר הינה תמונה 4 (מסנן בגודל )*. עם זאת, ניתן לראות כי ההפרש בין השגיאות המצוינות לעיל הינו קטן:*

*כמו כן, כפי שציינו מניתוח ויזואלי, גם במדד ה – MSE ניתן לראות כי הגדלת המסנן בין תמונה 8 ל – 9 הגדילה את השגיאה הריבועית.*

*2.2.*

*לאחר חזרה על הניסוי במשך 100 פעמים עם 20% רעש של מלח פלפל, ה – MSE הממוצע וסטיית התקן שלו הינם:*

**

איור 22: ממוצע וסטיית תקן של ה – MSE בין תמונת הירקות המקורית לתמונה המורעשת לאחר סינון עם מסנני חציון בגדלים שונים

*ה – MSE הממוצע דומה לגרף שהתקבל מהניסוי הבודד הראשוני. עבור המסננים עם שגיאה קטנה, השונות גם קטנה כיוון שבכל מקרה רוב הרעש מסונן ולכן אין הבדלים רבים בין הניסויים. עבור המסננים בעלי הממדים הקטנים, להם השגיאה גדולה יחסית לשאר המסננים, השונות גם היא גדולה היות והרבה מהרעש לא מסונן, דבר הגורם לטווח תוצאות סינון רחב יותר.*

### מסקנות:

בניסוי זה ראינו כי ניתן לסנן רעש מסוג "מלח פלפל" מתמונה שנלקחה מהמטלב בעזרת מסנני חציון ובחנו את השפעת גודל המסנן על טיב הסינון. ראינו כמצופה כי ככל שנגדיל את המסנן נקבל שיפור בסינון עד לגודל מסוים שהחל ממנו נקבל טשטוש של התמונה. תוצאה זו נובעת מכך שכאשר המסנן גדול מידי, ערכי פיקסלים רחוקים שונה ולכן ישפיעו על החציון באופן ניכר. לעומת זאת, עבור מסננים יותק קטנים בהם ערכי הפיקסלים קרוב, למעט הרעש, נקבל כי הרעש כמעט איננו משפיע ותוצאת הסינון טובה.

## ניסוי 3:

### היפותזה:

בניסוי זה נצפה להצליח לזהות את כיוון ההסתכלות מתוך ההנחה כי אין שונות רבה בין מקומי האישון באותו הכיוון. כמו כן, בבדיקת התיוגים של הסרטון נצפה לקבל שרוב הטעויות נמצאות בשכנים קרובים בשל כך שכמות הדגימות בכל זווית איננה גדולה ולכן לא תספיק על מנת לבנות התפלגות מדויקת ובעלת הפרדה בין הכיוונים השונים. בנוסף, עבור כיווני הסתכלות 1,4 נצפה לקבל שגיאה גדולה יותר כיוון שבכיוונים אלה העישון חתוך ומאבד את צורתו העגולה בתמונה ולכן נצפה שזה שיקשה על הזיהוי.

### מתודולוגיה:

מכשור וציוד: מצלמה, תוכנת MATLAB,

מהלך הניסוי: בניסוי זה שמרנו תמונות לפי 4 כיווני הסתכלות וסרטון המכיל את כל הכיוונים הנ"ל. ראשית, ביצענו עיבוד מקדים לתמונות ולאחר מכן כתבנו מספר פונקציות שמטרתן הינה שערוך כיוון ההסתכלות. הפונקציה הראשונה מצאה את מרכז האישון על ידי קורלציה עם תמונת אישון חתוכה. לאחר מכן כתבנו פונקציה שבונה מטריצת כיול מכל המיקומים ובה השתמשנו בפונקציה שלישית לבניית התפלגות לכל כיוון מבט. לאחר שבדקנו את נכונות הפונקציות על התמונות, חזרנו על השערוך עבור הסרטון והשוונו את תיוג האלגוריתם לתיוג ידני.

### תוצאות:

3.1

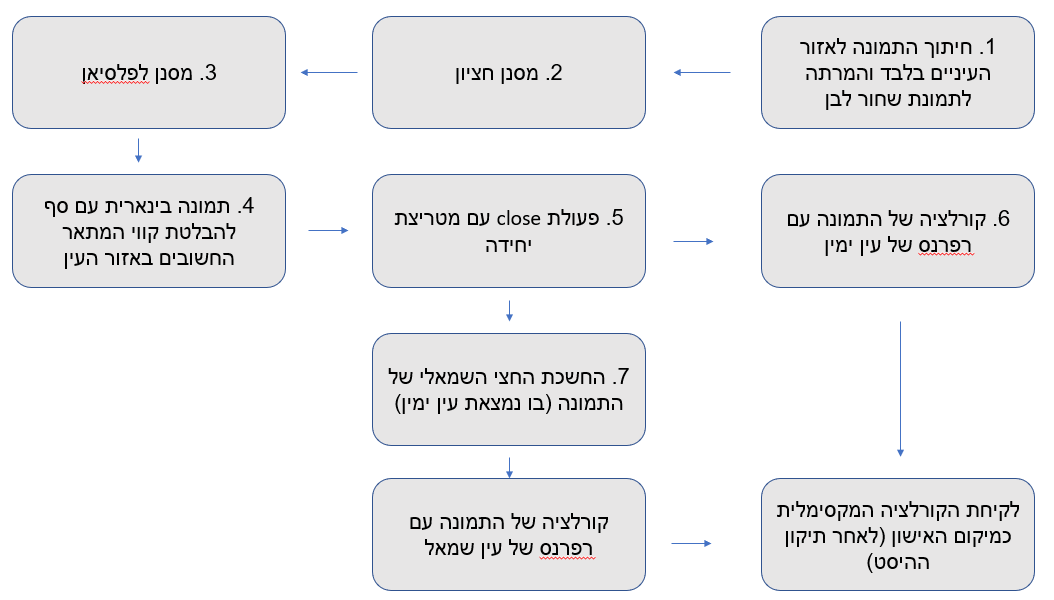
דוגמא לזיהוי האישונים עבור כל כיוון הסתכלות:



איור 23: זיהוי האישונים באמצעות EyePoistion\_FUNC עבור ארבעת כיווני ההסתכלות

ניתן לראות דוגמא לזיהוי האישונים לכל כיוון הסתכלות באמצעות האלגוריתם שיצרנו.

3.2. דיאגרמת בלוקים של האלגוריתם:



איור 24: דיאגרמת בלוקים של אלגוריתם למציאת מיקום האישונים בתמונה

1. חיתוך התמונה לאזור העיניים בלבד והמרתה לשחור לבן. החיתוך נועד למנוע ההפרעות מחלקים לא רלוונטים של התמונה. המרה לשחור לבן מאפשרת פעולות עיבוד בצורה פשוטה יותר.

2. מסנן חציון – לסינון עדין של רעשים. מהווה .

3. מסנן לפלסיאן – מבצע נגזרת שנייה מרחבית על התמונה לקבלת גבולות (קווי מתאר כגון עיניים, גבות וכו'), להם יש נגזרת גדולה יותר כיוון שבאזורים אלה יש מעבר חד בין הגוונים.

4. מעבר לתמונה בינארית – מצאנו סף מתאים (4%) עבורו הפרטים שיש לנו עניין בהם – פרטי אזור העיניים מקבלים את הערך 1, כלומר נצבעים בלבן, וכמה שיותר משאר הפרטים בתמונה מקבלים את הערך 0, כלומר שחור.

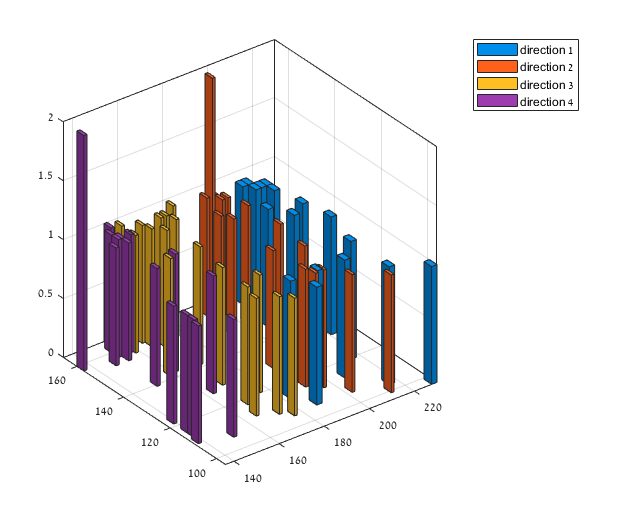
5. פעולת close עם מטריצת היחידה בגודל לשם החלקת התמונה – מילוי רווחים שחורים בלבן באזורי בעלי הפרטים החשובים (עיניים) והפחתת כתמים לבנים באזורים הכהים.

*6. קורלציה בין תמונת הרפרנס החתוכה של עין ימין (רפרנס שנלקח מתמונה שעברה את אותו עיבוד עד שלב זה) עם תמונת האינפוט, מתוך הנחה כי הקורלציה המקסימלית תתקבל באזור בו התמונות זהות (או דומות). חילוץ הקואורדינטות בהן הקורלציה מקסימלית כמיקום האישון הימני (עם היסט קבוע, כיוון שהקואורדינטות הנתונות הן של הפינה הימנית התחתונה של תמונת הרפרנס ואנו מניחים שהאישון במרכז התמונה הנ"ל).*

*7. החשכה של החצי השמאלי של התמונה – היות והעפעף של עין שמאל מסתיר רבות מהעין, ישנה קורלציה יותר גבוהה בין הרפרנס של עין שמאל עם עין ימין, ולכן ביצענו קורלציה של הרפרנס של עין שמאל רק עם החצי הימני של התמונה (בו מופיעה עין שמאל), כלומר עם התמונה בה החלק שמאלי מוחשך. חילוץ הקואורדינטות של אישון שמאל באותו אופן כמו שביצענו עבור אישון ימין.*

