

**עבודת גמר**  
**קורס "זרקור למחקר"**

## **מציאת צורות בתמונה בשיטת Hough Transform**



**נדב שלו יוסף, עומר רון-כהן, יובל תומר**

МОГШ ЛЕРНОВІ АРИАЛ ШМІР ЧАЛК МАТОВНІТІСТІСІНІМ СЛ ВІТ ТЕСФР АФІ АРЗІ ІМДУІИ МАХСІВ  
АУНІВЕРСІТІТ РІЙСМЕН  
АОКТЮБР 2022

## **תוכן העניינים**

3 .....	מבוא
4 .....	זיהוי גבולות בשיטת Canny
5 .....	אלגוריתם Hough Transform
9 .....	אלגוריתם Generalized Hough Transform
11 .....	מימוש אלגוריתם Hough Transform
14 .....	Template Matching
21 .....	סיכום
22 .....	רשימת מקורות

## מבוא

עם התפתחות המהירה בטכנולוגיה והגידול המשמעותי בצריכה לנצר שינוי בתהליכי הייצור, הבקרה ועיבוד המידע בתעשייה. תהליכי ידניים מוחזרים פוגעים בתפקה, לנצל הערך באוטומציה של תהליכי פשוטים אלה במטרה ליעל אותם. אחד המרכיבים הבולטים במכונות אוטומטיות הם סטודרים, קרי חישנים שמספקים מערכות קליטים מהסביבה ומאפשרים להן לפעול בהתאם. קיימים סוגים רבים של חישנים, כמו חישני טמפרטורה, לחות, קרינט אינפרא-אדום וכן מצלמות. במרבית המקרים נחוצות יכולות קליטה ועיבוד מידע מדויקות על ידי החישנים. בפיתוח מכונות שעוסקות בעיבוד תמונת נעשה שימוש נרחב במערכות שיזרימות מידע בצורה תמונות או סרטוני וידאו, ומהידע מעובד בהתאם לצורכי המכונה.

בתהליכי ניתוח אוטומטי של תמונות דיגיטליות מתעורר לעיתים קרובות הצורך בזיהוי צורות כגון קוים ישרים, עיגולים או אליפסות. Hough Transform (התמרת האף) היא אחת השיטות החשובות והידועות בעולם הראייה הממוחשבת לאותור אלמנטים בתמונות ספרתיות. בשיטה זו נעשה שימוש נרחב בישומים רבים כגון זיהוי נתיבים ותרוריות, זיהוי ביומטרי, ייצור מודלי תלת-ממד של ערים, בדיקת צנורות וכבלים וזיהוי עצמים מתחת למים.

במסמך זה נסקור את התפתחות שיטת Hough Transform, החל ברעיון הראשוני וכלה בהרחבות ובפיתוחים מתקדמים. כמו כן, נציג אלגוריתמים שמהווים אבני דרך בהתפתחותה של השיטה לאורך השנים וכן אלגוריתמים נוספים שמחיבים את השיטה ומאפשרים עיבוד נוסף של מידע מתחום.

## זיהוי גבולות בשיטת Canny

זיהוי גבולות בתמונה (edge detection) הוא מרכיב חשוב בעולם עיבוד התמונה והראייה הממוחשבת, והוא תנאי הכרחי למשימות כמו תפיסת עומק וזיהוי אובייקטים בתמונות ובסרטוני וידאו. בין היתר, הוא שלב מקדים לאלגוריתמים רבים, כמו Hough Transform. גבול מוגדר כמעבר בין גוונים שונים ובין רמות בהירות שונות בתמונה. בכל הנוגע לזיהוי אובייקטים, גבולות מופיעים בין שולי האובייקט לרקע שמאחוריו. אלגוריתמים שונים לזיהוי גבולות, ביניהם אלגוריתם Sobel, אלגוריתם Laplacian ואלגוריתם Canny בוណון, מזהים גרדיאנט גדול בין פיקסלים סמוכים המעידים על שינויי משמעותיים בהירות ובגון.

מספר דוגמאות לשימושים מרכזים בזיהוי גבולות הם זיהוי טביעות אצבע תוך הפחחת הרעש בסריקה, כך שנוצר מיקוד גבוה בצורת טביעת האצבע; דימות רפואיים תוך יצירה מיקוד בחלקים הרלוונטיים בסריקות זיהוי anomalיות בהן; זיהוי אוטומטי של כל רכב ממטרה להקטין את מרכיבות התמונה שמערכות רכבים אוטונומיות נדרשות לעבד, תוך שימירה על המידע הרלוונטי.

אלגוריתם Sobel הוא אחד מאבני הדרך המרכזיות בתחום עיבוד התמונה בכלל ובזיהוי גבולות בפרט. האלגוריתם פותח בשנת 1986 על ידי ג'ון קני, פרופסור באוניברסיטת ברקלি. האלגוריתם מקבל כקלט תמונה שמורמת (במידת הצורך) לתמונה בגוני אפור, ושני פרמטרים נוספים עיקריים: סף עליון וסף תחתון. הפלט של האלגוריתם הוא תמונה גבולות בשחור-לבן, בעל ממדים זהים לתמונה המקורית, בה הגבולות מסומנים בצבע לבן ושאר התמונה במצב שחור.

האלגוריתם פועל בשלבים הבאים:

1. חישוב פונקציית הגרדיאנט, כולל מציאת הנגורות החלקיות. לכל נקודה בתמונה מתתקבל וקטור נגזרת.
2. מציאת גודל וקטור הנגזרת וכיונו בכל נקודה.
3. מיון הנקודות ביחס לקלט הסף: הנקודות שהן גודל וקטור הנגזרת קטן מהסף התחתון (שהווין כקלט) נפסלות, ואילו הנקודות שהן גודל הוקטור גדול מהסף העליון מסומנות כגבולות.
4. סיווג הנקודות לפי כיוון הוקטור: הנקודות של כל נקודה מהנקודות שנותרו מקרבת לאחד מארבעה כיוונים: אופקי, אנכי או אלכסוני (אחד משני הכוונים האלכסוניים).
5. הנקודות שעוזרות ליצור רצף של נקודות בעלות אותו כיוון נגזרת יחד עם נקודות שכבר סומנו כגבולות מסומנות גם הן כגבולות.
6. הנקודות שסומנו כגבולות נקבעות לבן.



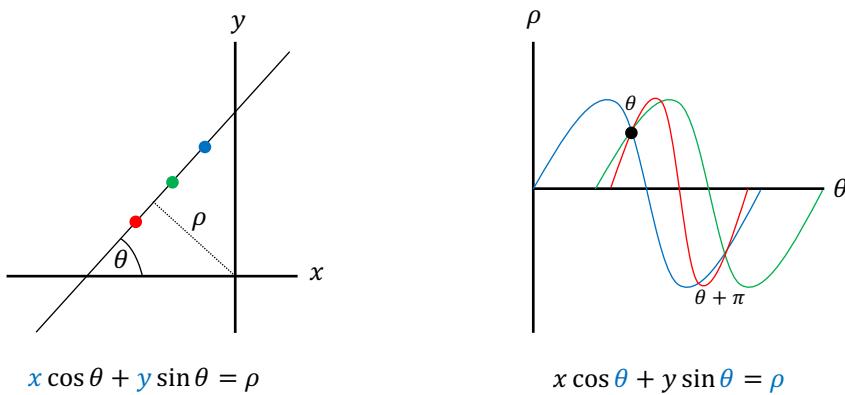
# Hough Transform

## תיאור כללי

Hough Transform הוא אלגוריתם שפותח במקור על ידי פול האג בשנת 1962, ומטרתו לוחות קווים בתמונות. מאז פיתוחו, האלגוריתם שונה וושופר כך שיאפשר זיהוי צורות נוספות כדוגמת עיגולים ומרובעים, ואף צורות שאינן מוגדרות. האלגוריתם פועל על גבולות בתמונה. כאמור, זיהוי גבולות מסייע לאתר מיקומים של אובייקטים בתמונה מקור. Hough Transform עשו שימוש בתמונות הגבולות, בה ערכי הפיקסלים הם "ビニアרים", כדי לאתר בתורו את האובייקטים שתואמים לצורה המבוקשת.

