# ShapeMaster


**מגישים:** נדב כהן – 208661819

לירון ליבוביץ – 209144773

רני אילון – 209307859

**שם המרצה :** פרופ' תמי ריקלין רביב

**תאריך:** 24.03.2025

**תקציר:**

shapemaster הוא משחק מוגבל בזמן המראה יכולות עיבוד תמונה בזמן אמת באמצעות מצלמת רשת. הפרויקט מיישם מערכת ראייה ממוחשבת העוקבת אחר תנועות יד, מזהה מיקומי אצבעות, ומאפשרת למשתמשים לצייר ולשחזר את הצורות באמצעות תנועת אצבעות. המערכת משלבת מספר טכניקות עיבוד תמונה הכוללות סגמנטציה של צבע עור באמצעות מרחבי הצבע HSV ו - YCrCb, זיהוי קצוות באמצעות שיטות מבוססות גרדיאנט, ניתוח קונטורים באמצעות אלגוריתמי Hull Convex, ומעקב אחר מיקום האצבע באמצעות פילטר קלמן .המשתמשים מנסים לשחזר את הצורות באמצעות תנועת האצבעות שלהם, כאשר הביצועים מוערכים על ידי השוואה כמותית בין הציור של המשתמש לבין צורת המטרה.
הפרוייקט מראה יישום מעשי של עקרונות עיבוד תמונה הנלמדו במהלך הקורס תוך מתן חווית משתמש אינטראקטיבית ומעניינת.

**הקדמה:**

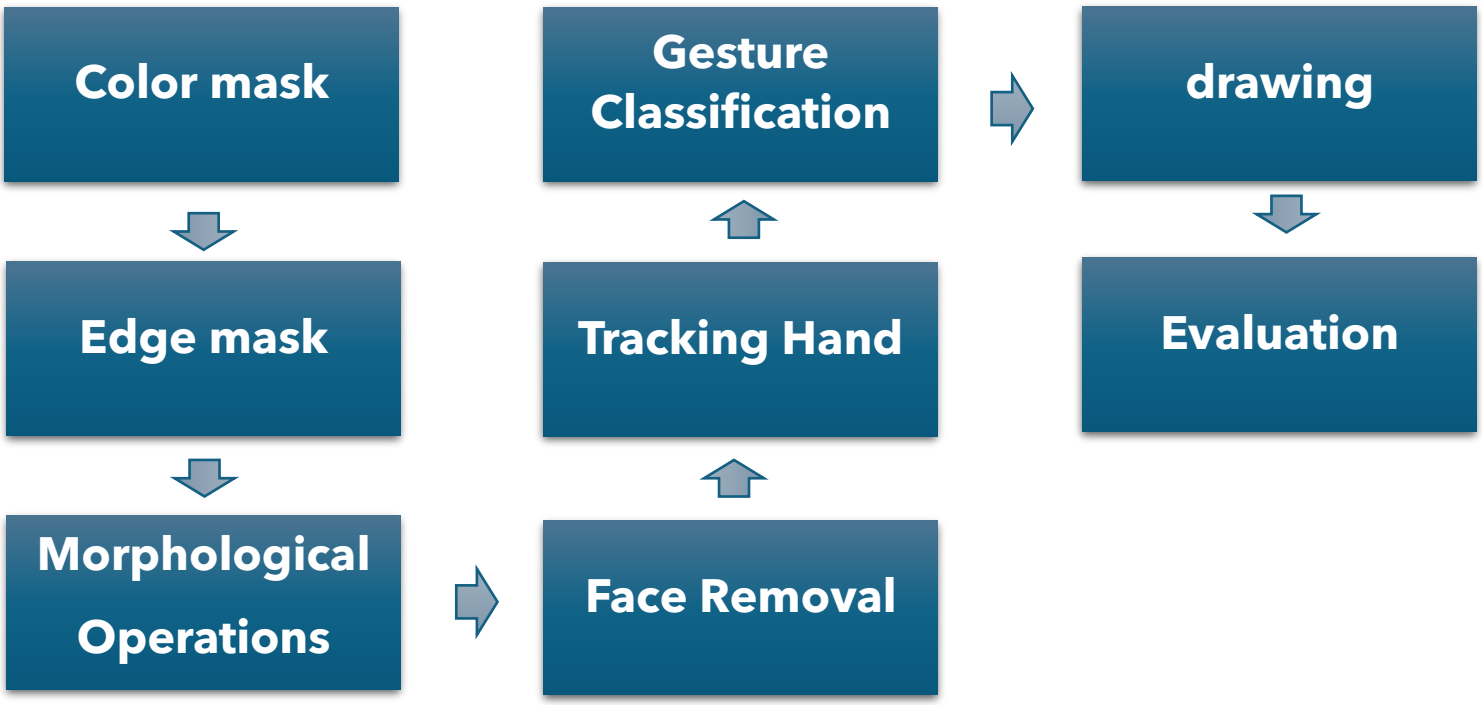
- **תיאור כללי:**

  משחק אינטראקטיבי רב שלבי המוגבל בזמן המשתמש במצלמת רשת ובאלמנטים קלאסיים של עיבוד תמונה.

- **מטרת הפרויקט ותוצאות מצופות:**
  - **מטרה :**להדגים יכולות עיבוד תמונה בזמן אמת דרך משחק.
  - **תוצאות מצופות :**הפרדת היד מהרקע, מעקב אחר היד, זיהוי נכון של תנועות היד, ציור על בסיס תנועות היד והצגת ניקוד המשתנה בהתאם לדיוק השחזור.

- **תיאור המערכת:**

מערכת המיישמת עיבוד תמונה בשלבים – קליטת תמונה, סגמנטציה של היד על בסיס צבע וזיהוי קונטורים, איתור הפנים ויצירת מסיכה בינארית. לאחר מכן מעקב על ידי Kalman Filter והמסיכה שיצרנו. ציור בעזרת זיהוי כמות ומיקום האצבעות בעזרת Convex Hull ולבסוף השוואת הציור שיצא עם הצורה המצופה בעזרת Quantitative Evaluation.

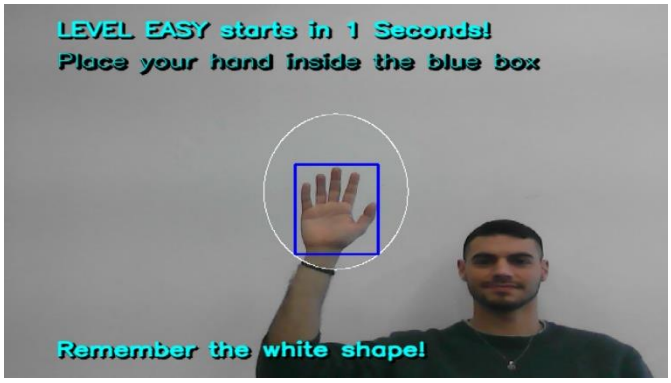


- **מטרת המשחק:**

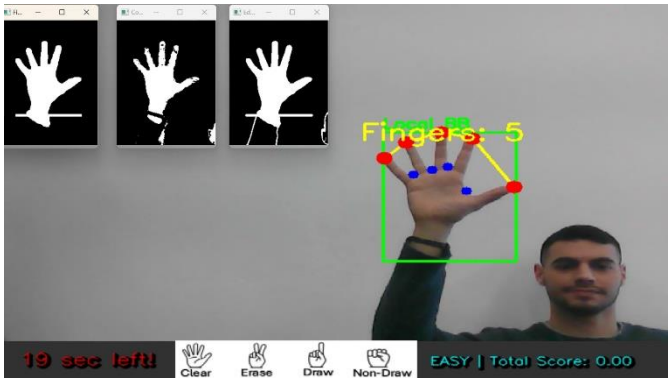
על המשתמש לשחזר צורה ומיקומה במרחב באמצעות תנועות אצבע, תוך ניסיון להגיע לניקוד הגבוה ביותר.