*3.3.*

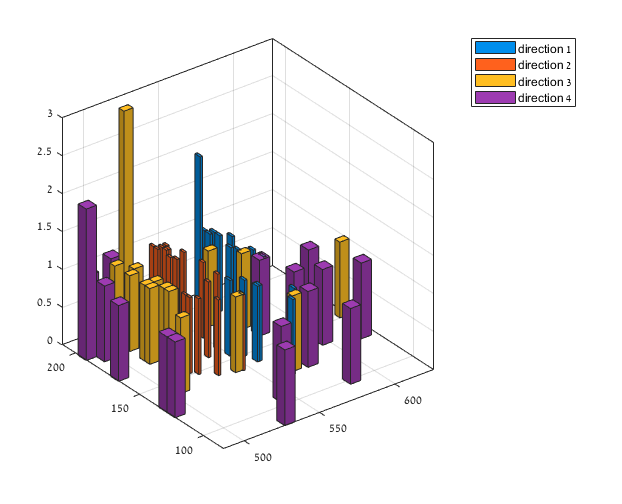
*היסטוגרמת עין ימין:*

**

איור 25 – היסטוגרמת מטריצת הכיול עבור עין ימין

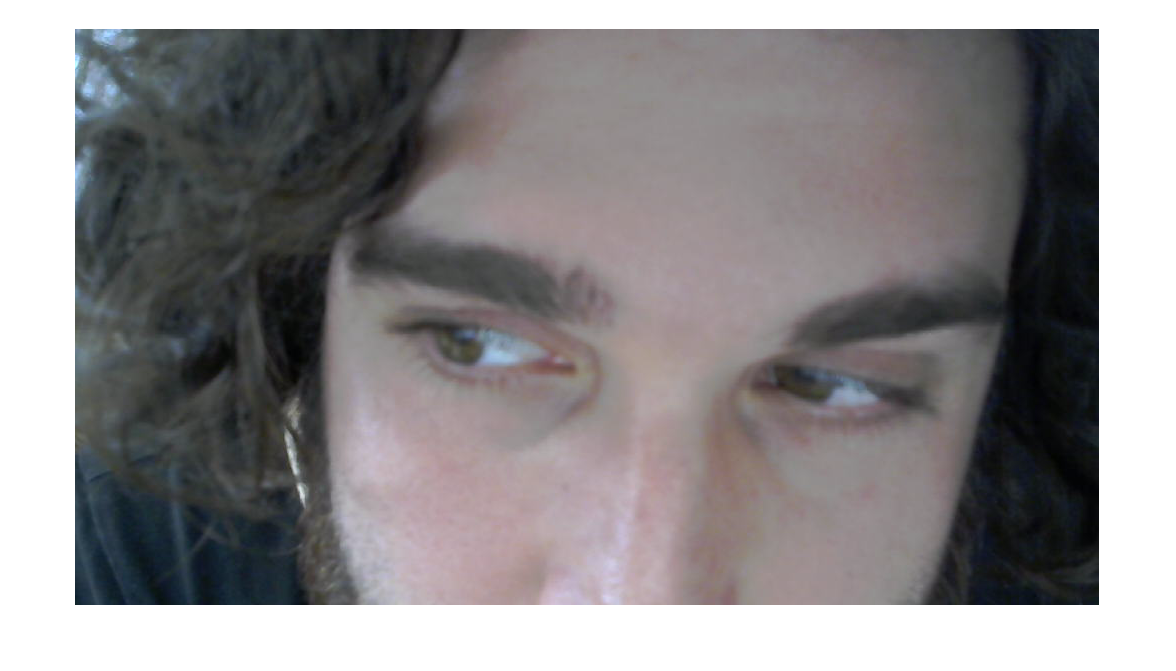
כפי שניתן לראות, ישנה הפרדה ברורה (לא מוחלטת) בין ההיסטוגרמה של כל כיוון כרצוי. ניתן לראות כי בכל ארבעת הכיוונים, ישנה צפיפות גבוהה באותו ערך של ציר y (באזור y=160), בהתאם לכך שמה שהשתנה הוא הכיוון האופקי אליו מסתכלים ולא האנכי. ההבדלים בשינוי הכיוון בציר האנכי יכולים להיות מוסברים משינוי בכיוון ההתבוננות (בצירה האנכי) של הנמדד או משגיאת האלגוריתם לזיהוי האישון.

היסטוגרמת עין שמאל:



איור 26: היסטוגרמת מטריצת הכיול עבור עין שמאל

ניתן לראות כי עבור עין שמאל ישנו פיזור גדול יותר של הערכים, בפרט עבור כיוון 4 ו – 3. ניתן לראות כי בעין שמאל העפעף מכסה חלק גדול יותר של העין, דבר המקשה על זיהוי האישון:



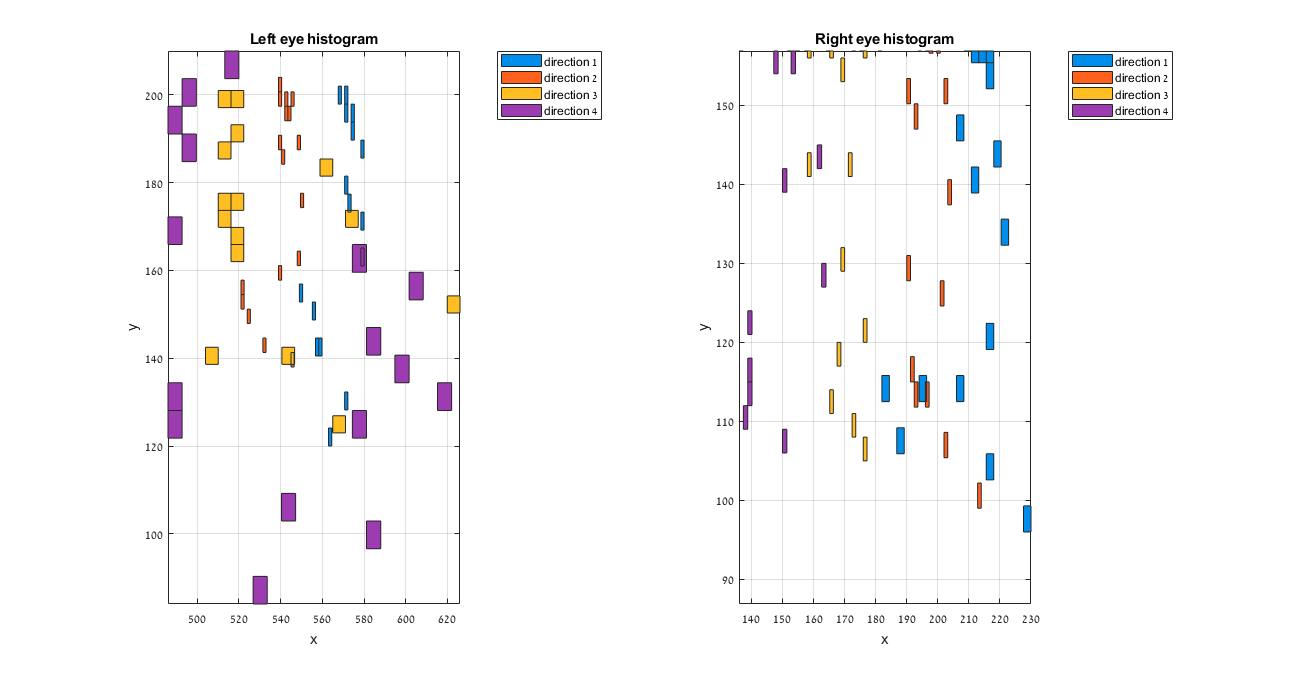
איור 27 – דוגמה לתמונה מכיוון 4

כפי שרואים, העפעף של עין שמאל מסתיר חלק ניכר מן העין, מה שמוביל לשגיאה בזיהוי.

3.4.

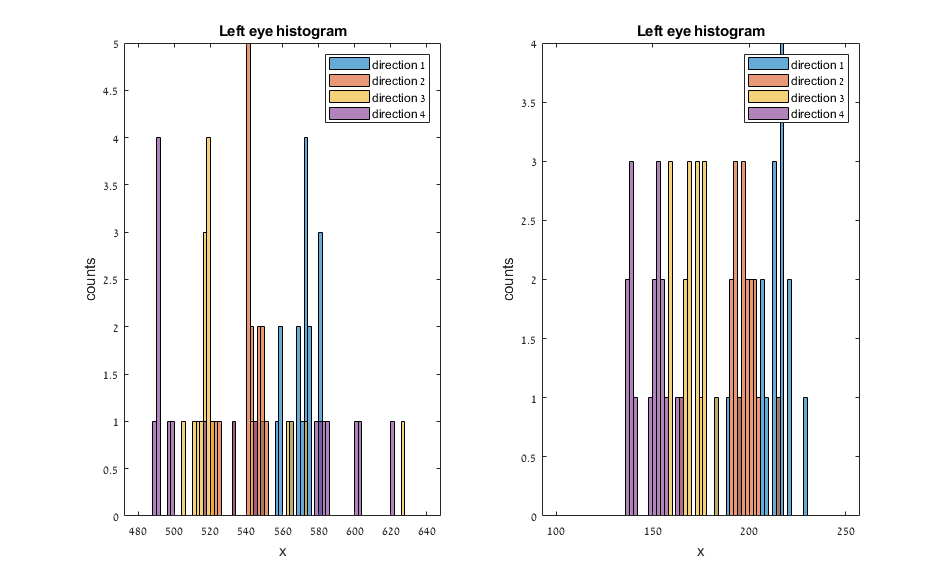
3.4.1

ראשית, נתבונן על ההיסטוגרמות המרחביות של שתי העיניים ממבט על:



איור 28: היסטוגרמת עין ימין (צד ימין) והיסטוגרמת עין שמאל (צד שמאל)

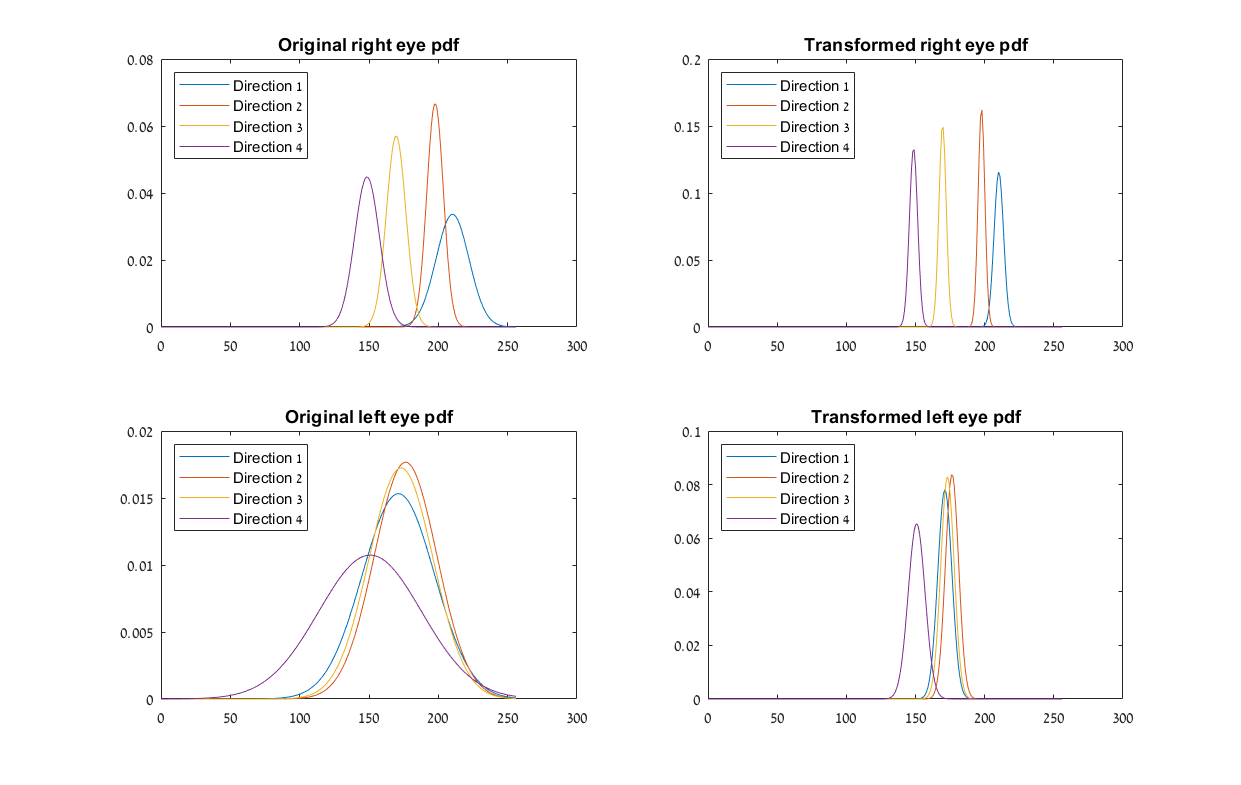
בניסוי זה, הבטנו על ריבועים הנמצאים באותו קו אנכי, אך במיקומים אופקיים שונים. אם כך, הגיוני כי השינוי המהותי בין ההתבוננויות בכיוונים השונים יהיה ניכר בעיקר בציר איקס (האופקי). באיור זה ניתן לראות בבירור, בעיקר עבור עין ימין את נכונות טענה זו, ולכן כדי לבצע הפרדה סטטיסטית בין הכיוונים, נוריד את ממד הבעיה מבעיה דו ממדית לבעיה בציר איקס בלבד. היסטוגרמות הכיוונים לכל עין בציר איקס:



איור 29: היסטוגרמות העיניים עבור ציר איקס (ציר אופקי)

מהיסטוגרמות אלה, ניצור התפלגות עבור כל כיוון הסתכלות. מתוך ההנחה שמיקום ההסתכלות אליו מכוונים לא משתנה, נצפה כי אם היו לנו מספיק דגימות, התפלגותן הייתה נורמלית סביב מרכז כיוון ההסתכלות. באיור 21 ניתן לראות כי במקרה שלנו מספר הדגימות קטן וההתפלגות אינה באמת נורמלית, אך נשערך התפלגות נורמלית מהמדידות. לאחר ניסיון שימוש בהתפלגויות הנורמליות המתקבלות ראינו כי השונות גדולה מדי ויש חיתוך גדול מדי בין ההתפלגויות ולכן לקחנו כל pdf ששוערך מהנתונים והגדרנו התפלגות נורמלית עם שונות שהינה שורש השונות שהתקבלה מהשערוך ואותו ממוצע. עבור כל תמונה בה נזהה אישון, על פי מיקומו, נבחר את הכיוון לפי ההתפלגות בעלת ההסתברות הגבוהה ביותר מתוך ההתפלגויות הנורמליות ששערכנו.

פונקציות ההתפלגות הנורמליות שהתקבלו מהנתונים, והתיקון באמצעות הקטנת השונות:



איור 30: פונקציות ההתפלגות הנורמליות שהתקבלו מהנתונים עבור כל כיוון הסתכלות לפני ואחרי הטרנספורמציה

ניתן לראות מספר דברים מאיור זה. ראשית, לפני הטרנספורמציה, ההתפלגויות עבור שתי העיניים היו בעלות חפיפות רבות ולכן צפוי כי לא יעבדו כראוי לשם סיווג כיוון ההסתכלות. לאחר הטרנספורמציה בעין ימין יש הפרדה טובה יותר משמעותית בין פונקציות הפילוג עבור כל כיוון. שנית, כיוון שהשוניות עבור עין שמאל גדולות מאוד ביחס להבדלים בין הממוצעים שלהן, ההפרדה בין פונקציות הפילוג גרועה מאוד וגם הקטנת השונות באמצעות שימוש בשורש השונות כפרמטר ההתפלגות לא תיקנה בעיה זו. כיוון שאנחנו משתמשים במקסימום בין ההסתברות של שתי העיניים, סביר להניח כי עין ימין היא זו שלפיה יקבע לרוב כיוון ההסתכלות.

3.4.2

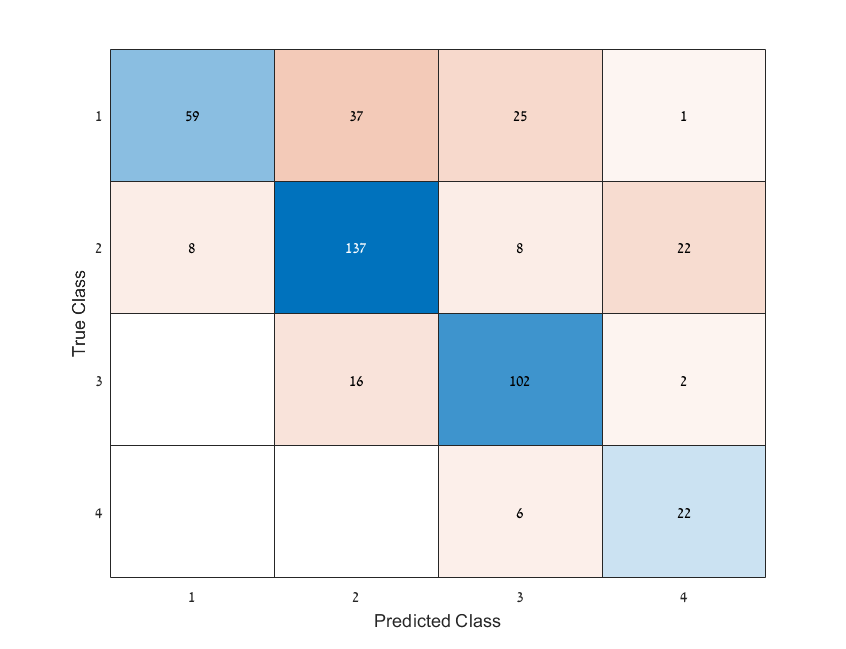
טבלה 1 – תוצאות חיזוי כיוון ההסתכלות

|  |  |
| --- | --- |
| כיוון הסתכלות | תוצאת האלגוריתם |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |

ניתן לראות כי עבור התמונות אותן בדקנו החיזוי הינו נכון. מהדיון בסעיפים הקודמים לגבי בעייתיות הזיהוי של עין שמאל, הסקנו כי תוצאות החיזוי עבור עין זו יהיו פחות טובות. כאשר בדקנו את מימוש הפונקציה עבור כל עין בנפרד ראינו כי אכן החיזוי של עין שמאל פחות מדויק בצורה משמעותית. מתוך ההנחה כי אדם מסתכל לאותו הכיוון עם שתי העיניים הוספנו תנאי הבודק איזה חיזוי עבור כל עין קיבל הסתברות גבוהה יותר ובחרנו בחיזוי זה להיות כיוון ההסתכלות המשוערך של שתי העיניים.

3.6

מטריצת הבלבול המציגה את כיווני ההסתכלות שנמדדו ביחס לתיוג באמצעות הסגמנטציה הידנית:



איור 31: מטריצת הבלבול של כיוון ההסתכלות שאותר ביחס לתיוג הידני

ניתן לראות כי תוצאות החיזוי יחסית מדויקות וכי השגיאות הן בעיקר היסט לכיוון שמאל – כלומר התיוג השגוי הוא של כיוון הנמצא שמאל יותר מכיוון ההסתכלות המתויג. ייתכן כי תוצאה זו נובעת משינוי במיקום הראש בתמונות הקליברציה לעומת הוידאו. בנוסף, ישנן תמונות בהן הזיהוי שגוי כתוצאה מתמונה שנלקחה במהלך שינוי כיוון ההסתכלות או שנלקחה בעת מצמוץ. כיוון שהשתמשנו בקורלציה למציאת מיקום האישונים, בעת מצמוץ יזוהה אישון איפשהו בתמונה אך לא במקום המתאים.

### מסקנות:

בניסוי זה כתבנו אלגוריתם לזיהוי מיקומי האישונים ובעזרתם זיהוי כיוון ההסתכלות. על ידי מימוש התאוריה שלמדנו ביצענו עיבוד מקדים לתמונות על מנת שנוכל לבצע את הזיהוי בצורה קלה ומדויקת יותר. ראינו כמצופה כי בלקיחת תמונת רפרנס של אישון, נקבל את הקורלציה הגבוהה ביותר במרכז האישון של התמונה הנבדקת (כאשר הרפרנס נלקח מתמונה אחרת). כמו כן, ראינו כי ניתן לממש את האלגוריתם הרצוי באמצעות בניית התפלגות לאחר הקטנת השונות, מה שהביא לתוצאות טובות. נצפה כי ניתן יהיה לשפר את תוצאות האלגוריתם באמצעות צילום של יותר תמונות על מנת לקבל הסתברות מדויקת יותר.

## ניסוי 4:

### היפותזה:

בניסוי זה ננסה לשערך את מהירות ההליכה של האדם באמצעות מרקרים. בניגוד לניסוי הקודם, בניסוי זה צורת המרקר עגולה ויותר מדויקת ובהנחה כי אין גורמים שיסתירו אותה (כמו העפעף בניסוי הקודם), נצפה לקבל זיהוי מדויק של המרקרים. בשל המהירות ההליכה האיטית נצפה לקבל תמונות יחסית ברורות ברובן מה שיעזור לזיהוי. מתוך ההנחה כי בעת הליכה החלק התחתון של הרגל זז בציר האופקי יותר מאשר החלק העליון, נצפה לראות כי למרקר הרביעי (התחתון) יש טווח ערכי x גדול יותר מלאחרים. כמו כן, נצפה שלא יהיה שינוי מהותי בציר הY בין הפריימים. בנוסף, בניתוח הזוויות של הירך והברך נצפה לקבל שזווית הירך הינה זווית חדה ובעלת טווח ערכים קטן ואילו לברך נצפה לקבל זווית כהה בעלת טווח ערכים קטן כתוצאה מתנועת ההליכה.

### מתודולוגיה:

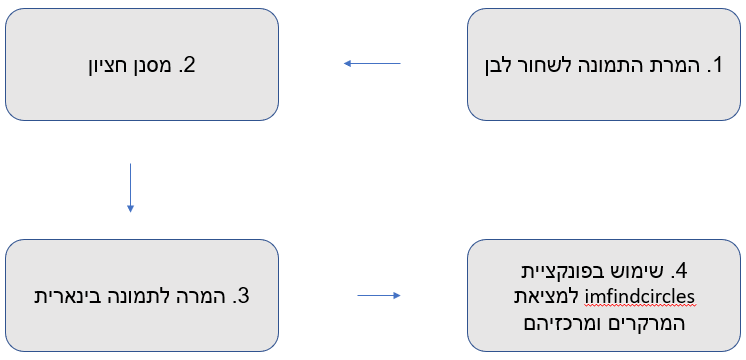
מכשור וציוד: מצלמה, תוכנת MATLAB,

מהלך הניסוי: בניסוי זה הצבנו 4 מרקרים על גבי הירך והשוק ברגל ימין. ביצענו צילום מאותה הזווית של האדם הולך במהירות קבועה של 2 קמ"ש למשך חצי דקה. לאחר מכן ביצענו ניתוח של המידע במטרה לשערך פרמטרים מתוך סרטון זה.