## הבסיס המתמטי לאלגוריתם

נביט בשני מרחבים: מרחב התמונה ומרחב הפרמטרים. מרחב התמונה הוא מישור המייצג את המNONות הגבולות שהתקבלו לאחר הפעלה של אלגוריתם זיהוי גבולות, כמו אלגוריתם Canny. כל קו ישר במרחב התמונה ניתן לייצוג כמשוואת מהצורה  $\rho = x \cos \theta + y \sin \theta$ , כאשר הקבועים  $\rho$  (הזווית השטוחה, החדה או הישרה שהקו יוצר עם החלק החיבוי של ציר ה- $x$ ) ו- $\theta$  (מרחב הקוו מהראשית), והציריים  $x$  ו- $y$  מייצגים את המשתנים  $x$  ו- $y$  בהתאם. במרחב הפרמטרים הקבועים הם  $x$  ו- $y$ , והציריים  $x$  ו- $y$  מייצגים את הפרמטרים  $\theta$  ו- $\rho$  בהתאם.



לכן, כל "נקודה" (פיקסל) במרחב התמונה מייצגת גל סינוס במרחב הפרמטרים. נשים לב שככל נקודה  $(x_i, y_i)$  במרחב התמונה מיוצגת על ידי המשוואת  $\rho = x_i \cos \theta + y_i \sin \theta$  במרחב הפרמטרים. זהה משווה את ישר בה הקבועים הם  $x_i$  ו- $y_i$ . היה שדרך כל שתי נקודות עבר ישר יחיד, כל הנקודות  $(x_i, y_i)$  המונחות על הישר  $\rho = x \cos \theta + y \sin \theta$  (עבור  $\theta$  ו- $\rho$  מסוימים) במרחב התמונה מיוצגות על ידי גלי סינוס העוברים דרך הנקודה  $(\rho, \theta)$  במרחב הפרמטרים ונחתכים בה. לכן, נקודות במרחב התמונה מונחות על אותו ישר אם ורק אם גלי הסינוס המייצגים אותן במרחב הפרמטרים נחתכים בשתי נקודות (שיעוריו ה- $\theta$  שלhn ה- $\theta$  ו- $\pi + \theta$ ).

חשוב להזכיר כי מידת המרחבים סופיים:  $\theta$  הוא מספר בין  $-\frac{\pi}{2}$  ל- $\frac{\pi}{2}$ ;  $\rho$  מייצג את מרחק הישר מהראשית במרחב התמונה, לכן הערך המוחלט שלו הוא מוגבל מלרע על ידי 0 ומולע על ידי אלכסון התמונה. לכן, מספר הנקודות במרחב התמונה הוא סופי ובהתאם מספר גלי הסינוס

במרחב הפרמטרים, המיצגים את הנקודות במרחב התמונה, סופי. מכאן שקיים מספר סופי של נקודות חיתוך בין גלי סינוס שונים במרחב הפרמטרים, ומספר גלי הסינוס העוברים דרך כל נקודה במרחב זה סופי גם כן.

### תיאור האלגוריתם

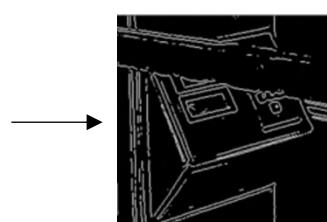
בשלב הראשון ניצור כאמור תמונה גבולות מקורו בעזרת אלגוריתם לזיהוי גבולות, כמו Canny edge detection. עבור כל פיקסל במרחב התמונה שזוויה חלק מגבול נבצע את ההמרה המתוארת לעיל לגלי סינוס במרחב הפרמטרים. לאחר מכן, ניצור מערך דו-ממדי צובר שמייצג את מרחב הפרמטרים. לצורך פשוטות ההסביר, המערך יהיה בגודל  $m \times \theta$ , כך שכל תא במערך מייצג "נקודה" (פיקסל) במרחב הפרמטרים. עבור כל גל סינוס במרחב הפרמטרים, נגדיל את הערך בתאי המערך בהם עבר הגל ב-1. קיבל מערך בו התאים עם הערכים הגבוהים ביותר (על פי ערכי סף מסוימים) מייצגים את הנקודות במרחב הפרמטרים בהן נחיתך המספר הרוב ביותר של גלי סינוס. בכיוון השני, נמיר את הנקודות האלה לקוים ישרים במרחב התמונה. אלה הקוים הבולטים ביותר בתמונה הגבולות, ולכן גם בתמונה המקור.



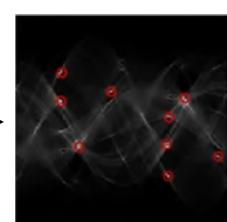
תמונה המקור



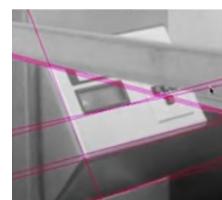
גרדיינט



תמונה גבולות



Hough Transform  
 $A(\theta, \rho)$



התוצאה

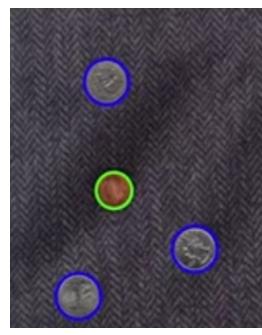
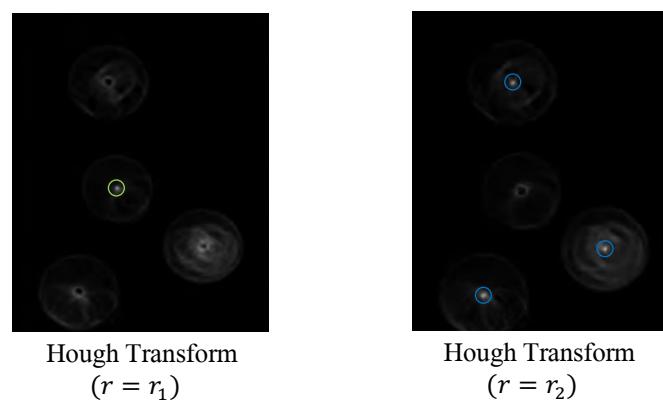
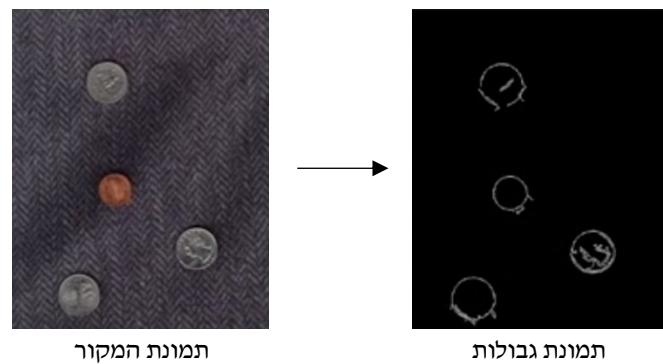
## בעיות ומוגבלות

- תיארנו אפוא תהליך מציאת קווים בתמונה בעזרת Hough Transform, עם זאת חשוב לשים לב למספר נקודות הכהרכיות לדיקוק האלגוריתם:
1. **גודל המערכת הצובר** – כאמור, לשם הפשטות הגדרנו את המערך הצובר להיות בגודל  $m \times \theta$ . במצב כזה, בו כל תא במערך מייצג פיקסל יחיד בתמונה, עלולה להיווצר רגישות גבוהה מאוד לרעש. זה יגרום במצב בו הערכים בתאים שונים יחסית ותאים רבים לא יעברו בערכם את ערך הסף לזיהוי, שכן קוים בתמונה המקורי לא יזוהו. מנגד, הגדרה של תאים גדולים מדי תוביל במצב בו קוים שונים בתמונה המקורי שמותרנים במספר נקודות חיתוך קרובות למרחב הפרמטרים ייספרו תחת תא זהה במערך, וכך יסומנו חוזרת��ו יחיד בתמונה המקורי.
  2. **קביעת ערכי הסף להגדלת קוים** – ראיינו שתא בעל ערך גבוה במערך הצובר מעיד על קו בתמונה המקורי. עבור כל קו תיווצר קבוצת תאים קרובים בעלי ערכים גבוהים. אם כן, יש להגדיר סף רגישות מסוים שרק לערכים הגבוהים ממנו נתיחס כמו, וכך נזהה את נקודות השיא בכל קבוצה כזו.
  3. **אי-דיקוקים בתמונה הגבולות** – אלגוריתם Hough Transform מתבסס על ייצרת תמונה גבולות בשלב המוקדים. ברוב המוחלט של המקרים, הגבולות בתמונה הגבולות לא יתקבלו כקוים רציפים, אלא יהיו בהם רעש. על מנת להתמודד עם הרעשים הללו, מומלץ עבור כל נקודה במרחב סינוס למרחב הפרמטרים לעדכן את ערכם של מספר תאים שונים, במקום תא אחד.