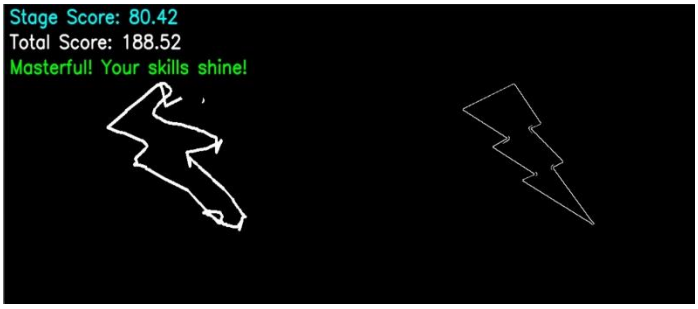
- **הוראות המשחק:**

1. המשתמש ממקם את ידו בתוך תיבה כחולה.
2. מתבקש לשנן את הצורה הלבנה ומיקומה במרחב.
3. באמצעות תנועות האצבע, המשתמש מנסה לשחזר את הציור.
4. המשתמש מקבל ניקוד על בסיס התאמה בין הציור שלו לבין הצורה המצופה.







- **אתגרים עיקריים:**
  - זיהוי מדויק של תנועות היד בזמן אמת תוך התמודדות עם רעש .
  - שמירה על עקביות באיכות הקלט במצבים משתנים (לדוג' תנועה פתאומית או שינוי בתנאי הסביבה, שינוי בתנאי תאורה).

- **מיקום המצלמות ויעודן:**

  מצלמה בודדת הממוקמת מול השחקן בזווית קבועה המקבילה ליד כך שתתפוס את אזור הפעולה המוגדר.

- **הנחות ואילוצי הפרויקט:**
  - שימוש ברקע אחיד על מנת להקל על תהליך הסגמנטציה וזיהוי הקונטורים.
  - כיול של היד בתוך אזור ספציפי על מנת לדגום את צבע היד של המשתמש.
  - על היד להיות במקביל למסך כך שהמצלמה תתפוס את כל היד ללא עיוותים.

## תיאור מפורט של האלגוריתמים לעיבוד תמונה והיישום

בסעיף זה אנו נציג את האלגוריתמים המרכזיים לעיבוד תמונה ומעקב בהם אנו משתמשים. נסקור את האלגוריתמים ליצירת מסכת עור, זיהוי קונטורים של היד וקצוות האצבעות, ייצוב תנועת היד באמצעות פילטר קלמן, זיהוי מחוות לציור ולמחיקה, ולבסוף דירוג הציור של המשתמש.

**יצירת מסכת צבע עור על בסיס 2 מרחבי צבעים (HSV ו-YCrCb):**

מסיכות הצבעים מבוססות על הרצאה 3 בנושא:

Filters cont., Scale-space, Colors.

- **דגימה של צבע היד:**

הצעד הראשון בתהליך הוא דגימה של צבע היד תוך שימוש במרחבי הצבע HSV ו-YCrCb. המטרה היא לזהות פיקסלים בעלי צבע עור ולסווג אותם כלבנים במסכה בינארית, בעוד שאזורים שאינם עור יסומנו בשחור.

- **כיול צבע העור:**

לשם כך, אנו מסתמכים על שתי פונקציות עיקריות:

  - **calibrate_skin_color(frame)** כיול ידני בתחילת כל שלב, בו מוצג ריבוע קטן על המסך שבו המשתמש מתבקש למקם את ידו. הפונקציה דוגמת את אזור בו נמצאת היד כדי לאמוד את צבע העור. במרחבי הצבע אנו משתמשים בחלק מערוצי הצבע בלבד(אלה שמצאנו כי הם מפרידים בצורה הטובה ביותר בין היד לשאר הפריים). הפונקציה מחשבת את הממוצע (μ) ואת סטיית התקן (σ) של הערוצים הרלוונטים. לאחר מכן, נבחרים סף תחתון וסף עליון סביב הגוון הממוצע לדוגמה:

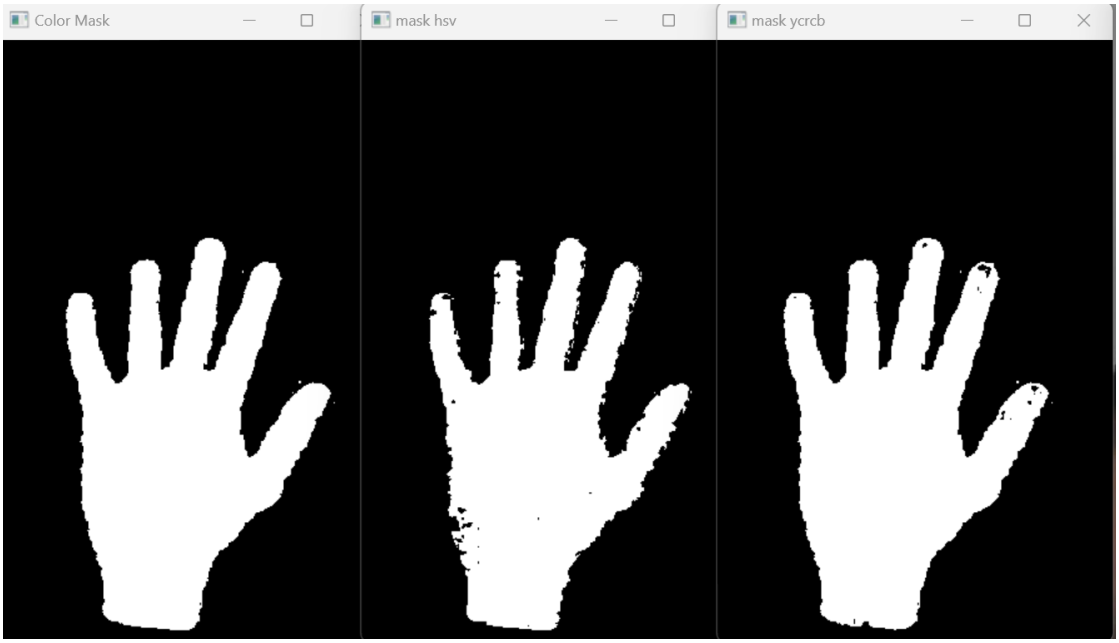
$$Lower\ Hue = \max(0, \mu - 2\sigma)\ , Upper\ Hue = \min(180, \mu - 2\sigma)$$

  - **auto_calibrate_skin_color(frame, cx, cy, box_size)** עדכון הצבע בזמן אמת, העדכון קורה כל 70 פריימים. הפונקציה דוגמת חתיכה בגודל 50×50 פיקסלים סביב מרכז היד ,אומדת מחדש את הממוצע ואת סטיות התקן, ומעדכנת את משתני הסף. במידה ונצפו שינויים דרסטיים הוא לא יתקן מכיוון שככל הנראה הוא לא דגם את היד. פונקציה זו באה להתגבר על שינוי בתאורה ובסביבה בעזרת עדכון של צבע היד.