### תוצאות:

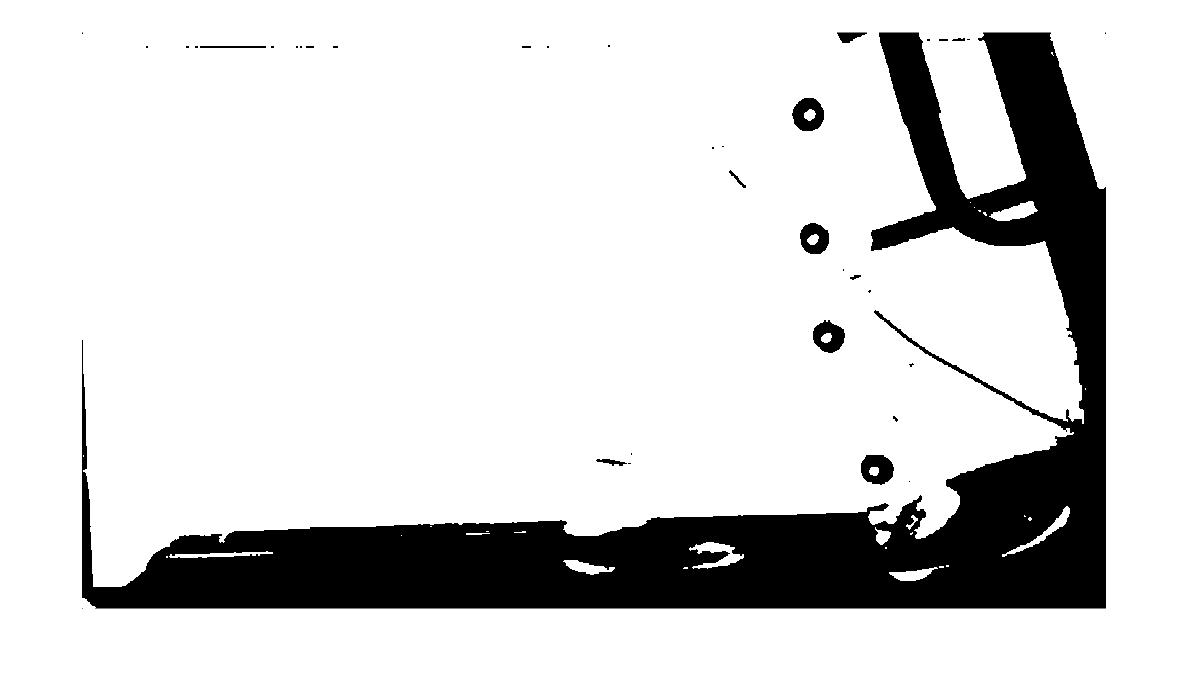
4.1

4.1.1 דיאגרמת בלוקים של האלגוריתם למציאת המרקרים:



איור 32: דיאגרמת בלוקים של אלגוריתם למציאת המרקרים

1. המרת התמונה לשחור לבן לשם פישוט תהליך העיבוד.
2. מסנן חציון להחלקה קלה של התמונה
3. המרת התמונה לתמונה בינארית – מצאנו סף עבורו המרקרים בשחור, מרכזם לבן, וסביבתם לבנה גם היא:



איור 33: דוגמא לתמונה בינארית מהסרטון

ניתן לראות בבירור את המרקרים וכי צורתם הינה של עיגול. לכן בחרנו להשתמש באלגוריתם למציאת עיגולים בתמונה.

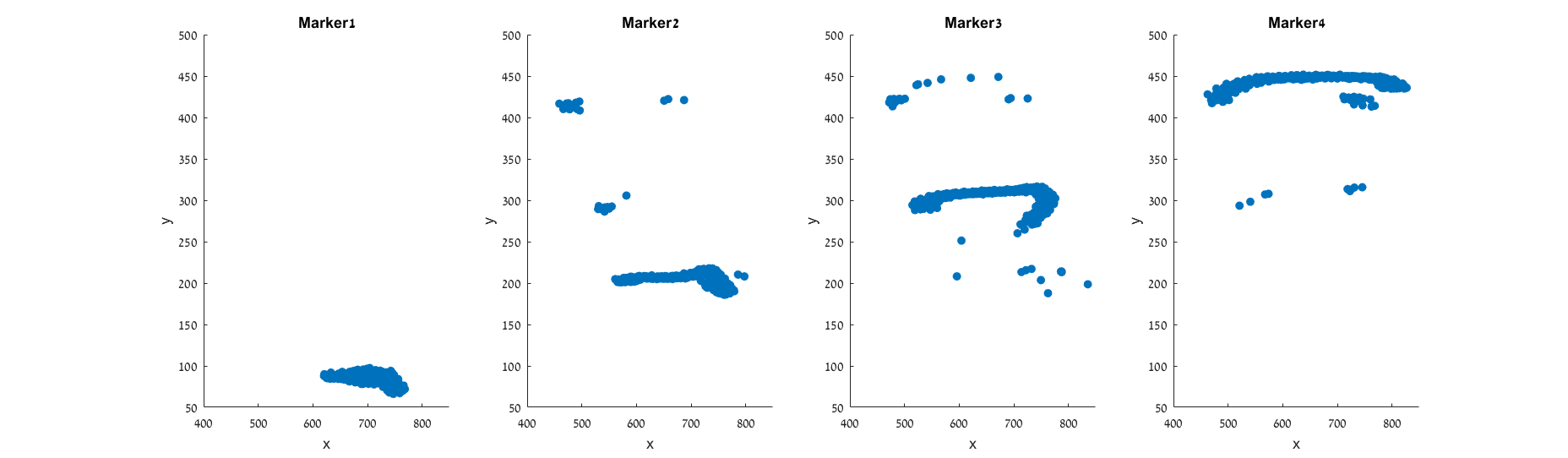
1. שימוש בפונקציית imfindcircles למציאת המרקרים מתוך התמונה הבינארית:

פונקציה זו משתמשת ב - שהינה טכניקה נפוצה בעיבוד תמונה. במקרה של מציאת עיגול הפונקצייה מבצעת את הפעולות הבאות עבור כל פיקסל:

1. חישוב הגרדיאנט.
2. לפי טווח הרדיוסים שנתנו, מציב כמרכז מעגל מספר נקודות שמרחקן בטווח הרדיוסים שהזנו כאינפוט ובכיוון הגרדיאנט.

לבסוף, נספרים מס' הפעמים שמתקבל כל מרכז מעגל מתוך *הפעולות הנ"ל על כל הפיקסלים. באמצעות פרמטר הרגישות נקבעים מס' המרכזים שעוברים את הסף לקבל הגדרה של מרכז מעגל. הגדרת הפולריות ככהה קובעת כי מחפשים מעגל כהה יותר מהרקע שסביבו, כלומר בחרנו לחפש את המעגלים השחורים ואת מרכזיהם.*

4.1.2

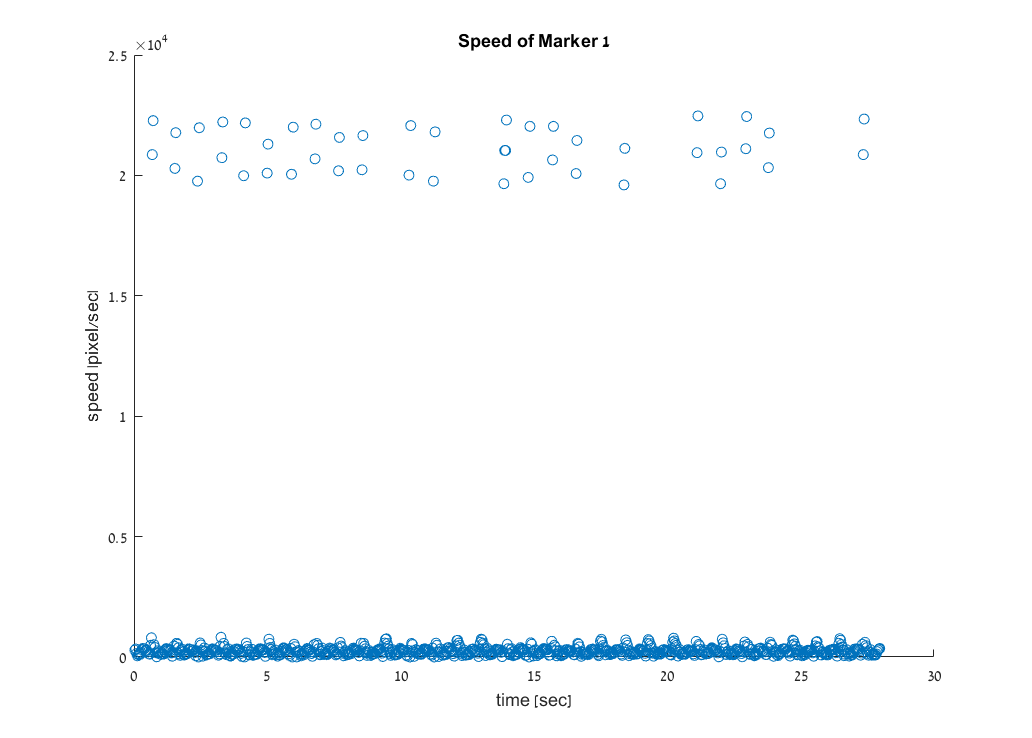


איור 34: מיקומי המרקרים במרחב

ניתן לראות כי לכל מרקר יש גובה אופייני בהתאמה לכך שהוצב בגובה אחר על הרגל. נציין כי הגובה בו הוצב המרקר על הרגל הוא ביחס הפוך להופעה שלו בגרף זה, כלומר מרקר 1

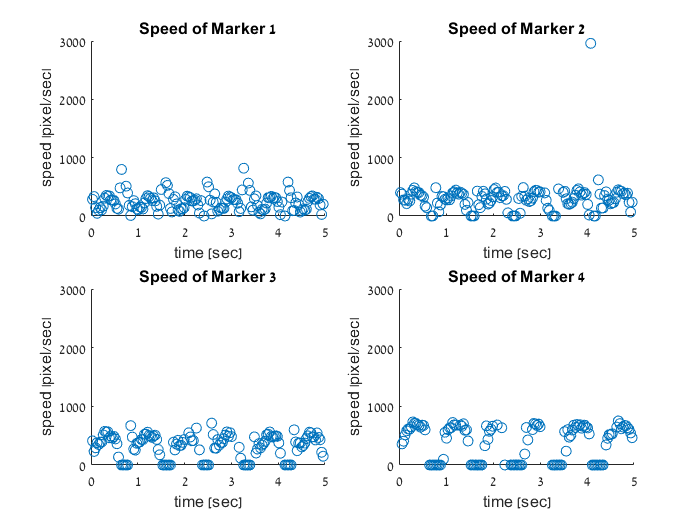
(Marker 1) הינו המרקר שהוצב הכי גבוה על הרגל. כמו כן, ניתן לראות כי מרקר זה בעל טווח התנועה האופקי (ציר x) הקטן ביותר, וזאת בהתאמה לכך שבהליכה, השוק (מרקר 4) עושה את התנועה האופקית הארוכה ביותר וקצהו העליון של הירך (מרקר 1) עושה את התנועה האופקית הקטנה ביותר.