## הרחבת מעגלים ולצורות שניתן לייצג בעזרת משוואות

האלגוריתם שהציגנו מtabסס על תכונות הישר ועל משווהת הישר  $r = x \cos \theta + y \sin \theta$  המורכבת משני נעלמים וממשני פרמטרים. נוכל להפעיל את האלגוריתם באופן דומה על צורות אחרות נתן לייצג באותו אופן.

נביט במשווהת המעגל:  $r^2 = (y - b)^2 + (x - a)^2$ .  $a$  ו- $b$  הם פרמטרים כך שהנקודה  $(a, b)$  היא מרכז המעגל, ו- $r$  הוא רדיוס המעגל (קבוע). לכל הנקודות  $(y, x)$  המקיימות את המשווהה זו הנקודות אשר מונחות על המעגל. כפי שהציגנו קודם, במרחב הפרמטרים ציר ה- $x$  מייצג את הפרמטר  $a$  וציר ה- $y$  מייצג את הפרמטר  $b$ . כך, כל נקודה במרחב התמונה מייצגת מעגל במרחב הפרמטרים. נקודות במרחב התמונה המונחות על אותו מעגל שמרכזו בנקודה  $(a, b)$  והרדיוס שלו  $r$  (כאמור, קבוע) במרחב התמונה. נבנה מערך צובר באופן דומה, כך שעבור כל מעגל נגדיל את ערכי התאים בהם המעגל עובר ב-1. התאים עם הערכים הגבוהים ביותר (על פי ערכי סף מסוימים) מייצגים את הנקודות במרחב הפרמטרים בהן נחיתך המספר הרב ביותר של מעגלים. נמיר את הנקודות האלה למעגלים במרחב התמונה. אלה המעגלים הבולטים ביותר בתמונה הגבולות, וכך גם בתמונה המקורי.

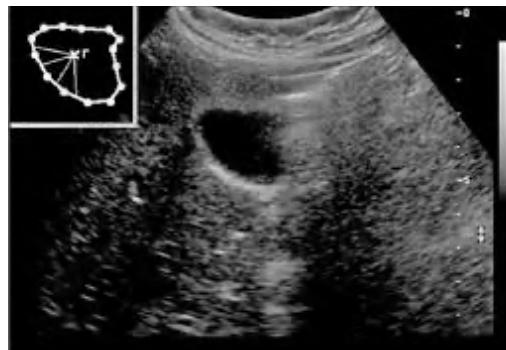


ניתן אפוא להפעיל את האלגוריתם באויה דרך למציאת צורות אותן ניתן לייצג באמצעות משווהה. נבוד במרחב דו-ממדי לפי המשתנים ובמרחב דו-ממדי לפי הפרמטרים. עם זאת, השיטה אינה מוגבלת למרחבים דו-ממדיים. נביט שוב במשוואת המעלג, כשהפעם הרדיוס  $r$  איינו קבוע. נקבל מרחב פרמטרים תלת-ממדי, בו ציר ה- $x$  מייצג את הפרמטר  $a$ , ציר ה- $y$  מייצג את הפרמטר  $b$  וציר ה- $z$  מייצג את הפרמטר  $r$ . בהסתכלה במרחב הפרמטרים קיבל משווהת חרוט, כלומר כל נקודה במרחב הצורה מייצגת חרוט במרחב הפרמטרים. השתמש במערך תלת-ממדי צובר בו התאים בעלי הערכיים הגבוהים מייצגים נקודות חיתוך של חרוטים במרחב הפרמטרים ובהתאם מעגלים בעלי רדיוס שונה במרחב התמונה.

## אלגוריתם Generalized Hough Transform

### תיאור כללי

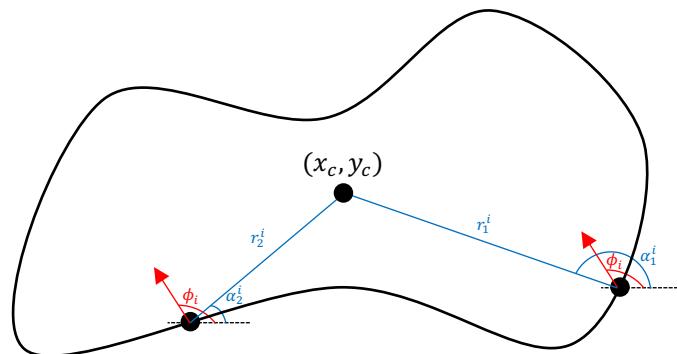
כאמור, אלגוריתם Hough Transform פותח תחילה כדי לזהות צורות מוגדרות אנליטית, כגון קוים ישרים, מעגלים, אליפסות ועוד. על כן, אלגוריתם זה מוגבל ואינו מאפשר מציאת צורות שאינן מוגדרות על ידי משואה או נוסחה כלשהי. לצורך זיהוי צורות כליליות כאלה פותח האלגוריתם Generalized Hough Transform, בעזרתו ניתן לזהות צורות שרירתיות, כולל צורות ללא נוסחה אנליטית פשוטה. אלגוריתם זה דורש מפרט מלא של הצורה המדויקת של אובייקט המטרה. האלגוריתם פותח על ידי דנה באלאד, פרופסור למדעי המחשב באוניברסיטת טקסס ב-1981.



### תיאור האלגוריתם

אלגוריתם זה, בשונה מאלגוריתם Hough Transform שראינו, דורש תמונות גבולות של האובייקט שנרצה לזהות, בה לכל נקודה גבול  $i$  נתון גם כיוון הגבול  $\phi_i$  (כאשר  $2\pi \leq \phi_i < 0$  לכל נקודה  $i$ ). בשלב מקדים לשימוש באלגוריתם נגדיר "נקודות ייחוס" של הצורה אותה אנחנו רוצים לחפש. נקודת הייחוס היא זו שנחפש במערך הצלבר של התמונה המקורי. נסמן אותה ב- $(x_c, y_c)$ . לכל נקודה גבול  $i$  בתמונה הגבולות של האובייקט נגדיר את הוקטור  $(r_k^i, \alpha_k^i) = \vec{r_k^i}$ , כאשר  $\vec{r_k^i}$  הוא מרחק הנקודה  $i$  מהנקודה  $(x_c, y_c)$  ו-  $\alpha_k^i$  היא הזווית שהקו המחבר בין שתי הנקודות יוצר עם ציר ה- $x$ .

נשים לב שעשויה להיות מספר נקודות עם כיוון גבול זהה ( $k$  נקודות כאלה).



לאחר מכון ניצור טבלת- $\phi$ , בה נשיק לכל כיוון  $i$  את הוקטורים  $\vec{r}_1^i, \dots, \vec{r}_k^i$  השווים לו. לדוגמה:

$\vec{r} = (r, \alpha)$	כיוון הגבול
$\vec{r}_1^1, \vec{r}_2^1, \vec{r}_3^1$	$\phi_i$
$\vec{r}_1^2, \vec{r}_2^2$	$\phi_i$
...	...
$\vec{r}_1^n, \vec{r}_2^n, \vec{r}_3^n, \vec{r}_4^n$	$\phi_n$

icut, המטרה היא לזיהות בתמונה המקור את מיקום נקודת הייחוס, אם האובייקט אכן נמצא בתמונה. נשתמש בתמונות הגבולות (הכוללת את כיווני הגבולות) של תמונה המקור. כל נקודת גבול  $i$  היא שלשה סדרה  $(x_i, y_i, \phi_i)$ , כאשר  $x_i$  ו- $y_i$  הם שיעורי הנקודה, ו- $\phi_i$  הוא כיוון הגבול בנקודת. ניעזר בטבלת- $\phi$  שבנוו בשלב הקודם ולכל נקודה  $i$ , לכל וקטור  $\vec{r}_1^i, \dots, \vec{r}_k^i$  נחשב את הערכים  $(x_i \pm r_k^i \cos(\alpha_k^i), y_i \pm r_k^i \sin(\alpha_k^i))$  ו- $(x_j = x_i \pm r_k^i \cos(\alpha_k^i), y_j = y_i \pm r_k^i \sin(\alpha_k^i))$ . נגדיל את ערכי התאים  $(x_j, y_j)$  ב-1 במערך הצובר. בדומה לאלגוריתם Hough Transform, נקודת המקסימום במערך היא נקודת הייחוס  $(x_c, y_c)$  שהגדנו מראש.