- **יצירת מסיכת צבע משולבת:**

לאחר כיול צבע היד, הפונקציה הראשית detect_hand אשר רצה במהלך המשחק עצמו ממירה את האזור העניין לאותם שני מרחבי צבעים, הפונקציה משתמשת בcv2.inRange אשר מקבלת את התמונה ואת שני הספים שיצרנו וכל פיקסל שנמצא בטווח יצבע בלבן ונוצרת מסיכה בינארית לכל מרחב צבע. לאחר מכן, שתי המסכות הבינאריות המתקבלות משוקללות יחד. הפיקסלים המסומנים כלבנים במסכה הם אותם אשר מתייחסים ליד המשתמש.

לאחר השלב הזה כפי שניתן לראות בתמונה אנו רואים מקבלים 3 מסיכות:




הימנית זו מסיכת הצבע במרחב YCrCb, האמצעית זו מסיכה במרחב HSV והשמאלית זו השילוב של שתיהן, כפי שניתן לראות השילוב של שתי המסיכות מייצר לנו מסיכה מדויקת יותר.

**יצירת מסכת קצוות על בסיס גרדיאנטים ושינויים משמעותיים בפיקסלים:**

התהליך ליצירת מסיכת הקצוות נעשה במספר שלבים עיקריים שמטרתם למצוא את קווי המתאר של היד.

נושא זה נלמד בהרצאה - Edges & How to combine them.

- **בניית מסיכת על בסיס ושינויים משמעותיים בפיקסלים :**
  - נמיר את התמונה לגווני אפור כדי להתמקד במבנה (intensity) ללא תלות בצבעים.

- זיהוי קצוות עם Canny : הפונקציה cv2.Canny מממשת אלגוריתם לזיהוי קצוות הכולל מספר שלבים מרכזיים :

1. הפחתת רעש- בתחילה נעשה טשטוש באמצעות Gaussian blur.
המטרה היא להחליק את התמונה בעזרת קונבולוציה ולהפחית רעשים. הוא מבוסס על פונקציית התפלגות גאוסיאנית, כאשר כל פיקסל מוחלף בממוצע משוקלל של הפיקסלים הסמוכים לו. הפילטר בגודל מסוים מוגדר כך שהפיקסלים הקרובים למרכז קיבלו משקל גבוה יותר לעומת הפיקסלים המרוחקים יותר, מה שמביא לטשטוש טבעי וחלק יותר של התמונה.

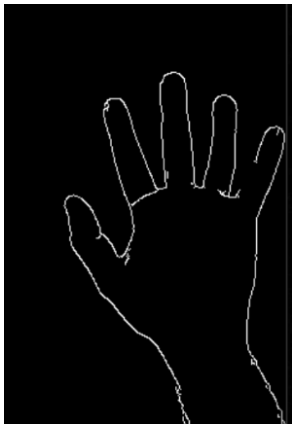
2. חישוב הגרדיאנט- בשלב זה מחושבים שינויים באינטנסיביות הפיקסלים עם אופרטור Sobel (יוסבר בהמשך) בכיוונים האופקי והאנכי. כך מתקבלות גם הערכים המייצגים את עוצמת השינוי (הגרדיאנט) וגם את הכיוון שלו.

3. דחיסת קצוות - בשלב זה מתבצעת דחיסה של קווי המתאר לעבר קו דק, כך שמוחקים פיקסלים שאינם מהווים את הקצה המקסימלי. זה עוזר להפיק קווים דקים וברורים בלבד.

4. סף כפול ומעקב - על פי שני ערכי סף - אחד גבוה ואחד נמוך - מסווגים הפיקסלים כ"קצוות חזקים" ו"קצוות חלשים". הפיקסלים עם ערכים גבוהים עוברי הסף מסומנים כקצוות אמיתיים, ואילו פיקסלים חלשים מסומנים רק אם הם סמוכים לקצה חזק.

הפונקציה canny מקבלת חלק מהפריים הנוכחי, מקבל שני ספים כפי שהסברנו את השימוש שלהם מקודם, ומחזירה לנו תמונה שחור לבן שהקצוות המופיעים בתמונה המקורית צבועים בלבן.

לאחר שלב זה נקבל :



כפי שניתן לראות לאחר שלב זה אנו מקבלים קצוות של היד אך מסיכה זו לא מושלמת.

- **בניית מסיכת על בסיס גרידאנטים :**

חישוב הגרדיאנט בכיוון האופקי ובכיוון האנכי בעזרת cv2.Sobel המחשבת את הנגזרת של התמונה, היא עובדת במספר שלבים :

cv2.Sobel היא פונקציה לחישוב נגזרות (גרדיאנט) של התמונה, והיא משמשת להדגשת שינויים באינטנסיביות הפיקסלים אשר מצביעים על קצוות או קווים בתמונה. להלן פירוט של שלבי הפעולה שלה :

1. בחירת כיוון הנגזרת - ניתן לחשב את הנגזרת בכיוון אופקי (x), אנכי (y) או בשני הכיוונים. הפרמטרים שניתנים לקבוע מאפשרים לבחור את הכיוון הרצוי לחישוב השינויים.
2. מסננים מוגדרים - הפונקציה משתמשת במסנן מסוים שמטרתו לחשב את השינוי סביב כל פיקסל. לדוגמה, במסנן אופקי, כל ערך מחושב על ידי חישוב ההפרש בין הפיקסלים מימין ומשמאל.
3. חישוב הגרדיאנט - עבור כל פיקסל, הפונקציה מחשבת את הערך של הגרדיאנט לפי הכיוון שנבחר. הערך שמתקבל מציין את עוצמת השינוי ,כמה השינוי חזק, ואת הכיוון בו מתקיים השינוי.
4. עיבוד ערכי הפלט - התוצאה המתקבלת היא תמונה שבה ערכי הפיקסלים מייצגים את גודל השינוי באות במיקום זה. ערכים גבוהים מצביעים על שינויים חדים (ולכן על קצוות), בעוד שערכים נמוכים מצביעים על אזורים אחידים.

הפונקציה sobel מקבלת חלק מהפריים הנוכחי , סוג מהשתנה של תמונת המוצא, סדר הנגזרת בכל אחד מהצירים ואת גודל הגרעין, ומחזירה לנו תמונה שחור לבן שהקצוות המופיעים בתמונה המקורית צבועים בלבן.

כעת חישבנו עוצמת הגרדיאנט בעזרת cv2.magnitude פונקציה זו מחשבת את גודל וקטור הגרדיאנט בכל פיקסל באמצעות הנוסחה :

$$\text{magnitude} = \sqrt{\nabla_x^2 + \nabla_y^2}$$

כך מתקבלת תמונה שבה ערך הפיקסל מציין את עוצמת השינוי .

לאחר שלב זה נקבל:

אנו רואים כי במסיכה זו היד מוצגת באופן ברור ומלא אך ישנם קצוות לא רצויים בתמונה.

אני משלבים בין מסיכת ה – canny לבין מסיכת הגרדיאנטים מכיוון שבשילוב המסכות אנחנו מנצלים את היתרונות של כל שיטה כדי לקבל מסיכת קצוות יציבה ומקיפה יותר:

- מסכת Canny - מתמקדת בזיהוי קצוות חזקים ומוגדרת היטב, אך עלולה לפספס קצוות עדינים או להיות רגישה להגדרות הסף.
- מסכת Sobel (גרדיאנט) - מחשבת את עוצמת השינוי בכל פיקסל ומספקת מידע על קצוות גם פחות בולטים, אך יכולה לכלול רעש או ערכים לא רצויים.