4.2



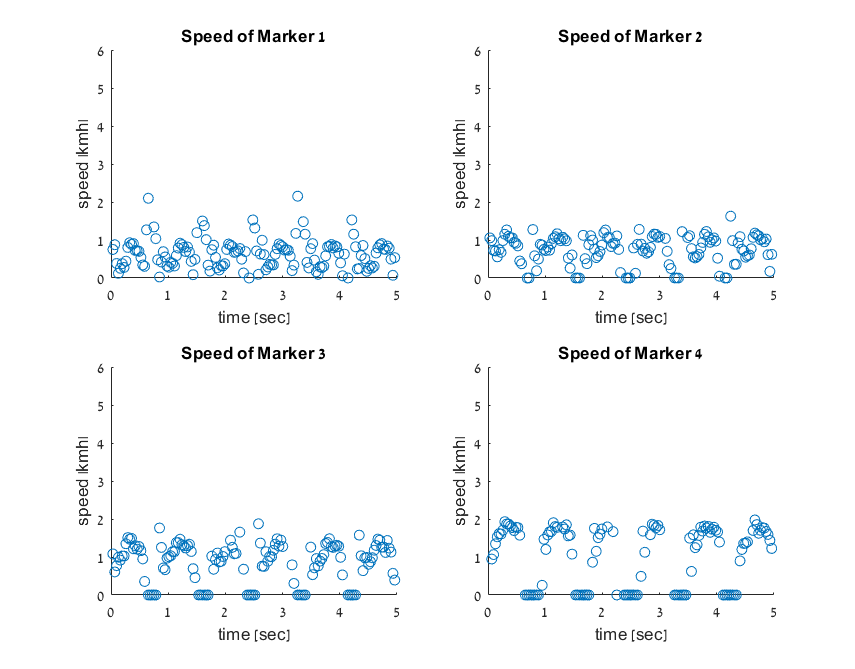
איור 35: דוגמא למהירות מרקר 1 כפונקציה של הזמן

ניתן לראות כי יש מספר outliers שנובעים משגיאות זיהוי של המרקרים ולכן נגביל את תחום התצוגה לתחום הרלוונטי:



איור 36: מהירות המרקרים בפיקסל לשנייה

ראשית, ניתן לראות כי ככל שהמרקר יותר נמוך על הרגל כך מהירותו הממוצעת גדולה יותר. שנית, נשים לב כי המהירות היא פונקציה אוסילטורית. נשער כי המהירות המקסימלית המופיעה בגרף היא בקרוב מהירות ההליכון (2 קמ"ש וההמרה המתאימה לפיקסל לשנייה), בעוד בזמנים שהרגל באוויר, כיוון שהגוף נע אחורה על ההליכון והרגל נעה קדימה באוויר המהירות (היחסית, בתמונה) בזמן זה היא בקרוב אפס. נראה גרפים אלה עם ההמרה למהירות בקמ"ש:

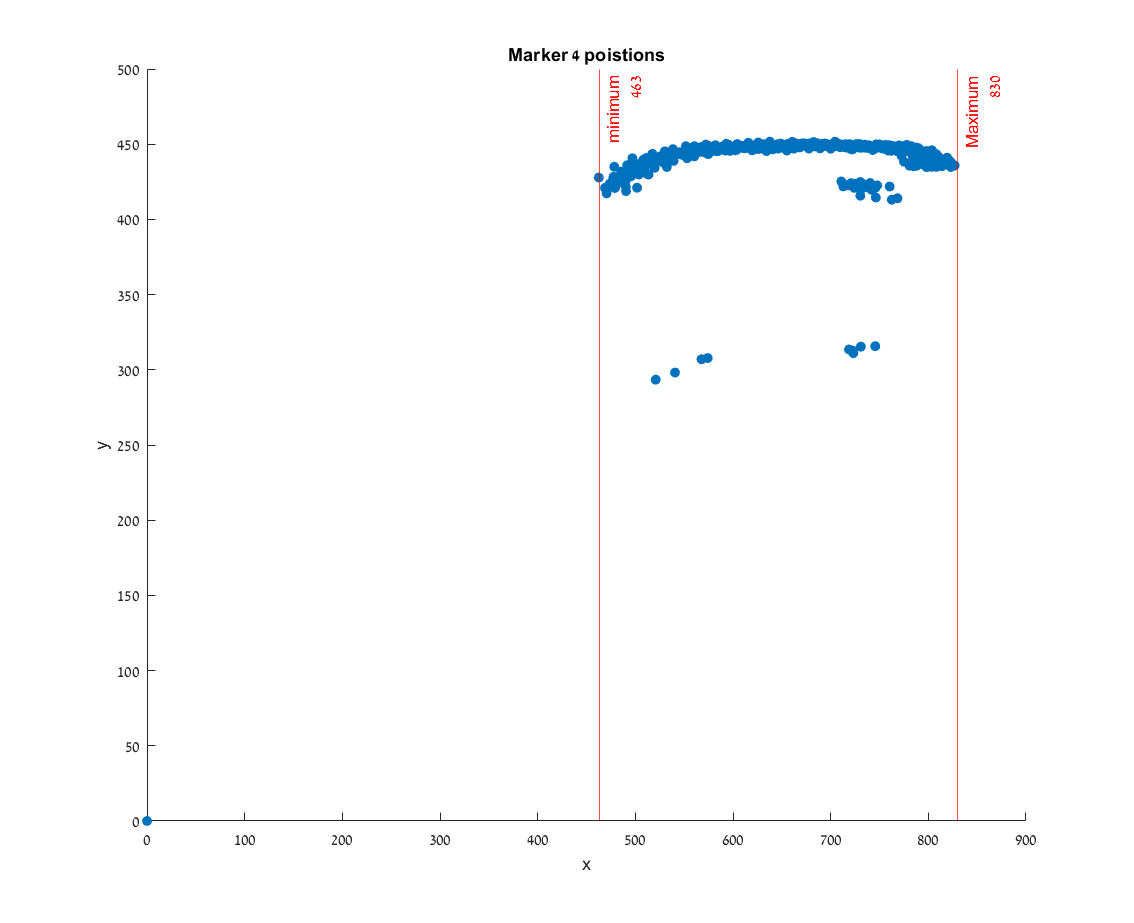


איור 37: מהירות המרקרים בקמ"ש

כפי שניתן לראות, בהתאמה להשערתנו המהירות המקסימלית באוסילציה (עבור מרקר 4 הקרוב לרצפה) הינה בערך 2 קמ"ש. את ההמרה ביצענו ע"י מדידת רדיוס המרקרים בפיקסלים והמרה ליחידות של ס"מ באמצעות ההנחה כי המרקר ברדיוס 1 ס"מ.

4.3.

מהתבוננות באיור 37 ניתן לראות כי ישנה מחזוריות בפונקציית המהירות. מתוך כך ומהתבוננות בסרטון, נניח כי אין שינוי מהותי במיקום ההליכה על ההליכון עצמו. אם כך, ניתן להסיק כי עבור מרקר 4, ההפרש בין קואורדינטות האיקס המקסימלית והמינימליות שלו (עם התחשבות ב -Outliers) מייצגות את גודל הצעד:

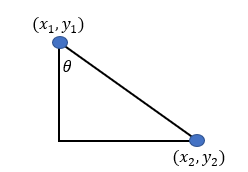


איור 38: מיקומי מרקר 4 במרחב

ניתן לראות כי סימנו את המקסימום והמינימום של הקבוצה הצפופה כך שלא נקבל שגיאה כתוצאה מ – outliers:

*כלומר גודל הצעד הוא . ההמרה בוצעה ע"י כך שקוטר המרקר שמדדנו בפיקסלים הינו , והנחנו כי גודלו בס"מ הינו 1 ס"מ.*

*4.4 התייחסנו לזווית הירך בתור הזווית בין הישר העובר בין מרקר 1 למרקר 2 לציר האנכי:*

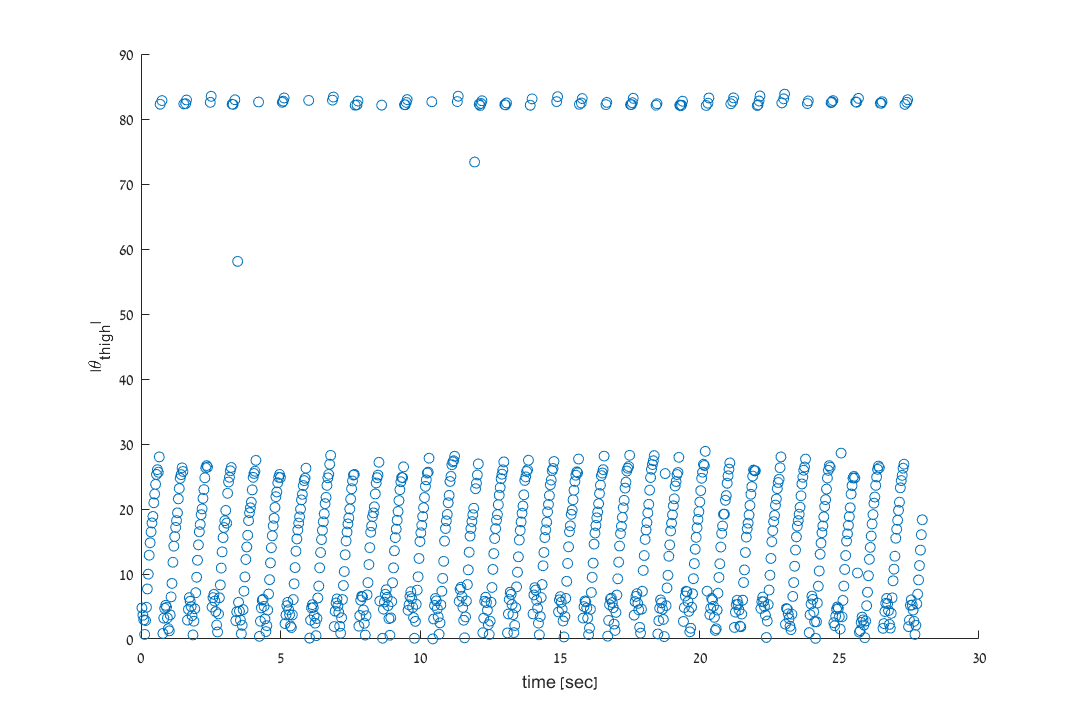
**

איור 39: תרשים המשולש

כאשר הנקודות הכחולות באיור מייצגות את שני המרקרים העליונים, והזווית הנמדדת מסומנת.

*את הזווית חישבנו כך:*

*גרף הזווית בערך מוחלט כפונקציה של הזמן:*

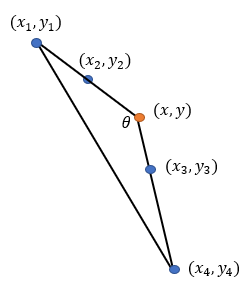
**

איור 40: זוויות הירך בערך מוחלט כפונקציה של הזמן

כפי שניתן לראות, זווית הירך איננה גדולה ומתנהגת בצורה אוסילטורית מסוימת כצפוי. נציין כי מהתבוננות בסרטון נראה כי המרקרים לא נמצאים בדיוק באותו ציר לאורך הרגל, דבר שגורם לשגיאה בשימוש בהם לחישוב זווית הירך.