נציין שבאופן תאורי ניתן להרchip את האלגוריתם למקרים יותר כלליים, כך שיוכל לזיהות את האובייקט גם בגודלים שונים ובזוויתות שונות, למשל. נעשה זאת על ידי הוספה של פרמטרים נוספים, למערך הצובר, תוך הגדלת הממד שלו בהתאם, וכן לחישוב הערך  $(y_j, x_j)$  של כל וקטור. עם זאת, במקרים כאלה זמני החישוב עלולים להיות ארוכים מאוד וצריכת הזיכרון תהיה גדולה, מה שיבילל לאלגוריתם להיות לא יעיל בפועל.

## מימוש אלגוריתם Hough Transform

נכיג סקריפט Python שմMESS את אלגוריתם Hough Transform הבסיסי ליזיהוי קווים ישרים בתמונה. הקוד נמצא באתר GitHub בכתובת [github.com/yuvaltomer/Hough-Transform.git](https://github.com/yuvaltomer/Hough-Transform.git)

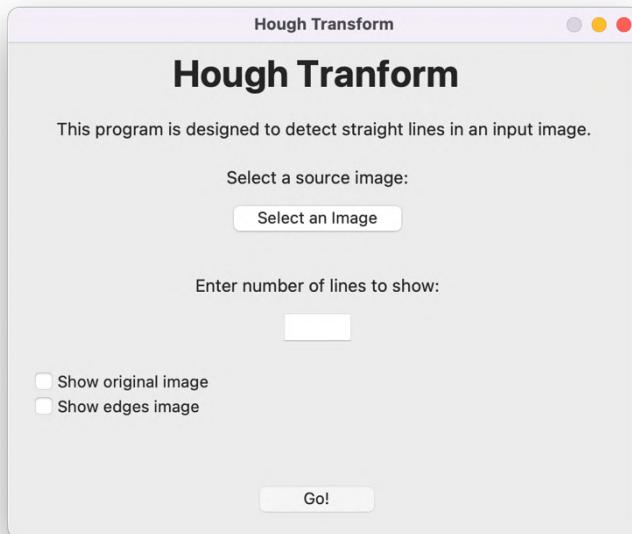
העקרונות שהנחו אותנו הם התמקדות בלוגיקה שמאחורי האלגוריתם, פשוטות וקריאות של הקוד, יצרת GUI בסיסי ותאיימות למגוון רחב של מוכנות. לכן, בחרנו למש את האלגוריתם בשפת Python. בIMPLEMENTATION נעזרנו בשתי ספריות עיקריות:

1. ספריית Python – OpenCV-Python – שמציעה פתרונות לביעות ראייה ממוחשבת.

השירותים העיקריים של הספרייה בהם השתמשנו נמצאים בפונקציה `(main)`, והם משמשים בעיקר ליצירת תמונות הגבולות. מעבר לכך, הספרייה מאפשרת את קריאת תמונת המקור וניהול הצגת תמונות הפלט.

2. ספרייה המשמשת לעובדה עם מערכיים.

החלון הראשי בתוכנית מאפשר לבחור את תמונה הקלט (מאחד הפורמטים הנתמכים), את מספר הקווים שיוצגו בתמונה הפלט, והאם להציג גם את התמונה המקורית ואת תמונה הגבולות.



פרמטרים נוספים, המוגדרים כמשתנים קבועים, נתונים לקונפיגורציה בקוד המקור.

בעת לחיצה על כפתור "Go!", נקראת הפונקציה `(main)`. בפונקציה נקרא נתיב תמונה המקור, ומוצגת התמונה (אם המשתמש בחר באפשרות זו). בשלב הבא מתבצע חישוב של תמונה הגבולות בעזרת הפונקציה `(Canny)`. לשם כך, יצרנו קודם לכון תמונה grayscale מהתמונה המקורית. ולאחר מכן השתמשנו במסנן גאוס (Gaussian Filter) על מנת להפחית רעש בתמונה grayscale. הפונקציות הללו נמצאות בספרייה `cv2`, ולא נפרט על אופן המימוש שלהם מכיוון שבמקרה זה הן

משמעות שלב מקדים ל-Hough Transform בלבד. עם זאת, ערכי הפרמטרים שנבחרו על ידיינו בקריאה לפונקציית אינס מותאמים לכל התמונות, ותמונה שוננות דורשות ערכים שונים על מנת להניב תוצאות מדויקות. בשלב הבא, מוצגת תמונה הגבולות (אם משתמש בחר באפשרות זו) ונקראות הפונקציות העיקריות שביצעו את ה-Hough Transform, אחת אחרי השנייה.

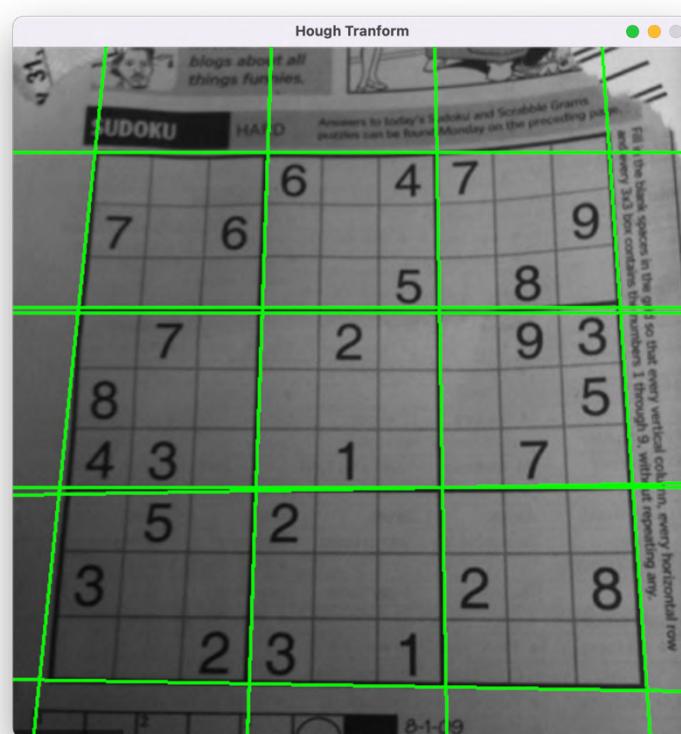
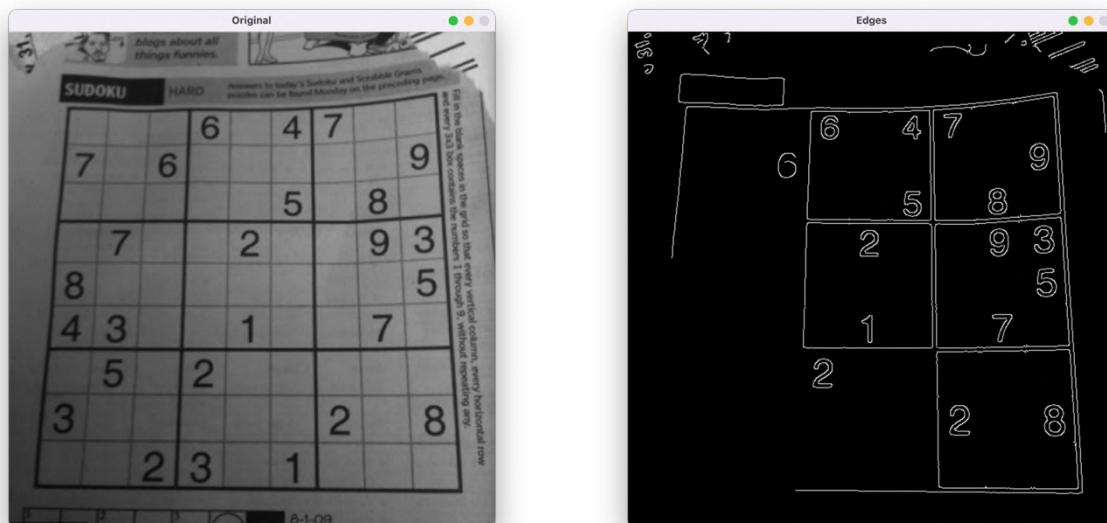
הfonkziaה הראשונה, `ConstructAccumulatorArray(image)`, מקבלת כקלט את תמונה הגבולות. הפונקציה יוצרת מערך של ערכי  $\theta$  אפשריים ומערך של ערכי  $r$  אפשריים. כאמור, הטווח האפשרי של ערכי  $\theta$  הוא  $-\frac{\pi}{2}$  עד  $\frac{\pi}{2}$  והטווח האפשרי של ערכי  $r$  הוא  $-r_{\text{diagonal}}$  עד  $r_{\text{diagonal}}$ , כאשר  $r_{\text{diagonal}}$  הוא אורך האלכסון. המרווח בין כל זוג ערכים בכל מערך נתון לשינוי, ועשו להשפיע על מידת הדיקוק והרגישות לרעש של האלגוריתם. לאחר מכן, הפונקציה יוצרת את המערך הצובר שישמש אותנו בהמשך ומאפסת אותו. המערך שנוצר רוחבו כמספר ערכי  $\theta$ - $r$  וגובהו כמספר ערכי  $\theta$ - $r$  שנקבעו במערכות הקודם. בשלב הבא, הפונקציה יוצרת רשימה של כל ערכי  $(x, y)$  של נקודות הגבול מתמונה הגבולות. עבור כל נקודה צו, לכל ערך  $\theta$  (מערך ערכי  $\theta$ ) מחושב ערך  $r$  תואם לפי הנוסחה  $r = \cos \theta + y$ . הערך שחושב מוקrb לערך שנמצא ברשימה ערכי  $\theta$ - $r$  (נסמננו ב- $\tilde{r}$ ), והערך במקומות  $(\theta, \tilde{r})$  במערך מוגדל ב-1. בסיום המעבר על כל נקודות הגבול, הפונקציה מחזירה את המערך הצובר וכן את המעריכים של ערכי  $\theta$  ו- $r$ . נציג כי התא במקומות  $(j, i)$  במערך מתאים לערך  $\theta$ - $r$  שנמצא במקומות  $i$ - $j$  במערך ערכי  $\theta$ - $r$  שנמצא במקומות  $i$ - $j$  במערך ערכי  $\theta$ .