על ידי שילוב של שתי המסכות, אנחנו משיגים:

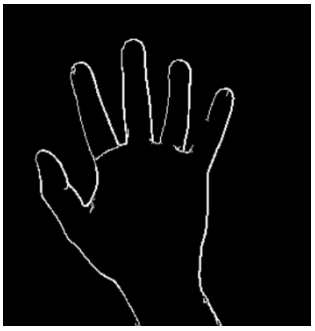
- ניצול היתרונות של שתי השיטות - קבלת מסיכה שמשלבת את הדיוק של Canny בזיהוי קצוות ברורים עם העושר במידע על קצוות עדינים שמתקבל מ-Sobel.
- הפחתת רעש - תהליך השילוב עוזר למתן את השפעות הרעש וליצור מסיכה אחידה יותר שתורמת לשלב העיבוד הבא (כמו מציאת קונטורים ומילוי האובייקטים).

כך מתקבלת תוצאה אמינה יותר לזיהוי הקצוות בתמונה, המנצלת את היתרונות של שתי השיטות. בשילוב המסכות אנחנו מנצלים את היתרונות של כל שיטה כדי לקבל מסיכת קצוות יציבה ומקיפה יותר.

- **שילוב שתי השיטות :**

שקלול שתי המסיכות באמצעות cv2.addWeighted, המשלבת את מסיכת ה - Canny ואת עוצמת הגרדיאנט לפי יחס של 0.6 למסיכת ה – Canny ו0.4 לעוצמת הגרדיאנט. היחס נבחר על ידי ניסוי וטעיה, יחס זה נתן לנו התוצאה המיטבית.

לאחר שלב זה נקבל:



מסיכה זו היא שילוב של שתי מסיכות הקצוות ואנו רואים כי אנו מקבלים שילוב של שתי המסיכות אך עדיין יש חורים קטנים שאנו נסגור בהמשך.

- **שיפור המבנה באמצעות פעולות מורפולוגיות:**

נושא זה לא נלמד במהלך ההרצאות.

סגירה מורפולוגית על המסיכה מוחלת פעולה של סגירה

(cv2.MORPH_CLOSE) cv2.morphologyEx בעזרת Kernel בגודל 3

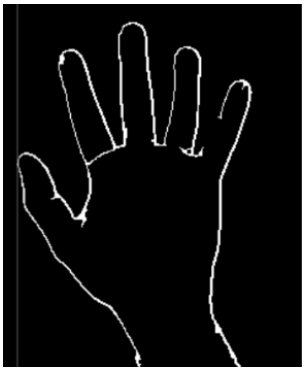
פעולה זו מורכבת משתי פעולות:

1. דילציה - בשלב הראשון מתבצעת התפשטות, שבה האובייקטים בתמונה "מתרחבים". פעולה זו מוסיפה פיקסלים לאזורי הגבולות של האובייקטים, מה שמוביל למילוי חורים קטנים בתוכם ולחיבור קצוות קרובים. כך, גם אם ישנם אזורים קטנים שבהם האובייקט לא רציף, הם מתמלאים והתמונה הופכת לאחידה יותר.
2. ארוזיה -לאחר ההתפשטות, מבצעים שחיקה שמטרתה להחזיר את האובייקטים לגודלם המקורי במידה רבה. בשלב זה, הגבולות המיותרים שנוצרו בשלב ההתפשטות "נחתכים" או "נמשכים" חזרה פנימה, אך החורים הקטנים שהתמלאו נותרו מלאים. כך מתקבלת מסיכה שבה האובייקטים מוצגים באופן אחיד וללא הפרעות קטנות.

ניתן להגדיר את כמות האיטרציות ששתי הפעולות האלה יתרחשו, ככל שנגדיל את מספר האיטרציות נקבל השפעה חזקה יותר של הפעולה.

פעולת ה - cv2.morphologyEx מקבלת תמונת המקור, סוג הפעולה – פה השתמשנו בcv2.MORPH_CLOSE , גרעין, כמות איטרציות. הפונקציה מוציאה את תמונה עם קווים מחוברים יותר.

לאחר שלב זה נקבל:



אנו רואים שלאחר הפעולות המורפולגיות רוב החורים נסגרו.

- **הפקת מסיכת קצוות מלאה:**

ראשית נמצא את כל הקונטורים - באמצעות cv2.findContours הפועלת על תמונה בינארית בצורה הבאה:

1. סריקת התמונה - הפונקציה סורקת את התמונה ומזהה את האזורים שבהם יש הבדל בין רקע לאובייקט, באזור שבו הערכים משתנים מ-0 ל-255.
2. איתור קונטורים - היא בונה קבוצה של נקודות (כל אחת מייצגת פיקסל) המקיפות את האזור האחיד, כלומר, מייצרות את הגבול של האובייקט.
3. הפרדה בין קונטור חיצוני לפנימי - בשימוש בפרמטר cv2.RETR_EXTERNAL, הפונקציה מחפשת רק את הקונטורים החיצוניים, כלומר הגבולות החיצוניים של האובייקטים בתמונה.
4. הפרמטר cv2.CHAIN_APPROX_SIMPLE מקטין את מספר הנקודות המאוחסן עבור כל קונטור – הוא שומר רק את נקודות הקצה החשובות וכך מפשט את הקונטור.

הפונקציה cv2.findContours מקבלת תמונה בינארית, האם אנו רוצים קווי מתאר פנימיים או חיצוניים, ושיטת קירוב קווי המתאר כפי שכתוב לעיל. הפונקציה מחזירה רשימה של כל קווי המתאר שזוהו כאשר כל קו מתאר הוא רשימה של נק' ומידע על הטופולוגיה של קווי המתאר.

לאחר שמצאנו את הקונטור נרצה למלא אותם , נייצר מסיכה חדשה בה נעתיק את הקונטורים ונמלא אותם בעזרת cv2.drawContours כך שכל אובייקט יהיה מלא.

הפונקציה cv2.drawContours מקבלת תמונה ריקה שעליה יצוירו קווי המתארת, מקבלת רשימה של קווי המתאר שאנו רוצים לצייר(בפורמט שיוצא מcv2.findContours ), האינדקס של קו המתאר לציור, צבע קו המתאר בפורמט, עובי קו המתאר- אנו ממלאים את כל הצורה. הפונקציה לא מחזירה פלט אלא משנה את התמונה שהיא מקבלת.

ולבסוף קיבלנו :

מסיכה זו לאחר מילוי הקונטורים ובכך אנו מקבלים את היד מאלה לקראת שילוב עם מסיכת הצבע.

**זיהוי הפנים ומחיקתם מהמסיכה:**

אנו מזהים את הפנים בעזרת מסווג מסוג ה־haarcascade. זיהוי הפנים מתבצע בשיטת הדומה אך לא זהה לעץ החלטה כפי שנלמד בהרצאה בנושא Machine Learning: Supervised Learning.

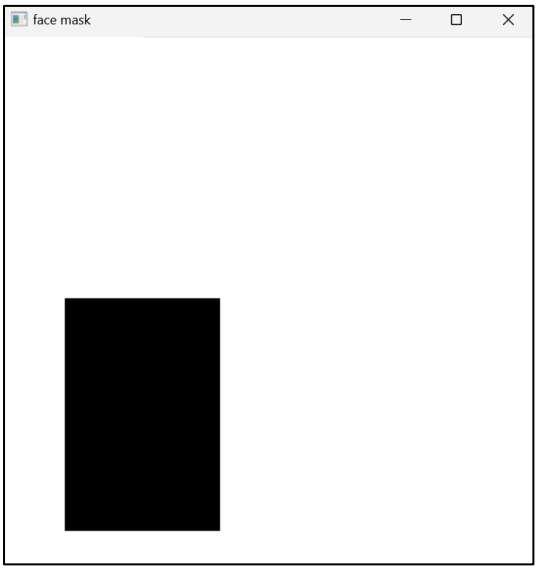
ה־ haarcascade_frontalface_default הוא מסווג (classifier) שמיועד לגילוי פנים בפרונט. להלן הסבר על אופן פעולתו :

1. מבנה הקאסקדה- ניתן לראות את מסווג ה־Haar Cascade כמעין "עץ החלטה" שבו בכל שלב מתקבלת החלטה עצמאית על בסיס תכונות מסוימות :
   - חלוקה בשלבים- בכל שלב, כמו בצומת בעץ, נבחנים חלונות תמונה על פי תכונות Haar (הסבר בהמשך). אם החלון עונה על הקריטריונים, הוא "עובר" לצומת הבאה. אם לא, הוא "נדחה" מיד ולא נבדק בהמשך.