4.5

את זווית הברך קבענו להיות הזווית בין הישר העובר בין מרקרים 1 ו - 2 , לישר העובר בין מרקרים 3 ו – 4. תחילה, מצאנו את החיתוך בין המשכי הישרים, ואז השתמשנו במשפט הקוסינוסים לחישוב הזווית:

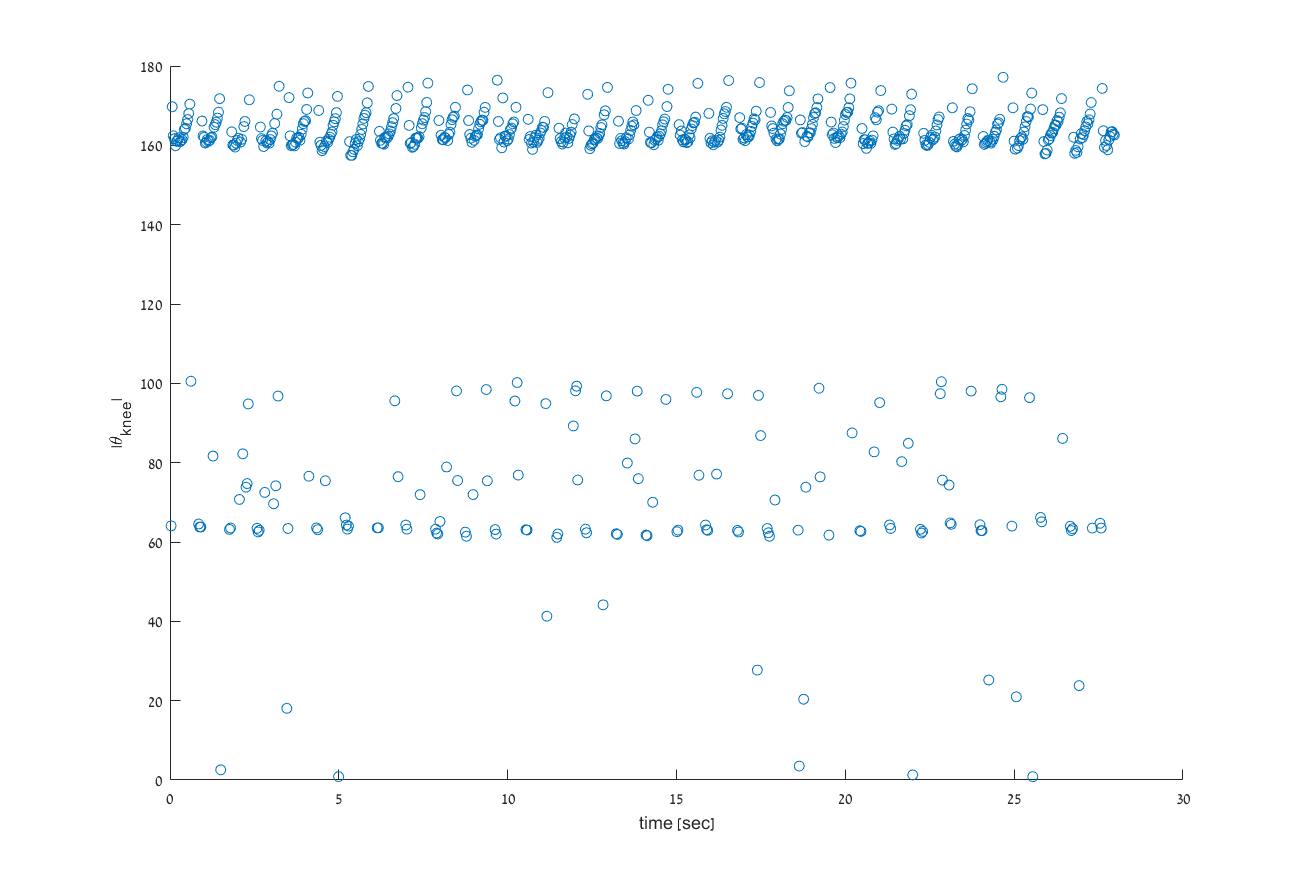


איור 41: תרשים המשולש

כאשר הנקודות הכחולות באיור מייצגות את שני המרקרים העליונים, והזווית הנמדדת מסומנת.

*בדרך זו מצאנו את הנקודה המסומנת בכתום באיור 41. לאחר מכן השתמשנו במשפט הקוסינוסים:*

*גרף הזווית בערך מוחלט כפונקציה של הזמן:*



איור 42: : זוויות הברך בערך מוחלט כפונקציה של הזמן

כמו עבור זווית הירך, גם כן נראה התנהגות יחסית מחזורית בזמן בהתאמה לתנועה מחזורית של הרגל. במקרה זה מרבית הזוויות שחושבו הינן כהות, כצפוי עבור הזווית בין הירך לשוק.

### מסקנות:

### בניסוי זה ביצענו זיהוי של מרקרים על גבי רגל ימין של אדם הולך במהירות קבועה. למדנו על טרנספורם Hough והשתמשנו בו למציאת המרקרים בסרטון. ראינו כי המרקר העליון ביותר ביצע את התנועה האופקית הקטנה ביותר ואילו המרקר התחתון ביצע את התנועה האופקית הגדולה ביותר תוך שמירה על הגובה (ציר y) בשניהם, כפי שציפינו מתוך האנטומיה של הליכה. בנוסף, למדנו כיצד להמיר גודל של אובייקט בסרטון מפיקסלים לס"מ והשתמשנו בזה על מנת לבדוק את המהירות שהאלגוריתם שלנו שערך. כמו כן, שערכנו את זווית הברך וזוויות הירק מתוך מיקומי המרקרים. ראינו כי כתוצאה מטווח תנועה קטן של הרגל, טווח הזוויות קטן גם הוא, והן זוויות חדות וכהות בהתאמה כמצופה.

# מסקנות כלליות

במעבדה זו רכשנו כלים בסיסיים בעיבוד תמונה. ביצענו שיפור איכות לתמונה, זיהוי גבולות, זיהוי אובייקטים ושימוש במידע שניתן לחלץ מהתמונה למטרות שונות. בניסוי הראשון חילצנו מתוך תמונת גרגרי אורז על רקע כהה גרגרים בעלי אוריינטציה לכיוון שמאל. לשם כך, ראשית מצאנו את רקע התמונה באמצעות פעולת erosion כך שהגרגרים יוסרו. לאחר מכן מצאנו את גבולות הגרגרים באמצעות מסנן נגזרת מסדר שני ולבסוף השתמשנו בגבול של כל גרגר כיחידה נפרדת כדי לקבוע את האוריינטציה שלו. בניסוי השני למדנו על שימוש במסנן חציון כדי להפחית רעש מלח ופלפל. השימוש במסנן זה נובע מהנחה שבסביבה מספיק קרובה, הפיקסלים השייכים לתמונה לא מורעשת הם בעלי ערכים קרובים ולכן מסנן חציון יציב לא יושפע מערכים קיצוניים אלא מכמות הערכים הקרובים אחד לשני בסביבה עליה עובר הפילטר. ראינו כי הנחה זו נכונה עד כדי גודל מסוים ולכן צריך לאזן בבחירת גודל הפילטר בין כמות הרעש שמנסים לסנן (הגדלת גודל המסנן) לבחירת סביבה בה פיקסלי התמונה המקורית אכן קרובים מספיק אחד לשני. בניסוי 3 יצרנו אלגוריתם לזיהוי כיוון הסתכלות של אדם על מסך מתוך ארבעה כיווני הסתכלות אפשריים. תחילה, ביצענו זיהוי של מיקום האישונים בתמונה. נוכחנו לראות כי בתמונות אמיתיות יכולים להופיע ארטיפקטים כמו השתקפות אור על העיניים המקשים על מימושים קלאסיים ישירים. לאחר זיהוי האישונים, השתמשנו במספר תמונות ליצירת מטריצת קליברציה המורכבת מזיהוי האישונים במספר תמונות. ראינו כי התפלגותן הייתה לעיתים עם שונות גדולה וכי ייתכן כי שימוש במספר רב של תמונות כבסיס למטריצת קליברציה יכול לאפשר סטטיסטיקה יותר טובה. השתמשנו במטריצת הכיול כדי לקבוע את כיוון ההסתכלות בכל פריים של סירטון. ראינו כי ישנם שני גורמים שיכולים לפגוע בזיהוי. אחד קשור לתמונה עצמה – מעבר בין כיווני הסתכלות, מצמוץ. השני קשור לגישה בה נקטנו המתבססת על כך שהאדם המתבונן משאיר את פניו באותו מקום בתמונה ומזיז רק את עיניו. ייתכן שכדי לבצע זיהוי טוב יותר צריך לממש אלגוריתם שלא מתבסס על הנחה זו. בניסוי האחרון ביצענו זיהוי של מרקרים עגולים בתמונה ושימוש במיקומים שלהם למיצוי מאפיינים כגון מהירות הרגל, גודל הצעד, זוויות בין חלקים שונים של הרגל. לזיהוי המרקרים השתמשנו בטרנספורמציית hough, הנפוצה בעיבוד תמונה לזיהוי אובייקטים גאומטריים ומאפיינם גרמיים בתמונה, ובפרט במקרה שלנו זיהוי עיגולים.

# נספחים

**קוד המטלאב:**

הקוד הראשי:

%% post

%% 1.

%1.1

rice = imread('rice.png');

figure; imshow(rice);

%1.2

%histogram of the image

figure; imhist(rice); xlabel('gray scale'); ylabel('count')

%1.3

% convertion to binary image

BinRice = im2bw(rice,115/255); figure; imshow(BinRice);

%1.4

figure; imshow(rice);

imdistline; %measuring radius of rice grain

% performing open and close on the photo to recive the background photo

disk = strel('disk',13,8);

background\_rice = imopen(rice,disk);

background\_rice = imclose(background\_rice,disk);

figure; imshow(background\_rice)

% original image without the background

no\_background = rice - background\_rice;

figure; imshow(no\_background)

%1.5

% repeating 1.2 and 1.3 on the image without background

figure; imhist(no\_background)

bin\_noback = im2bw(no\_background,50/255);

figure; imshow(bin\_noback)

%edge detection:

%laplacian

laplace = ones(3,3);

laplace(2,2) = -8;

rice\_edge1 = imfilter(bin\_noback,laplace);

figure; imshow(rice\_edge1)

% dilation and removal of binary image from the expended binary image to

% recive edges

expended\_im = imdilate(bin\_noback,ones(3,3));

figure; subplot(1,2,1); imshow(bin\_noback); title('original')

subplot(1,2,2); imshow(expended\_im); title('expended')

rice\_edge2 = expended\_im - bin\_noback;

figure; subplot(1,2,1); imshow(rice\_edge1); title('laplacian')

subplot(1,2,2); imshow(rice\_edge2); title('expention')

%1.6

% removing rice grains that are partialy outside the image

no\_borders = imclearborder(rice\_edge2);

figure; imshow(no\_borders)

% detecting amount of rice grains

[~,num\_shapes] = bwlabel(no\_borders,8);

% using negative laplacian to diffrintiate between the grains

N\_lp = -1\*laplace;

rice\_edge3 = imfilter(bin\_noback,N\_lp);

figure; imshow(rice\_edge3)

% removing rice grains that are partialy outside the image

no\_borders = imclearborder(rice\_edge3);

figure; imshow(no\_borders)

% detecting amount of rice grains

[groups,num\_shapes2] = bwlabel(no\_borders,8);

%1.7

% centroids of rice grains:

s = regionprops(no\_borders,'centroid');

centroids = cat(1,s.Centroid);

% presenting the centroid on the picture

figure; imshow(no\_borders)

hold on

plot(centroids(:,1),centroids(:,2),'r\*')

hold off

%mask of left inclined grains

mask = zeros(256,256);

for i=1:num\_shapes2

[y,x] = find(groups == i);

if centroids(i,2) > max(y(x == min(x)))

uni\_y = unique(y);

for j = 1:length(uni\_y)

k = find(y == uni\_y(j));

mask(uni\_y(j),x(min(k)):x(max(k))) = 1;

end

end

end

figure; imshow(mask)

%1.8

% the original rice grains photo with only left inclined grains

figure; imshow(uint8(double(rice).\*mask))

%% 2.