הfonkziaה הבאה, `findMaxIndices(accArray, num0fLines)`, מקבלת כקלט את המערך הצובר שהתקבל מהfonkziaה `ConstructAccumulatorArray` ואת מספר הקווים להציגו שהוזן על ידי המשתמש. הפונקציה מפעילה פעולה לולאה `num0fLines` פעמים, ובכל איטרציה מוצאת את האינדקסים של התא עם הערך המקסימלי במערך, מוסיפה אותם לרשימה `indices` ולאחר מכן מאפסת את התאים שבביבת התא המקסימלי (כולל התא המקסימלי). מטרת האיפוס להימנע מרושע שעלווה להתקבל מכו אחד בתמונה המקורית שייצר קבוצה של תאים בעלי ערכיהם גבוהים באותה סביבה, שיתורגומו חוזרת למספר קווים שונים. באיטרציה הבאה יימצא התא הבא בעל הערך המקסימלי. גודל הסביבה אותה הפונקציה מאפסת נקבע קבוע `SQUARE_SIZE`. הפונקציה מחזירה את המערך `indices`, שמכיל קבוצת אינדקסים של הערכים הגבוהים ביותר שהוא במערך הצובר.

הfonkziaה الأخيرة, `drawLines(thetas, rhos, indices, image)`, מקבלת כקלט את רשימות ערכי  $\theta$ - $r$  וה- $r$  שהוחזרו מהfonkziaה `ConstructAccumulatorArray` ואת רשימת האינדקסים של התאים בעלי הערכים המקסימליים מהמערך הצובר שהוחזרו מהfonkziaה. לכל אינדקס מהרשימה `indices`, הפונקציה מותאמת את ערך  $\theta$ - $r$  ואת ערך  $r$ - $\theta$  המתאימים מהרשימות `thetas` ו-`rhos` בהתאם. לאחר מכן מבוצע הנדוס לאחרור כדי להציג

לשתי נקודות שנמצאות על הקו המקורי ולמתו כו' בינהן. לשם כך, בחרנו בשני ערכי קצה של  $x$  (אחד בכל קצה של התמונה), על מנת שהקו יעבור דרך התמונה כולה. ערך  $u$  תואם לכל ערך  $x$  מחושב שוב בעזרת המשוואה  $\rho = y \cos \theta + x \sin \theta$ .

בסיום, תמונת המקור מוצגת ועליה מסומנים הקווים שהאלגוריתם זיהה.



## Template Matching

נציג את מושג ה-Template Matching ונכיעו אלגוריתם מתאים בהתבסס על המאמר .Fast and accurate template matching with silhouette masking (Hayashi and Kadosaki, 2010)

### מבוא ל-Template matching

Template matching הוא תהליך שמטרתו לזהות אובייקטים מתוך תמונה מקור בתמונה קלט גדול יותר. קיימות שתי דרכים "נאיביות" נפוצות לביצוע התהליך: הראשונה היא שיטת brute force, קרי העברת חלון חיפוש על כל שטח תמונה הקלט ובדיקה האם החלק מהתמונה שנמצא בחלון החיפוש תואם את תמונה המקור; השניה, active search, עשוה שימוש בחיתוך היסטוגרמי הצבעים (histogram intersection) של תמונה המקור ושל האזור בתמונה הקלט שנמצא תחת חלון החיפוש. היסטוגרמת צבעים מדמה את הפיזור של פיקסלים בתמונה באמצעות הצגת גוף של מספר פיקסלים בכל אחת מרמות עצמת הצבע. משום כך, היסטוגרמה מאפשרת אפיון תמונה לפי הצבעים השונים שבה ופריסתם. חיתוך מקסימלי מצבע על רמת זיהוי גבוהה באזורי חלון החיפוש.

שתי השיטות המתוארות לעיל מסתמכו על ההנחה שchiposh האובייקט בתמונה הקלט מתבצע לפי תמונה המקור במלואה, כולל הרקע של האובייקט בתמונה המקור. ההסתמכות בשיטת brute force נובעת מיידית מצורת החיפוש, וההסתמכות בשיטת active search נובעת מהשפעת הרקע של האובייקט על ההיסטוריה של תמונה המקור. ההנחה זו יוצרת במקרים רבים טעויות בזיהוי אובייקטים.

התמונות מטה מתארות שניקרים של חיפוש אובייקט בשיטת active search. במקרה א מתבצע זיהוי מוצלח של האובייקט בתמונה הקלט, ואילו במקרה ב לא ניתן לזהות את האובייקט. (הצטברות נקודות אדומות באזור מסוים מעידה על מידת התאמה גבוהה).