- סינון הדרגתי - תהליך זה מאפשר לסנן במהירות את כל החלונות הלא רלוונטיים, כאשר רק החלונות שעברו את כל "צמתי ההחלטה" נחשבים כפוטנציאליים להכיל פנים. כך, הדמיון לעץ החלטה ניכר – הנתונים "מתפצלים" בכל שלב בהתאם להחלטות שמתקבלות בכל צומת.
- יעילות מהירה - כמו בעץ החלטה שבו כל החלטה מפלגת את הנתונים ומפחיתה את המרחב לחיפוש, המסווג מפצל את התמונה לשלבים ובכך משיג קבלת החלטות מהירה וחסכונית מבחינת משאבים.

2. תכונות Haar - האלגוריתם מתבסס על תכונות Haar –שמודדים הבדלים בין אזורים בהירים לאזורים כהים בתמונה. לדוגמה, פנים כוללות לעיתים אזור כהה (עיניים) באזור שמסביב לעיניים בהיר יותר. התכונות האלו מחושבות בצורה מאוד מהירה על ידי חישוב הפרשי סכומים באזורים מרובעים שונים בתמונה.
3. חלון סריקה - המסווג סורק את התמונה בגווני אפור בעזרת חלון קטן בגודל קבוע, ומחשב בכל מיקום את תכונות Haar. כדי לזהות פנים במידות שונות, סורקים גם בקנה מידה שונה – כלומר, מגדילים או מצמצמים את הגודל של החלון.
4. זיהוי והחזרת תוצאות- הפונקציה detectMultiScale סורקת את כל התמונה ומחזירה אזורים בהם זוהו פנים.

המסיכה שנקבל היא:




בתמונה המקורית הוא זיהה פנים בפינה השמאלית התחתונה ושם עליה ריבוע שחור שלאחר מכן נוריד אותה מהמסיכה הסופית.

**יצירת מסיכה סופית:**

- **שילוב עם מסכת הצבע עם מסיכת הקצוות:**

נמזג את מסיכת הקצוות עם המסכה הצבע באמצעות פעולת OR לכל פיקסל בשתי המסיכות בעזרת (cv2.bitwise_or), ובכך מתקבלת מסיכה משולבת שמכילה גם את המידע הצבעוני וגם את המידע של קווי המתאר. השימוש ב – OR מאפשר לדגום בצורה מדויקת יותר את גבולות היד ולהעביר מידע עשיר יותר לשלב העיבוד הבא, כמו זיהוי קצוות האצבעות או חיתוך האזור המתאים ביד.

- **שיפור המבנה באמצעות פעולות מורפולוגיות:**

שימוש בפתיחה מורפולוגית- על המסיכה מוחלת פעולה של פתיחה (cv2.MORPH_OPEN) cv2.morphologyEx בעזרת Kernel בגודל 3.
פעולת הפתיחה מורכבת משני שלבים עיקריים, והיא משלימה את פעולת הסגירה אך בסדר הפוך:

1. ארוזיה - בכל מיקום, הפיקסל החדש בתמונה הפלט הוא הערך הקטן ביותר בתוך חלון ה-kernel שמונח מעל התמונה המקורית. המשמעות היא שאם יש נקודת רקע (למשל, 0) בתוך החלון, הפיקסל המוחלש יהיה גם רקע. כך האובייקטים "מתכווצים", גבולותיהם מתכווצים והרעשים הקטנים נשטפים.
2. דילציה - בכל מיקום, הפיקסל החדש בתמונה הפלט הוא הערך הגדול ביותר בתוך חלון ה-kernel. כלומר, אם יש אפילו פיקסל אחד ברקע בתוך החלון, הוא "מרחיב" את הערך החיובי (למשל, 255) לשאר הפיקסלים. כך האובייקטים "מתרחבים", הגבולות מתמתחים והחורים הקטנים מתמלאים.

ולאחר מכן נשתמש בפעולת הסגירה כפי שהסברנו עליה מקודם.
שתי הפונקציות מקבלות קלט ומוציאות פלט כפי שציינו למעלה.
המסיכה הסופית נראית כך:

ניתן לראות כי הקו יד מזוהה בצורה ברורה ומלאה.

**מציאת כמות ומיקום האצבעות:**

- **זיהוי הקונטור העיקרי:**

לאחר הגעה למסיכה הסופית, נניח כי בתמונה נמצאת היד ועוד רעשים קטנים לכן נחפש את הקונטור הכי גדול ונוכל להסיק כי זאת היד. לשם כך, נעשה שימוש בפונקציה cv2.findContours לזיהוי קונטורים חיצוניים כפי שהסובר לעיל, ולאחר מכן נבחר הקונטור בעל השטח המקסימלי.

- **חישוב מעטפת קמורה:**

לאחר שמצאנו את הקונטור העיקרי, מתבצע חישוב מעטפת קמורה באמצעות הפונקציה cv2.convexHull, מתבצע באמצעות אלגוריתם גאומטרי העושה את הפעולות הבאות:

1. מיון נקודות - בתחילה, האלגוריתם ממיין את כל הנקודות המרכיבות את הקונטור לפי סדר מוגדר.
2. בניית הפוליגון הקמור - לאחר המיון, האלגוריתם עובר על הנקודות אחת אחת ובונה את הפוליגון בצורה הדרגתית:
   בכל שלב מוסיפים נקודה לרשימת הפוליגון.
   בודקים האם הכנסת הנקודה החדשה גורמת לעיוות – כלומר, האם נוצר זווית פנימית העולה על $180°$.
   אם כן, מסירים נקודות עד שהפוליגון נותר קמור.

בשיטה זו, כל נקודה שנבחרת נבדקת כדי לוודא שהפוליגון שנבנה עד כה שומר על תכונת הקמירות.

הפונקציה cv2.convexHull מקבלת מערך של נקודות או קו מתאר, ואנו ביקשנו על ידי משנה אופציונלי שתחזיר את האינדקסים של הנקודות במערך המקורי.

- **חילוץ קצוות האצבעות באמצעות ניתוח פגמי קמירות:**

לאחר מכן ננתח פגמים בקמירות בעזרת cv2.ConvexityDefects.

פונקציה זאת עובדת כך: בכל "פיסת" קו בין שני קצוות של הפוליגון, האלגוריתם בודק באיזה נקודה מהקונטור נמצא המרחק המקסימלי מהקו הישר שמחבר את שני הקצוות. נקודה זו, יחד עם נקודות ההתחלה והסיום של הקטע, ומהמרחק שלהן מהקו, מהווים את הדפקט – כלומר, החלק שבו הקונטור שוקע פנימה לעומת הפוליגון.