%2.1

Vegt = imread('vegetables.tif'); % vegetable iamge

Nveg = imread('Nvegetables.tif'); % vegtable image with salt and pepper noise

% presenting the images

figure; subplot(1,2,1); imshow(Vegt); title('original vegetables')

subplot(1,2,2); imshow(Nveg); title('salt&pepper vegetables')

% vectors for midean filter dimensions

m = [1 1 2 3 3 4 4 5 7];

n = [2 3 2 3 4 4 5 5 7];

MSE = zeros(1,length(m));

%filtering the noised veg image with median filter of different sizes,

%presenting the results and calculating the MSE from the original veg image

figure;

for i=1:length(m)

I = medfilt2(Nveg,[m(i) n(i)]);

subplot(3,3,i); imshow(I); title([num2str(m(i)) 'X' num2str(n(i)) ' meadian filter'])

MSE(i) = immse(I,Vegt);

end

% MSE of filtered images

figure; scatter(1:9,MSE, 'filled'); xlabel('index'); ylabel('MSE')

%2.2

% repeating the previous experiment 100 times and presenting the mean and

% std of the MSE

MSE\_2 = zeros(100,9);

for j=1:100

Nveg2 = imnoise(Vegt,'salt & pepper',0.2);

for i=1:length(m)

I = medfilt2(Nveg2,[m(i) n(i)]);

MSE\_2(j,i) = immse(I,Vegt);

end

end

mean\_MSE = mean(MSE\_2);

std\_MSE = std(MSE\_2);

figure; subplot(1,2,1); scatter(1:9,mean\_MSE,'filled'); xlabel('index'); ylabel('mean'); title('mean MSE')

subplot(1,2,2); scatter(1:9, std\_MSE,'filled'); xlabel('index'); ylabel('standart diviation'); title('standart diviation of MSE')

p = zeros(9,9);

for j=1:9

for i=1:9

p(j,i) = anova1([MSE\_2(:,i), MSE\_2(:,j)]);

end

end

%% 3.

eye\_photo = load('1.1.mat'); %load images

pics = eye\_photo.pics;

% removing last pic of each looking direction out:

pics3 = cell(1,4);

for i = 1:4

for j=1:16

pics3{1,i}{j} = pics{1,i}{j};

end

end

%3.2

%example of eye detection image

figure

for i=1:4

subplot(2,2,i);

I = rgb2gray(pics{1,i}{10}(150:400,250:950,:));

Eye\_Pos = EyePosition\_FUNC(I);

I = insertShape(I,'circle',[Eye\_Pos(1,:), 10]);

I = insertShape(I,'circle',[Eye\_Pos(2,:), 10]);

imshow(I)

title(['Direction ' num2str(i)])

end

%3.3

%2D histograms of each looking direction of left eye

EyeCalib\_L = EyeCalibration\_FUNC(pics,1);

figure; subplot(1,2,1);

for i=1:4

histogram2(EyeCalib\_L(:,2\*i-1),EyeCalib\_L(:,2\*i),20)

hold on;

end

title('Left eye histogram'); xlabel('x'); ylabel('y')

legend('direction 1', 'direction 2', 'direction 3', 'direction 4');

%2D histograms of each looking direction of right eye

EyeCalib\_R = EyeCalibration\_FUNC(pics,2);

subplot(1,2,2);

for i=1:4

histogram2(EyeCalib\_R(:,2\*i-1),EyeCalib\_R(:,2\*i),20)

hold on;

end

title('Right eye histogram'); xlabel('x'); ylabel('y')

legend('direction 1', 'direction 2', 'direction 3', 'direction 4');

%3.4

% horizontal dimension (x) histograms of left and right eyes

edge\_R = 100:2:250;

edge\_L = 480:2:640;

figure; subplot(1,2,1)

for i=1:4

histogram(EyeCalib\_L(:,2\*i-1),edge\_L)

hold on;

end

title('Left eye histogram'); xlabel('x'); ylabel('counts')

legend('direction 1', 'direction 2', 'direction 3', 'direction 4');

subplot(1,2,2);

for i=1:4

histogram(EyeCalib\_R(:,2\*i-1),edge\_R)

hold on;

end

legend('direction 1', 'direction 2', 'direction 3', 'direction 4');

title('Left eye histogram'); xlabel('x'); ylabel('counts')

% normal pdfs created from the data for each looking direction and each eye

x = 0:256;

figure

for i=1:4

pd\_R = fitdist(EyeCalib\_R(:,2\*i-1),'Normal');

y = pdf(pd\_R,x); subplot(2,2,1); plot(x,y); hold on

title('Original right eye pdf')

legend('Direction 1','Direction 2','Direction 3','Direction 4','Location','Northwest')

pd\_R\_2 = makedist('Normal','mu',pd\_R.mu,'sigma',sqrt(pd\_R.sigma));

y = pdf(pd\_R\_2,x); subplot(2,2,2); plot(x,y); hold on

title('Transformed right eye pdf')

legend('Direction 1','Direction 2','Direction 3','Direction 4','Location','Northwest')

pd\_L = fitdist(EyeCalib\_L(:,2\*i),'Normal');

y = pdf(pd\_L,x); subplot(2,2,3); plot(x,y); hold on

title('Original left eye pdf')

legend('Direction 1','Direction 2','Direction 3','Direction 4','Location','Northwest')

pd\_L\_2 = makedist('Normal','mu',pd\_L.mu,'sigma',sqrt(pd\_L.sigma));

y = pdf(pd\_L\_2,x); subplot(2,2,4); plot(x,y); hold on

title('Transformed left eye pdf')

legend('Direction 1','Direction 2','Direction 3','Direction 4','Location','Northwest')

end

%3.4.2

% calibration matrix of the images without the last image of each looking

% direction

EyeCalib\_R\_16 = EyeCalibration\_FUNC(pics3,2);

EyeCalib\_L\_16 = EyeCalibration\_FUNC(pics3,1);

% Looking direction test images of known looking direction, one for each

% direction

Eye\_Look = zeros(1,4);

for i=1:4

Frame = rgb2gray(pics{1,i}{17}(150:400,250:950,:));

Eye\_Look (i) = EyeLook\_FUNC\_2(Frame,EyeCalib\_R\_16);

end

%3.5

% extracting the images out of avi file 1.2.avi that contains video of

% looking to different directions:

% saving the video to seperate images files

outputFolder = uigetdir(path);

% Read in the video

viddir = uigetdir(path);

viddir = fullfile(viddir,'1.2.avi');

obj = VideoReader(viddir);

vid = read(obj);

% Number of frames

frames = 445;

for x = 1 : frames

%Create a filename

outputBaseFileName = sprintf('Frame %4.4d.png', x);

outputFullFileName = fullfile(outputFolder, outputBaseFileName);

imwrite(vid(:,:,:,x), outputFullFileName, 'png');

end

vid\_im = cell(1,frames); %cell containing the frames of video 1.2.avi

for i=1:frames

if i<10

name = ['Frame 000', num2str(i), '.png'];

elseif i>=10 && i<100

name = ['Frame 00', num2str(i) , '.png'];

else

name = ['Frame 0', num2str(i) , '.png'];

end

vid\_im{i} = imread(name);

end

% manual segmentation of looking directions in the video

manual\_segment = [ ones(1,13) 2\*ones(1,3) ones(1,19) 2\*ones(1,30) ...

3\*ones(1,31) 2\*ones(1,28) 3\*ones(1,27) 4\*ones(1,28) ...

3\*ones(1,30) 2\*ones(1,34) ones(1,30) 2\*ones(1,27) ...

ones(1,32) 2\*ones(1,29) 3\*ones(1,32) 2\*ones(1,2) ones(1,28) 2\*ones(1,22)];

%3.6

% using the looking diredtion detection algorithm on the video frames

confusion\_mat = zeros(4,4);

for i=1:445

Frame = rgb2gray(vid\_im{i}(150:360,220:920,:));

Eye\_Look = EyeLook\_FUNC(Frame,EyeCalib\_R,EyeCalib\_L);

confusion\_mat(Eye\_Look, manual\_segment(i)) = confusion\_mat(Eye\_Look, manual\_segment(i)) + 1;

end

figure; confusionchart(confusion\_mat') %results of the algorithm

%% 4.

%4.1

%4.1.1

% extracting the images out of avi file 2.1.avi that contains video of

% walking on a treadmill:

% saving the video to seperate images files

outputFolder = uigetdir(path);

% Read in the video

viddir = uigetdir(path);

viddir = fullfile(viddir,'2.1.avi');

obj = VideoReader(viddir);

vid = read(obj);

% Number of frames

frames = 858;

for x = 1 : frames

%Create a filename

outputBaseFileName = sprintf('Frame\_leg %4.4d.png', x);

outputFullFileName = fullfile(outputFolder, outputBaseFileName);

imwrite(vid(:,:,:,x), outputFullFileName, 'png');

end

vid\_im = cell(1,frames); %cell containing the frames of video 1.2.avi

for i=1:frames

if i<10

name = ['Frame\_leg 000', num2str(i), '.png'];

elseif i>=10 && i<100

name = ['Frame\_leg 00', num2str(i) , '.png'];

else

name = ['Frame\_leg 0', num2str(i) , '.png'];

end

vid\_im{i} = imread(name);

end

%

%finding the approximate radius of the dark circls:

I = vid\_im{3};

I = rgb2gray(I);

med\_pic = medfilt2(I); %median filter for noise reduction

bwim2 = im2bw(med\_pic,75/255); %binary photo to emplify edges

figure; imshow(bwim2)

imdistline; %result - 28.73 diameter

% finding the marker coordinates of each image:

centers = cell(1,frames); %intialization of cell for the centers in each photo. each cell will contain a 4X2 matrix of the coordaintes

radii = cell(1,frames); %intialization of the cell for the radius of the black marker circle. each cell will contain 4X1 vector of the 4 radiuses

for i=1:frames

I = vid\_im{i};

I = rgb2gray(I);

med\_pic = medfilt2(I); %median filter for noise reduction

bwim2 = im2bw(med\_pic,75/255); %binary photo to emplify edges

[centers{i},radii{i}] = imfindcircles(bwim2,[13 16],'ObjectPolarity','dark', ...