מקרה ב



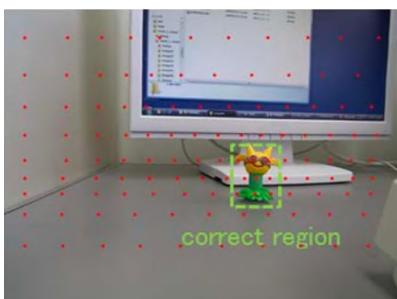
תמונה המקור

מקרה א

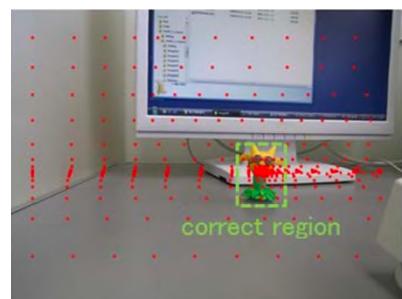


תמונה המקור

תמונה הקלט



תמונה הקלט

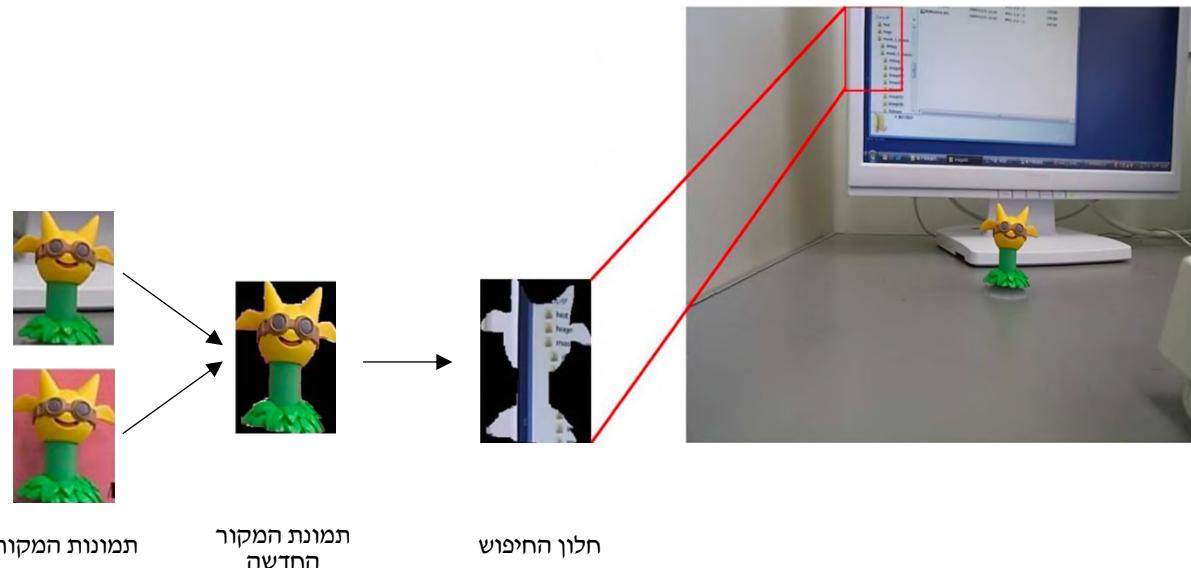


תמונה הקלט

הבדלים בין שני המקורים נובעים כאמור מההנחה שרקע האובייקט בתמונה הקלט תואם את רקע האובייקט בתמונה המקור: במקרה א התאמה כזו אכן קיימת, וailo במקרה ב, על אף שהאובייקט נוכח בתמונה, שינוי הרקע בתמונה המקור מונע זיהוי מוצלח שלו.

### אלגוריתם Silhouette Masking

אלגוריתם silhouette masking מציע פתרון יעיל לנטרול השפעות רקע האובייקט על הזיהוי. שלב מקדים לתהיליך חיפוש האובייקט בתמונה הקלט, מתבצע חיתוך של האובייקט מתוך המוקור, השחרת השארית ושימוש בה כמעין תבנית לזיהוי האובייקט בתמונה הקלט.



במאמר מוזכרות שתי שיטות לחיתוך התמונה (חיתוך האובייקט מהרקע) – LazySnapping ו-Kirie. עם זאת, לשם הפשטות, נניח שהמשתמש מסמן באופן ידני את היקף האובייקט וכל מה שמחוץ להיקף שסומן מושחר ומופרד מהאובייקט התבנית.

טהיליך הזיהוי המרכזי מתבצע לפי השלבים הבאים :

1. חישוב היסטוגרמת הצבעים של תמונה המקור החדשה  $H_0 = \{h_{01}, h_{02}, \dots, h_{0N}\}$ .
2. מיקום חלון החיפוש בפינה השמאלית העליונה של תמונה הקלט (נקודה (0,0) במישור התמונה).
3. מיסוך חלון החיפוש בעזרת תבנית החיפוש שהתקבלה בשלב המקדים.
4. חישוב היסטוגרמת הצבעים של החלק בתמונה הקלט שנמצא בחלון החיפוש הממוסך  $H_1 = \{h_{11}, h_{12}, \dots, h_{1N}\}$ .
5. חישוב חיתוך ההיסטוגרמות שהתקבלו בשלבים 1-4,  $S(H_0, H_1)$ , באופן הבא :

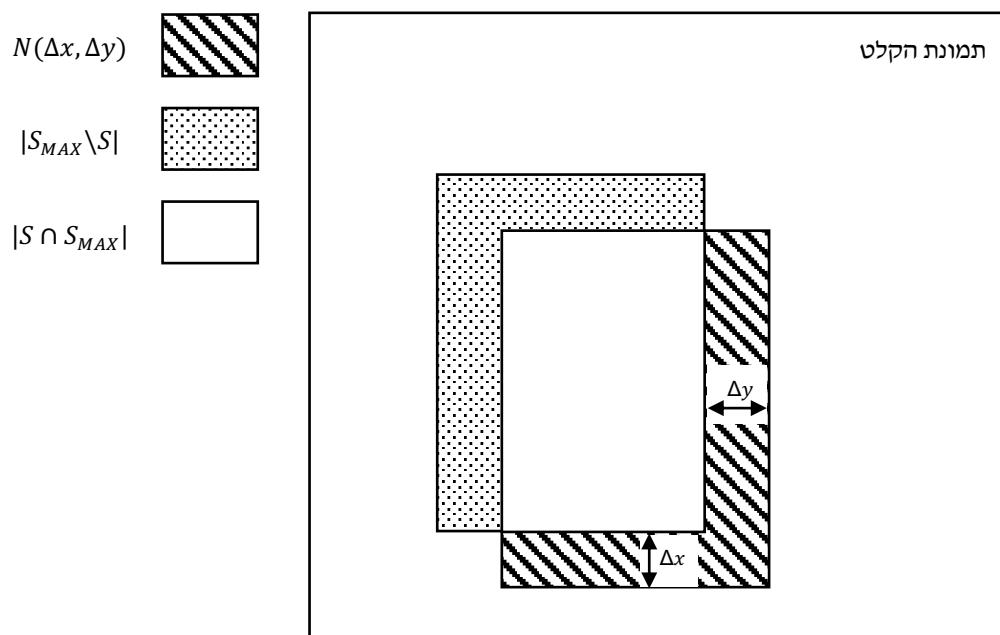
$$S(H_0, H_1) = \sum_{i=1}^N \min(h_{0i}, h_{1i})$$

6. הזזת חלון החיפוש כתלות ב-  $S(H_0, H_1)$  – טווח התזוזה ביחס הפוך לעוצמת החיתוך.
7. חוזרת על שלבים 3-6 עד סיום החיפוש.

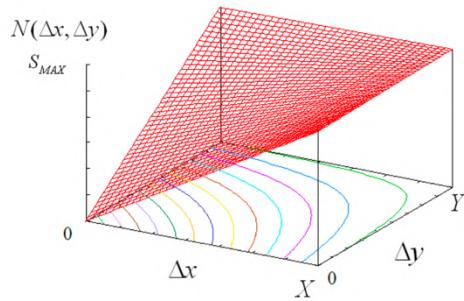
נביט באלגוריתם active search, בו חיפוש האובייקט מתבצע בעזרת תמונה המקור בשלמותה (לא החיתוך בשלב המקדים). נסמן ב- $S$  את חיתוך ההיסטוגרמות של תמונה המקור ושל חלון החיפוש הנוכחי וב- $S_{MAX}$  את חיתוך ההיסטוגרמות המקורי שמתקיים, כאמור לעיל, כאשר מתקבלת התאמה בין האובייקט לחלון החיפוש. נשים לב ש- $|S \setminus S_{MAX}|$  מייצג את מספר הפיקסלים בחלון החיפוש הנוכחי שצבעם אינם חלק מהאובייקט המקורי. נסמן ב- $\Delta x$  וב- $\Delta y$  את ההזזה של חלון החיפוש (בפיקסלים) בשלב 6 שליל בכל איטרציה על גבי הציריים  $x$  ו- $y$  בהתאם, וב- $N(\Delta x, \Delta y)$  את מספר הפיקסלים החדשם מתמונה הקלט שייתווסף לחלון החיפוש בכל חזזה שלו בשלב 6. כאשר מתקיים  $|S \setminus S_{MAX}| = N(\Delta x, \Delta y)$  ניתן כי חלון החיפוש הגיע לאזור בו נמצא האובייקט בתמונה הקלט.