הפונקציה - cv2.ConvexityDefects מקבלת קו המתאר שעבורו רוצים למצוא את פגמי הקמירות ואת האינדקסים של נקודות המעטפת הקמורה (לא הנק' עצמם) והיא מחזירה מערך של פגמי קמירות כאשר כל פגם מיוצג על ידי 4 ערכים כפי שהסברנו למעלה.

לאחר מכן, נחשב את זוויות באמצעות הפונקציה angle_between_points(a, b, c) אשר מחשבת את הזווית בנקודה b בין הקווים (a–b) ו (c–b). -זווית נמוכה מצביעה בדרך כלל על בולטת אצבע, בעוד שזוויות גדולות או נקודות סמוכות לפרק כף היד אינן מייצגות קצוות אצבעות.

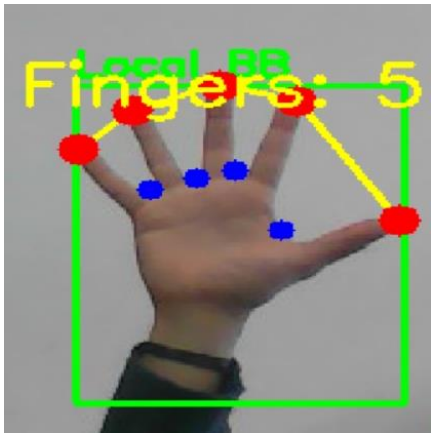
אנו נסנן נקודות בעזרת ההנחות הבאות:

- אנו אוספים את כל הפגמים שמציגים זוויות מתחת ל-100 מעלות מכיוון שהזווית בין האצבעות לא עולה על הזווית הזאת. בנוסף נתייחס רק לדפקטים בעלי מיקום מעל קו פרק כף היד בעזרת ההגדרה שכל הנקודות מתחת ל 65% מגובה היד לא ייחשבו.
- לבסוף, מתבצע מיזוג של נקודות קרובות באמצעות הפונקציה merge_close_points(points, distance_threshold=20) כדי למנוע ספירה כפולה של קצוות כמעט חופפים.

- **מקרה חריג:**

כאשר לא נותרו פגמים ברורים (למשל, כאשר היד כמעט סגורה או בזווית מאתגרת), נעבור לגישה פשוטה יותר, fallback_single_finger – אשר בוחרת את הנקודה הרחוקה ביותר בקונטור ממרכזו הגיאומטרי, וכך מבטיחה בדיקה נוספת במקרה של קצה אצבע אחד.

דוגמה לזיהוי אצבעות:




לאחר שלב זה נקבל את מספר האצבעות ואת הסימונים הבאים: קו המתאר המצויר בצהוב, הדפקטים בכחול והאצבעות באדום.

**עקיבה אחריי היד בעזרת פילטר קלמן**

נושא זה לא נלמד במהלך ההרצאות

- **הצורך בייצוב:**

בנוסף לשיטות שהשתמשנו עד עתה, קואורדינטות קצוות האצבעות יכולות להשתנות בין פריים לפריים בשל רעש תאורה, שינויי סף ושונות טבעית בזיהוי הקונטור.

- **יישום הפילטר:**

אנו משתמשים בפילטר קלמן, המאותחל ע״י init_kalman_filter() ומעודכן בכל איטרציה על ידי update_kalman(kf, measured_x, measured_y) אנו משתמשים בפילטר על מנת לעקוב אחר מרכז הכובד של היד.

- **הסבר מתמטי:**

פילטר קלמן ממדל את מצב המערכת כווקטור $x_k$, הכולל את קואורדינטות ה -x וה -y יחד עם מהירויות $(v_x, v_y)$ . אנו מניחים מודל עדכון ליניארי:

$$x_k = F \cdot x_{k-1} + w_{k-1}, \qquad z_k = H \cdot x_k + v_k$$

כאשר F היא מטריצת המעבר, H היא מטריצת המדידה, ו- $v_k$, $w_{k-1}$ הם רעשי התהליך והמדידה, בהתאמה. בקוד kf.transitionMatrix , ו- kf.measurementMatrix מוגדרים לייצג תנועה במהירות קבועה ומדידה ישירה של המיקום. בכל איטרציה, המערכת מנבאת מצב חדש בעזרת $F \cdot x_{k-1}$ ולאחר מכן מתקנת את הניבוי עם קבלת המיקום הנמדד $z_k$. תהליך דו-שלבי זה מבטיח עדכון חלק של המיקום ומונע תנודות פתאומיות.

**ציור, מחיקה וזיהוי מחוות**

- **פרשנות מספר האצבעות:**

המערכת מפרשת את מספר האצבעות כדי לקבוע את הפעולה על הקנבס הווירטואלי .

  - **ציור:**

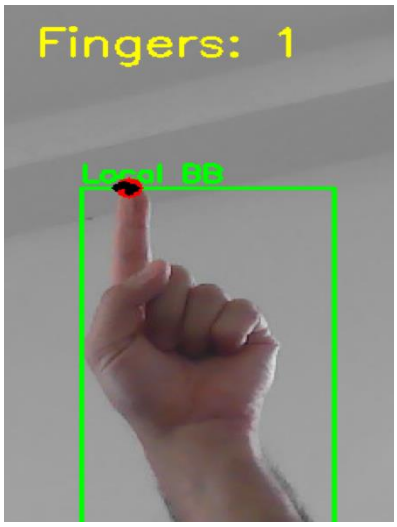
כאשר זוהה קצה אצבע בודד, המשתמש מצייר קו בצבע שחור המחבר את מיקום קצה האצבע הנוכחי למיקום הקודם על הקנבס.

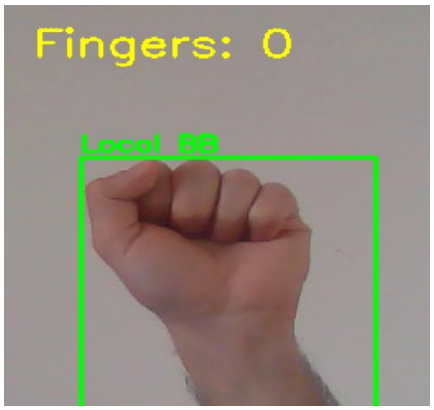
  - **מחיקה:**

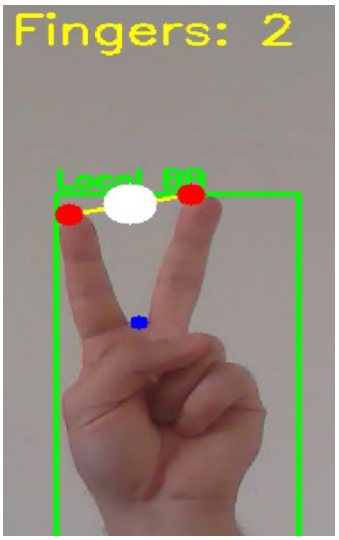
כאשר מופיעות שתי אצבעות, המערכת מפרשת את זה ככוונה למחיקה. הקוד מחשב נקודת אמצע בין שתי האצבעות ומצייר מעגל בצבע לבן כדי למחוק את הקווים.

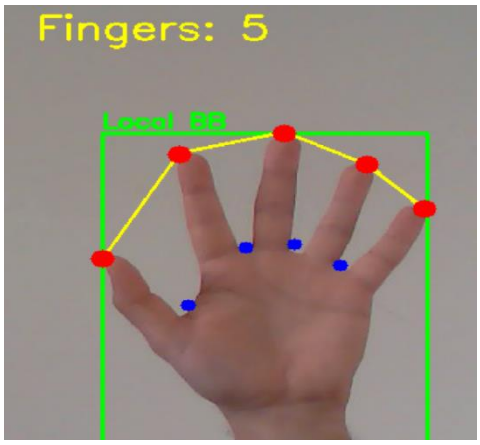
  - **איפוס הקנבס:**

זיהוי חמש אצבעות גורם לאיפוס הקנבס, כאשר הוא מצויר כולו בלבן.