'Sensitivity',0.97);

end

%% 4.1.2

n = 1:frames;

%sorting the centers so each center has the same location in the array

centers\_sorted = cell(size(centers));

for i=1:frames

if ~isempty(centers{i})

[centers\_sorted{i}(:,2),index] = sort(centers{i}(:,2));

centers\_sorted{i}(:,1) = centers{i}(index,1);

end

end

%matrixes for x and y locations of centers

x\_centers = zeros(5,858);

y\_centers = zeros(5,858);

for i=1:frames

for j=1:size(centers\_sorted{i},1)

x\_centers(j,i) = centers\_sorted{i}(j,1);

y\_centers(j,i) = centers\_sorted{i}(j,2);

end

end

% location of each marker in space

figure;

for i=1:4

subplot(1,4,i); scatter(x\_centers(i,:),y\_centers(i,:),'filled')

title(['Marker' , num2str(i)]); xlabel('x'); ylabel('y');

xlim([400 850]); ylim([50 500])

end

%4.2

fs = frames/28; %pixel/sec

Ts = 1/fs;

% clearing mistakely identified centers

x\_centers(5,:) = [];

y\_centers(5,:) = [];

% deriviative of centers location

difx = (diff(x\_centers'))';

dify = (diff(y\_centers'))';

% speed of markers

speed\_pix = sqrt(difx.^2+dify.^2)/Ts; %pixel/sec

t = Ts:Ts:(28-Ts);

% example of marker 1 speed with outliers

figure;

scatter(t,abs(speed\_pix(1,:)))

xlabel('time [sec]'); ylabel('speed |pixel/sec|');

title('Speed of Marker 1')

% markers speed without outliers

figure;

for i=1:4

subplot(2,2,i)

scatter(t,abs(speed\_pix(i,:)))

xlabel('time [sec]'); ylabel('speed |pixel/sec|');

title(['Speed of Marker ', num2str(i)])

xlim([0 5]); ylim([ 0 3000])

end

% finding the radius in pixels of the marker

figure; imshow(bwim2)

imdistline; %result - 13.7 diameter

marker\_radius = 13.7/2; %pixel

%conversion of speed to kmh

speed\_m = speed\_pix\*10^-2/(marker\_radius\*2); %m/s

speed\_kmh = speed\_m\*3.6; %kmh

% plot of markers' speed in kmh

figure;

for i=1:4

subplot(2,2,i)

scatter(t,abs(speed\_kmh(i,:)))

xlabel('time [sec]'); ylabel('speed |kmh|');

title(['Speed of Marker ', num2str(i)])

xlim([0 5]); ylim([ 0 6])

end

% real max and min location of marker 4

figure;

scatter(x\_centers(4,:),y\_centers(4,:),'filled')

title('Marker 4 poistions'); xlabel('x'); ylabel('y');

hold on; xline(463,'-r',{'minimum','463'}); xline(830,'-r',{'Maximum','830'});

% 4.4

% angles of thigh as function of time:

theta\_thigh = zeros(1,frames);

for i=1:frames

theta\_thigh(i) = atan((x\_centers(2,i)-x\_centers(1,i))/(y\_centers(1,i)-y\_centers(2,i)));

end

figure; scatter(t,abs(theta\_thigh(1:(end-1)))\*180/pi)

xlabel('time [sec]'); ylabel('|\theta\_t\_h\_i\_g\_h|')

%4.5

% angles of knee as function of time:

theta\_knee = zeros(1,frames);

for i=1:frames

ma = (y\_centers(2,i) - y\_centers(1,i))/(x\_centers(2,i) - x\_centers(1,i));

mb = (y\_centers(4,i) - y\_centers(3,i))/(x\_centers(4,i) - x\_centers(3,i));

x\_knee = (-1\*x\_centers(3,i)\*mb + y\_centers(3,i) + x\_centers(1,i)\*ma - y\_centers(1,i))/(ma-mb);

y\_knee = (x\_knee-x\_centers(1,i))\*ma+y\_centers(1,i);

a = sqrt((x\_centers(1,i)-x\_knee)^2+(y\_centers(1,i)-y\_knee)^2);

b = sqrt((x\_centers(4,i)-x\_knee)^2+(y\_centers(4,i)-y\_knee)^2);

c = sqrt((x\_centers(1,i)-x\_centers(4,i))^2+(y\_centers(1,i)-y\_centers(4,i))^2);

theta\_knee(i) = acos((a^2+b^2-c^2)/(2\*a\*b));

end

figure; scatter(t,abs(theta\_knee(1:(end-1)))\*180/pi)

xlabel('time [sec]'); ylabel('|\theta\_k\_n\_e\_e|')

פונקציית זיהוי מיקום האישונים:

function [Eye\_Pos]= EyePosition\_FUNC(Frame)

%% This function locates pupils in a picture of a face

%

% inputs:

% Frame - frame to locate eyes in. the frame is expected to be a grayscale

% image.

% outputs:

% Eye\_pos - 2X2 matrix. row one contains location of left eye x and y

% coordinates respectivly, row two is the same for right eye.

%%

%laplace filter: (for edge detection)

laplace = ones(3,3);

laplace(2,2) = -8;

% eyes refreneces: (that went throught the same image proccesing as the frames will go through)

dataR = load('eyeR.mat'); % right eye photo of proccessed picture to be used to cross-corelate with all the pictures

eyeR = dataR.eyeR;

dataL = load('eyeL.mat'); % left eye

eyeL = dataL.eyeL;

med\_pic = medfilt2(Frame); %median filter for noise reduction

edge2 = imfilter(med\_pic, laplace); %laplace filter for edge detection

bwim2 = im2bw(edge2,0.04); %binary photo to emplify edges

closed\_bw = double(imclose(bwim2,ones(3,3))); %using close operation for further enchancing of edges

corel\_R = xcorr2(closed\_bw,eyeR); %correlation between right eye example from a proccessed photo to a proccessed photo

[y\_R,x\_R] = find(corel\_R == max(max(corel\_R))); %maximum correlation indexes - location of right bottom corner of eye exmaple photo

y\_R = y\_R(1); x\_R = x\_R(1); % if there is more then one maximum

half\_pic = closed\_bw;

half\_pic(:,1:round(size(closed\_bw,2)/2)) = 0; %drakening left side of the picture to avoid correlation between left eye reffernce to right eye

corel\_L = xcorr2(half\_pic,eyeL); %correlation between left eye example from a proccessed photo to a proccessed photo

[y\_L,x\_L] = find(corel\_L == max(max(corel\_L)));

y\_L = y\_L(1); x\_L = x\_L(1);

Eye\_Pos = [x\_L-30 y\_L-32; x\_R-30 y\_R-32]; % the -32 and -30 on x and y coordinates are because the maximum value of the corelation function is given at the bottom right of the refference picture and we correct it to the pupils which are at the center of the refference

End

פונקציית מטריצת הקליברציה:

function [EyeCalib] = EyeCalibration\_FUNC(CalibFrames,eye\_side)

%% This function creates a calibration matrix of eyes location when looking into one of 4 different directions

% inputs:

% CalibeFrames - frames of face looking into one of 4 different directions

% eye\_side - The calibration matrix is for either right or left eye.

% use eye\_side = 1 for left eye, eye\_side = 2 for right eye.

% outputs:

% EyeCalib - The calibration matrix

%%

M = length(CalibFrames); L = length(CalibFrames{1,1});

pics2 = cell(1,M\*L); %initialize cell for all pics

% turning the cell of cells of photos into one cell array

for i=1:length(CalibFrames)

for j=1:length(CalibFrames{1,1})

pics2{L\*(i-1)+j} = CalibFrames{1,i}{j};

end

end

only\_eyes = cell(1,M\*L); %intialize cell for eye area photos

only\_eyes\_gray = cell(1,M\*L); %intialize cell for gray scale eye area photos

%cut only the eyes part area and create gray sacle images of them:

for i=1:M\*L

only\_eyes{i} = pics2{i}(150:400,250:950,:);

only\_eyes\_gray{i} = rgb2gray(only\_eyes{i});

end

%laplace filter for edge detection

laplace = ones(3,3);

laplace(2,2) = -8;

%first we extract a right and left eye examples out of one of the photos we will

%procces:

exmp\_pic = only\_eyes\_gray{19}; %gray scale image

med\_pic = medfilt2(exmp\_pic); %median filter for noise reduction

edge2 = imfilter(med\_pic, laplace); %laplace filter for edge detection

bwim2 = im2bw(edge2,0.04); %binary photo to emplify edges

closed\_bw = double(imclose(bwim2,ones(3,3))); %using close operation for further enchancing of edges

eyeR = double(closed\_bw(130:185,150:220)); % right eye photo of proccessed picture to be used to cross-corelate with all the pictures

eyeL = double(closed\_bw(175:220,495:570)); % left eye

save('eyeR.mat','eyeR');

save('eyeL.mat','eyeL');

EyeCalib = NaN(L,8); %initialization of the calibration matrix

for i=1:4

for j=1:L

Eye\_Pos = EyePosition\_FUNC(only\_eyes\_gray{(i-1)\*L+j});

EyeCalib(j,2\*i-1:2\*i) = Eye\_Pos(eye\_side,:);

end

end

end

פונקציית זיהוי כיוון ההסתכלות:

function [Eye\_Look]= EyeLook\_FUNC(Frame,EyeCalib\_R,EyeCalib\_L)

%% This function determines the looking location of the eyes out of 4 possible locations using a calibration matrix

% inputs:

% Frame - the image we want to locate the looking direction in

% EyeCalib\_R - calibration matrix of right eye

% EyeCalib\_L - calibration matrix of left eye

% outputs:

% Eye\_Look - direction of looking. a number between 1-4

%%

Eye\_Pos = EyePosition\_FUNC(Frame); %location of pupils in the frame

p\_R = zeros(1,4);

p\_L = zeros(1,4);

% creating normal distributions from each data column of the calibration

% matrix and using the pdf to calculate the highest probbility of the

% looking direction of the pupils in the image 'Frame'.

for i=1:4

pd\_R = fitdist(EyeCalib\_R(:,2\*i-1),'Normal');

pd\_R = makedist('Normal','mu',pd\_R.mu,'sigma',sqrt(pd\_R.sigma));

pd\_L = fitdist(EyeCalib\_L(:,2\*i),'Normal');

pd\_L = makedist('Normal','mu',pd\_L.mu,'sigma',sqrt(pd\_L.sigma));

p\_R(i) = pdf(pd\_R,Eye\_Pos(2,1));

p\_L(i) = pdf(pd\_L,Eye\_Pos(1,1));

end

% picking the highest probability between those generated from each eye:

[max\_R,Eye\_Look\_R] = max(p\_R);

[max\_L,Eye\_Look\_L] = max(p\_L);

if max\_R >= max\_L

Eye\_Look = Eye\_Look\_R;

else

Eye\_Look = Eye\_Look\_L;

end

end