תמונה המקור



נסמן ב- $X$  וב- $Y$  את רוחב וגובה תמונה המקור בפיקסלים בהתאם, ונשים לב שהפונקציה  $N(\Delta x, \Delta y)$  מונוטונית עולה לכל  $X < \Delta x < Y$  או  $\Delta y \geq Y$ . אם  $\Delta x \geq X$  או  $\Delta y \geq Y$  מתקיים  $|S \cap S_{MAX}| = N(\Delta x, \Delta y) = |S_{MAX}|$  – מספר הפיקסלים החדשם מתמונה הקלט שייתווסף לחלון החיפוש שווה לגודל של חלון החיפוש כולם, שווה לעוצמת החיתוך בין ההיסטוגרמות כאשר מתקבלת התאמה בין תמונה המקור לתמונה הקלט.



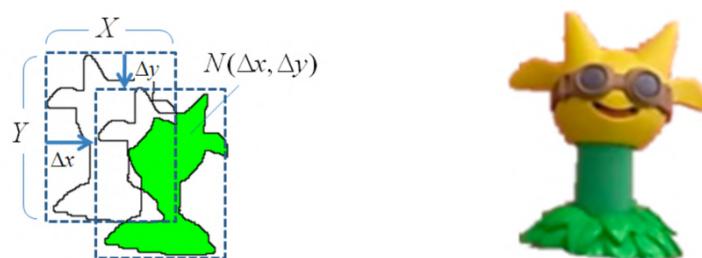
הפונקציה  $N(\Delta x, \Delta y)$

גודל חלון החיפוש ב- $N(\Delta x, \Delta y)$  הוא  $XY$ . נקבע

$$N(\Delta x, \Delta y) = XY - (X - \Delta x)(Y - \Delta y) \quad (1)$$

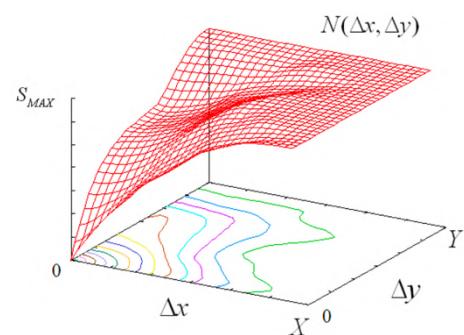
$$|S_{MAX}| = XY \quad (2)$$

בשימוש באלגוריתם Silhouette Masking התכונות של הפונקציה  $N(\Delta x, \Delta y)$  משתנות מכיוון שהפונקציה תליה בצורת האובייקט עצמו, לאחר מייסוק הרקע. באופן דומה, גם  $S_{MAX}$  מוגדר על ידי מספר הפיקסלים בשטח האובייקט עצמו. בנוסף, לא ניתן לחשב את טווח הזוזת חלון החיפוש  $(\Delta x, \Delta y)$  באופן מיידי, אלא יש צורך לחשב את תכונות הפונקציה  $N(\Delta x, \Delta y)$  טרם תחילת התהליך. עלות החישוב של תכונות הפונקציה גבוהות, لكن השיטה שאומצה במאמר היא קירוב של  $N(\Delta x, \Delta y)$  בעזרת המשוואה (1) שלעיל.



הזוזת חלון החיפוש על תמונה הקלט  
ומיציאת  $N(\Delta x, \Delta y)$

תמונה המקור

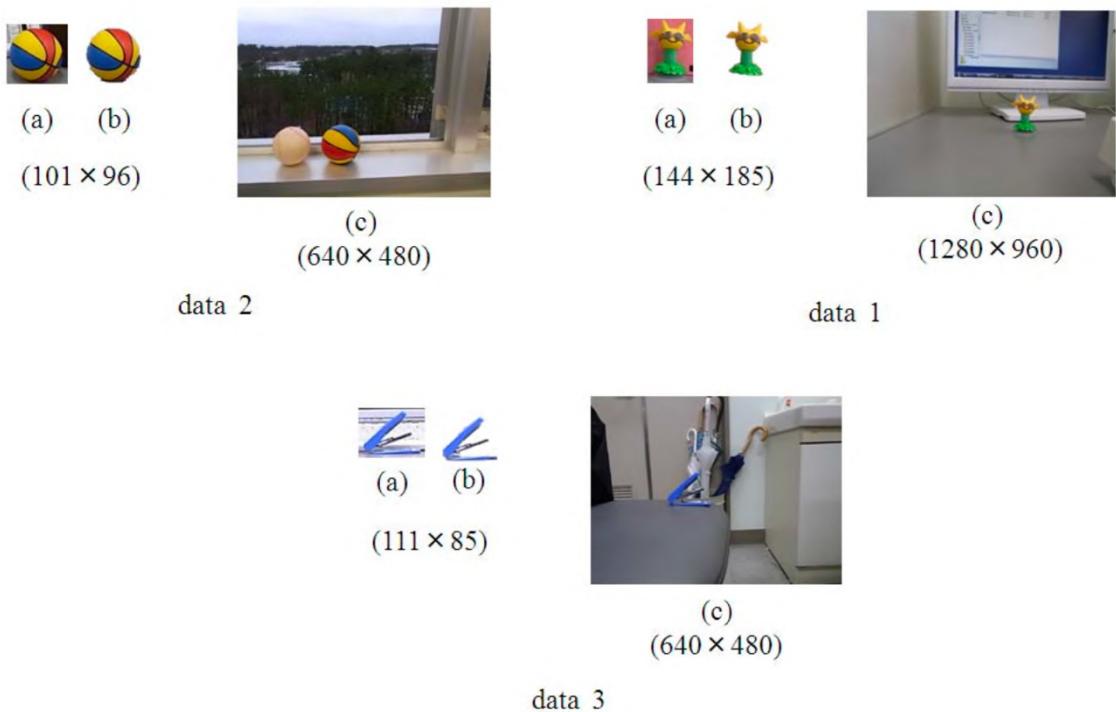


הפונקציה  $N(\Delta x, \Delta y)$

## הניסוי ותוצאותיו

המאמר בוחן את שיטת silhouette masking לעומת active search ו brute force. silhouettes masking מבחינה מהירות החישוב ובדיקה החיפוש. דיקח החיפוש מוגדר על ידי  $P = \frac{|A \cap B|}{|A|}$ , כאשר  $A$  הוא האזור שזוכה על ידי האלגוריתם,  $B$  הוא האזור הנכון והאופרטור  $| \cdot |$  מחזיר את מספר הpixels באזורה.

הבדיקה נעשתה על שלושה קלטים שונים כמפורט להלן :



עבור כל קלט, (a) היא תמונה המקור ששימשה עבור active search ועבור brute force (b) היא תמונה המקור לאחר חיתוך הרקע ששימשה עבור silhouette masking ו-(c) היא תמונה הקלט. תוצאות הניסוי מוארות בטבלה שללן :

data 1	accuracy	time (sec)
brute-force search	0.616	314.219
active search	0.649	0.047
proposed search	0.933	0.282