**דירוג ציור המשתמש בשיטת Quantitative Evaluation:**

נושא זה נלמד בהרצאה בנושא Grouping and Segmentation.

בסיום כל שלב, הפונקציה

correlation_score(user_img, ref_img, dilation_size, factor, iter) משווה בין הציור של המשתמש לבין הציור המקורי. לשם כך, נהפוך את שתי התמונות להיות בעלות אותה ניגודיות – הציור בצבע לבן והרקע בצבע שחור.

נסמן את קבוצת הפיקסלים הלבנים בתמונת המשתמש כ - U.

נסמן את קבוצת הפיקסלים הלבנים בתמונת ה - reference כ -R.

- **פעולת Dilation** :

כדי לקחת בחשבון אי-התאמות מינוריות או קווים דקים, מתבצעת פעולת dilation על U ו- R.

פעולת dilation – כפי שהסברנו קודם פעולות המורפולגיות היא מעבה את הקווי הצורה.

לאחר מכן, נמדוד את החיתוך והאיחוד (union) של הקבוצות:

$$overlap = |U \cap R|, \qquad union = |U \cup R|$$

- **חישוב הדיוק והציון:**

הדיוק הגולמי מחושב כ:

$$accuracy = \frac{overlap}{union}$$

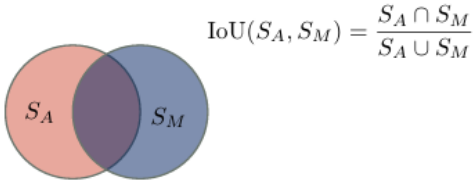
לבסוף, מכפילים את התוצאה ב-100 להמרה לאחוזים, והוספנו פקטור להקלה בניקוד:

$$Score = \min(accuracy \times 100 \times factor, 100)$$

נוסחה זו מבטיחה שהציון יהיה בין 0 ל-100, ומספקת מדד ברור לכמה הציור של המשתמש תואם לקווי הצורה בתמונת הreference.

בחרנו בשיטה זו כי רצינו לתת ניקוד גם על דיוק המיקום של הצורה במסך ולא רק על הקרבה לצורה עצמה לכן לא רצינו לעשות טרנספורמציה והתאמות של הצורה שצייר המשתמש לצורת הreference .

Quantitative Evaluation




שקף זה מההרצאה המתאר את שיטת הניקוד.

**ניתוח אלגוריתם לזיהוי מספר האצבעות בכף היד**

בחלק זה ניתחנו את ביצועי האלגוריתם לזיהוי מספר האצבעות בכף היד בהתבסס על עיבוד תמונה. בפרט, התמקדנו בהשפעת זווית כף היד על דיוק האלגוריתם.
הבדיקה כללה השוואה בין תוצאות זיהוי עבור יד במצב רגיל לבין יד המוטה בזווית של 45 מעלות לכיוונים שונים.

נבדקו שני מצבי יד:

1. **מצב רגיל** - כף היד ממוקמת בזווית ישרה כלפי המצלמה.
2. **מצב מוטה** - כף היד מוטה בזווית של 45 מעלות ביחס למצלמה.

בכל אחד מהמצבים, האלגוריתם נבחן במספר דגימות הכוללות ארבע קטגוריות שונות של מספרי אצבעות (0, 1, 2, 5). ביצועי המערכת הוערכו באמצעות confusion matrix , Hit Rate ומדד F1 Score.

השתמשתנו במדדים הבאים להעריך את הביצועים:

1. **Hit Rate** שיעור הדגימות שסווגו נכון מכלל הדגימות. מדד זה נותן אינדיקציה כללית לרמת הדיוק של האלגוריתם.
2. **F1 Score** מדד המאחד דיוק - Precision - מייצג את אחוז הדגימות שסווגו נכון מתוך כלל הדגימות שסווגו לקטגוריה מסוימת. כלומר, מתוך כל המקרים שבהם האלגוריתם זיהה מספר אצבעות מסוים ושליפה -Recall- מייצג את אחוז הדגימות שסווגו נכון מתוך כלל הדגימות שבאמת שייכות לקטגוריה מסוימת. כלומר, מתוך כל המקרים שבהם הקטגוריה הופיעה בפועל ומחושב באמצעות הנוסחה:

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

כאשר:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

מדד זה חשוב במיוחד כאשר יש חוסר איזון בין הכיתות, והוא מספק תובנה רחבה יותר לגבי ביצועי האלגוריתם.

בניתוח הנתונים נצפו ההבדלים בין שני מצבי היד :

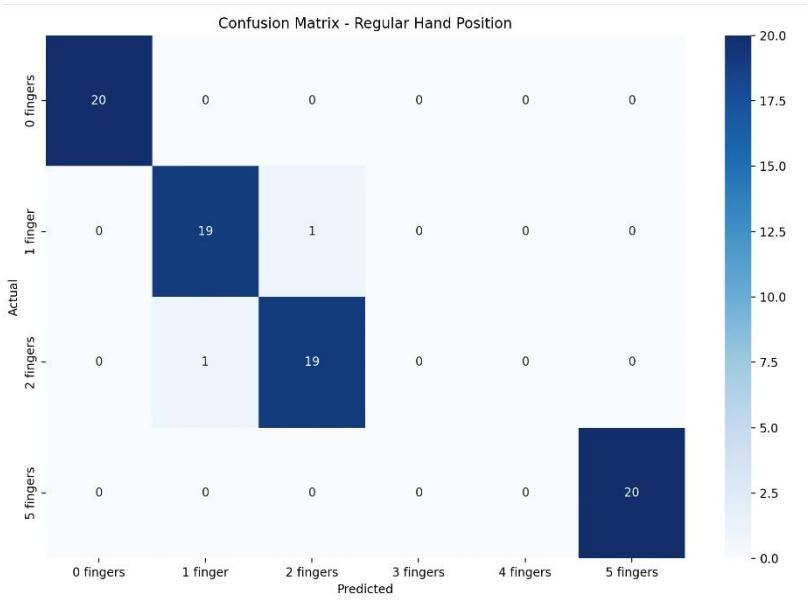
- במצב היד הרגיל, האלגוריתם השיג דיוק של 97.%
- במצב היד המוטה, נרשמו טעויות בזיהוי, בעיקר עבור סיווגים מסויימים כגון 5 אצבעות שהתבלבלו עם 3 או 4 אצבעות. הדבר מעיד על רגישות האלגוריתם לזווית כף היד.

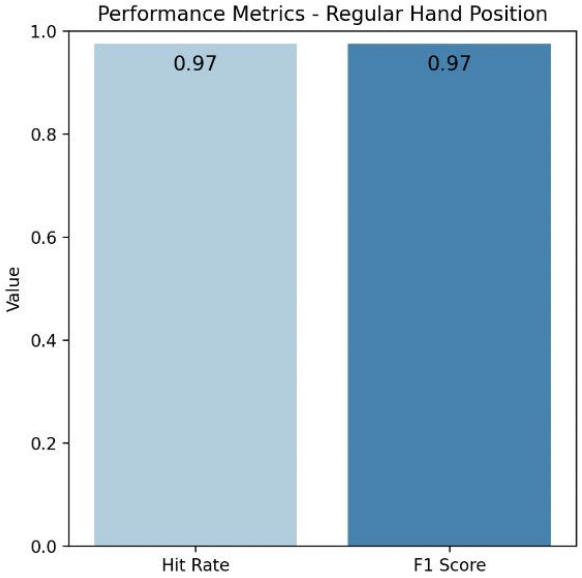
- **מסקנות**

מניתוח הנתונים ניתן להסיק כי האלגוריתם פועל בצורה מדויקת כאשר היד במצב ישר, אך קיימת ירידה בדיוק כאשר היד מוטה בזווית של 45 מעלות.
מסקנות אלו מצביעות על הצורך בהתאמות ושיפורים באלגוריתם כדי לשפר את עמידותו בפני שינויי זווית.