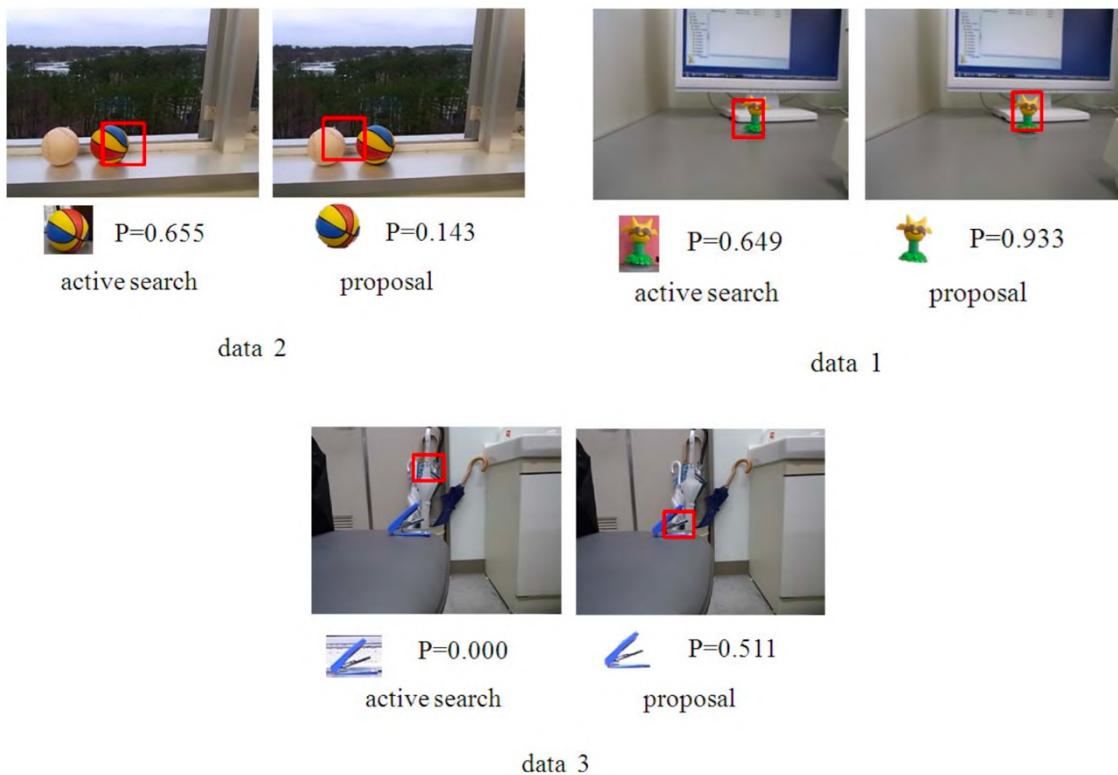
  

data 2	accuracy	time (sec)
brute-force search	0.776	34.641
active search	0.655	0.015
proposed search	0.143	0.016

data 3	accuracy	time (sec)
brute-force search	0.000	22.063
active search	0.000	0.015
proposed search	0.511	0.235

בנוסף, השוואת בין האזוריים שזווחו על ידי האלגוריתמים מוצגת להלן:



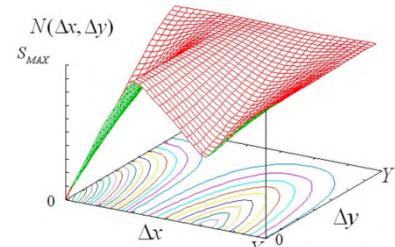
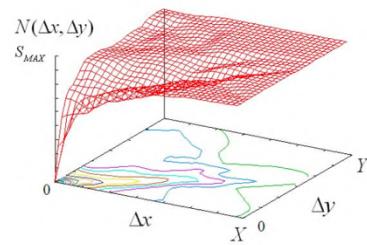
ניתן לראות שבקלטים 1 ו-3 נרשם דיק גובה יותר בחיפוש בעזרת אלגוריתם silhouette masking לעומת זאת, בקלט 2 רמת הדיק של האלגוריתם silhouette masking נמוכה מזו של האלגוריתמים הנאייביים. נשים לב שהצדור הצבעוני מופיע בתמונה המקורי ובתמונה הקלט בזווית שוננות, שכן קיימים הבדלים משמעותיים בין היסטוגרמת הצבעים של תמונה המקורי לשימוש silhouette masking לבין היסטוגרמת הצבעים של האזור הנקוד (בו מופיע הצדור) של תמונה הקלט. במקרים מסוימים זה האלגוריתם לא יכול ליזהות את תמונה המקורי באחוזי הצלחה גבוהה. כאן, באופן מקרי, הרקע של תמונה המקורי צפוף יותר מאשר תמונה הקלט, ולכן נרשם אחוזי דיק גבוהים יותר בשימוש באלגוריתמים אלה.

בקלט 3, השטח של הרקע בתמונה המקורי גדול ממשמעותית מזו של האובייקט עצמו, וכך יש לו השפעה גדולה על היסטוגרמת הצבעים של תמונה המקורי בשימוש באלגוריתמים הנאייביים, ולעומת זאת שימוש ב-silhouette masking מנטרל את ההשפעה זו. שימוש באלגוריתמים הנאייביים במקרה כאלה מושפע הרבה מהרקע של תמונה המקורי.

### מגבלות וצעדים להמשך

כאמור, המאמר מציע קירוב של הפונקציה ( $y, \Delta$ )  $N$  בעזרת משווה (1) שלעיל, אולם לקירוב כזה עלולות להיות השפעות שליליות על זיהוי נכון של אובייקטים בעלי צורה מורכבת או של מספר אובייקטים במקביל, כפי שנitinן לראות בדוגמאות שבהמשך. כאשר מתקבלים הבדלים גדולים בין הפונקציה האמיתית לפונקציה המקורבת, ניתן שטוחה שיחסוב יגרום לדילוג על האזור הנקוד בתמונה, וכ吐וצאה לכך לא יתבצע זיהוי נכון של האובייקט בתמונה הקלט. על כן, מציאת

דרך חישוב יעילה של הפונקציה  $N(\Delta x, \Delta y)$  טרם תחילת תהליך החיפוש תתרום לדיקוק האלגוריתם.



נוסף על כך, לפי התוצאות שהתקבלו מקלט 2 בניסוי, האלגוריתם עלול לא לפעול כמורה כאשר זווית הראה של האובייקט בתמונה המקורי שונה מזו שבתמונה המקורי. על מנת להגבר על המגבלה הזו, יש לספק מספר תמונות מקור שונות תוך פיתוח האלגוריתם בצורה יעילה כך שיוכל לשלב את תמונות המקור באופן שיאפשר חיפוש מכל זווית ראה.

## סיכום

שימוש תמונה וראייה ממוחשבת מהווים תחומי מחקר מובילים, רחבים וענפים במדעי המחשב. ישומים מעשיים בתחום אפשרים פוטנציאליים דרך שימושי בתעשייה, ותורמים רבים למגון רחב של תחומיים.

בעבודה זו סקרו בהרחבה אלגוריתם בתחום עיבוד התמונה, Hough Transform. נוכחנו בגלות שעקורות מטמטיים בסיסיים בתחום חישוב דיפרנציאלי, אלגברה ליניארית והנדסה אנליטית מאפשרים הפקה וניתוח של מידע מתוך מתרונות ומסרטוני וידאו. משום כך, קיים בסיס רחב למחקר ולפיתוח אלגוריתמים לעיבוד תמונה, כאשר כל אלגוריתם כזה מהווה בעצם בסיס לפיתוח מודלים מורכבים ולהמשך המחקר בתחום. ראיינו זאת, לדוגמה, בכך שאלגוריתם Canney ליזיהו גבולות בתמונה מהויה שלב מקדים לפעילותו של האלגוריתם Hough Transform.

שיטות שונות ליזיהו עצמים בתמונות מבוססות על עקרונות שונים, ואנו ראיינו שאלגוריתם template matching לsilhouette masking מושתת על ריעונות נפרדים מלאה שעליהם מושתת אלגוריתם Hough Transform. שילוב של אלגוריתמים כאלה, המבוססים על עקרונות מטמטיים שונים, מרחיב את היישומים המעשיים שניתן לפתח בעולמות עיבוד התמונה והראייה ממוחשבת.

באופן אישי, מעבר ללמידה והמחקר שביצענו על האלגוריתמים המוצגים בעבודה והמסקנות הנרחבות שהפקנו מהם, בעזרת עבודה מעשית זכינו לקבל טימה מתחומיים שונים ומגוונים החדשניים לנו : תכנות בשפת Python, שימוש בכלי בקרת תצורה, יצרת ממשק משתמש ותכנות frontend, תיעוד קוד ועבודה מבוצרת על פרויקט תוכנה.

## רשימת מקורות

- Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2), 111-122.
- Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 679-698.
- Chhaya, S. V., Khera, S., & Pradeepkumar, S. (2015). Basic geometric shape and primary colour detection using image processing on MATLAB. *International Journal of Research in Engineering and Technology*, 4(5), 505-509.
- Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1), 11-15.
- Hassanein, A. S., Mohammad, S., Sameer, M., & Ragab, M. E. (2015). A survey on Hough transform, theory, techniques and applications. *IJCSI*, 12(1), 139-156.
- Hayashi, T., & Kadosaki, Y. (2010). Fast and accurate template matching with silhouette masking. In T. Matsuo, N. Ishii & R. Lee (Eds.), *9<sup>th</sup> IEEE/ACIS International Conference on Computer and Information Science* (pp. 258-263). IEEE Computer Society.
- Hough, P. V. C. (1962). U.S Patent No. 3069654. U.S. Atomic Energy Commission.
- Zhou, P., Ye, W., & Wang, Q. (2011). An improved Canny algorithm for edge detection. *Journal of Computational Information Systems*, 7(5), 1516-1523.