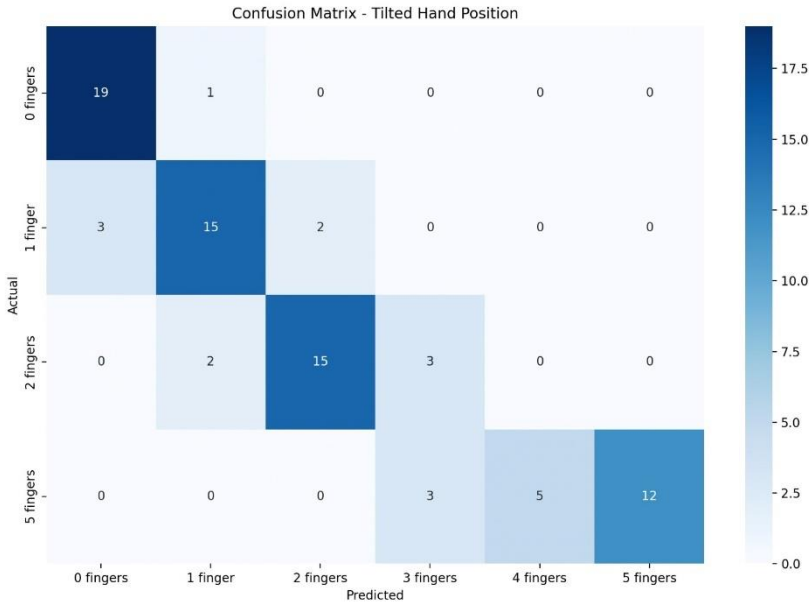
- **הצגת התוצאות**

יד ישרה:

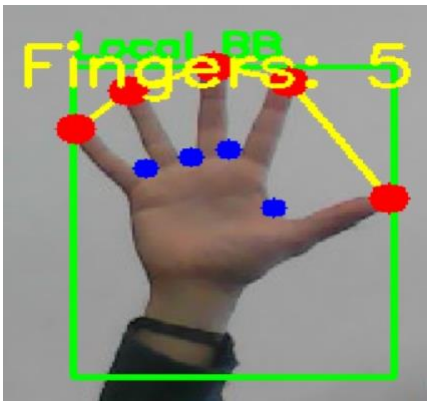


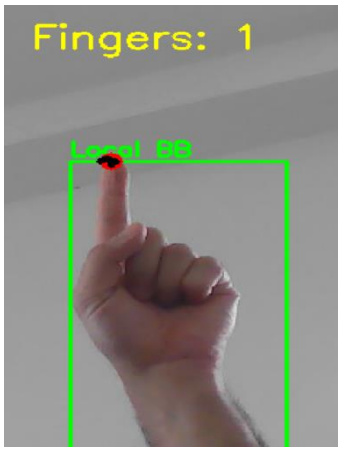

יד מוטה:



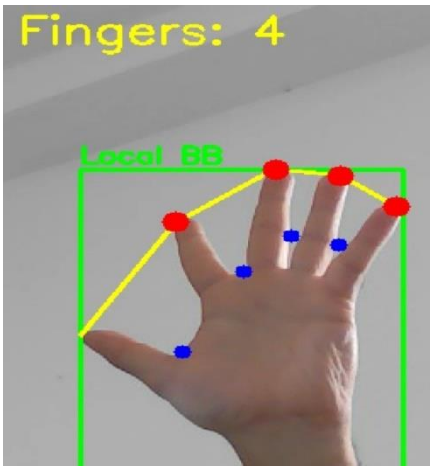

דוגמאות לסיווג נכון של האצבעות:



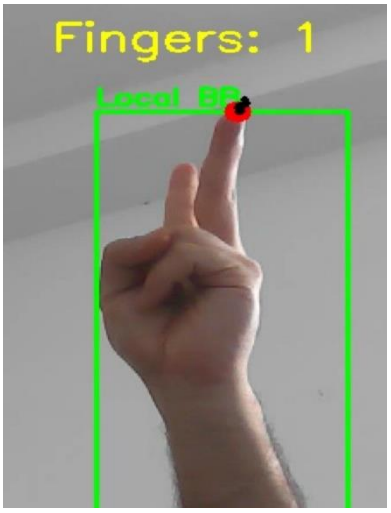




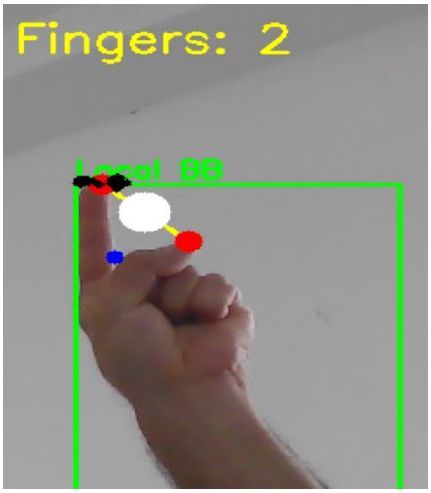

דוגמאות לסיווג לא נכון של האצבעות:




5 אצבעות שמסווגות כ -4 אצבעות




2 אצבעות שמסווגות כאצבע אחת




אצבע אחת שמסווגות כ-2 אצבעות

**סיכום**

בפרויקט זה פיתחנו מערכת אינטראקטיבית המיישמת טכניקות בראייה ממוחשבת לצורך עיבוד תמונה בזמן אמת. המערכת מזהה ומבודדת את היד בדיוק גבוה על ידי שילוב זיהוי צבע עור במספר מרחבי צבע (HSV ו-YCrCb) בשילוב עם אלגוריתמים לזיהוי קצוות כגון Canny וגישות מבוססות גרדיאנט. באמצעות ניתוח Convex Hull וזיהוי פגמי קמירות, נוכל לזהות בנוסף את מחוות הכף היד, ואת תנועת האצבעות.

במערכת משולב פילטר קלמן להבטחת מעקב יציב אחר תנועות היד, העוזר לעקוב אחרי היד למרות רעידות טבעיות ורעש מצלמה, ובכך ציור במשחק מתבצע באופן עקבי. בנוסף, פיתחנו מנגנון ניקוד מבוסס Quantitative Evaluation המספק משוב מהימן על דיוק שחזור הצורה.

באנלזית המערכת הראנו כי במצבי יד רגילים המערכת מדויקת ב-97%, עם ירידה לכ-85% בזוויות קיצוניות – מגבלה אותה ניתן לשפר בעתיד, בעזרת דברים כמו חילוץ מאפיינים בלתי תלויים בכיוון. פרויקט זה ממחיש כיצד ניתן לשלב אלגוריתמים בסיסיים בזיהוי קצוות, סגמנטציה ומעקב אחר אובייקטים ליצירת ממשקים אינטראקטיביים המגיבים למחוות אנושיות בזמן אמת.

**ביבילוגרפיה**

מצגות והרצאות הקורס מבוא לעיבוד ספרתי של תמונות.