**דו"ח הגשת מטלה 1**

**תרגיל 1 – Hough Transform on equilateral triangles:**

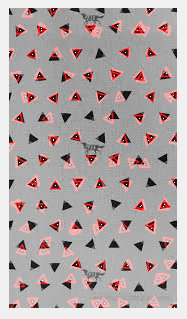
**תיאור שיטת הפרמטריזציה:**

שיטת הפרמטריזציה (כלומר הדרך בה ביצענו את ה-Voting) התבססה על תכונות של משולש שווה צלעות ושימוש בגרדיאנט שחושב לכל פיקסל בתמונה. השיטה התבצעה לכדלהלן:

מעבר על כל פיקסל שפה (Edge) שקיבלנו מ-Canny Edge Detector, כאשר בצעד הראשון חישבנו את מרכז צלע המשולש אליו הוא חשוד להיות שייך (לפי כיוון הגרדיאנט) ובצעד השני את נקודת מרכז המסה אליה שייך מרכז צלע המשולש אותו מצאנו בצעד הראשון, שני הצעדים על ידי חישובים טריגונומטריים. אנחנו מגדילים את ערך הנקודה ה-1 בטבלת הצבירה (Accumulator). בצורה כזו אנו אנחנו מחשבים מרכזי מסה פוטנציאליים עבור פיקסל השפה באיטרציה הנוכחית, על ידי שימוש האורכים שונים ממרכז הצלע (להלן גדלי steps) שחישבנו בצעד הראשון (מרכז הצלע לפי כיוון הגרדיאנט יכול להיות במקסימום חצי אורך צלע המשולש שנתון לנו **כלפי מטה וגם כלפי מעלה** ולכן אנחנו רצים על כל ערך בין מינוס חצי אורך הצלע עד אורך הצלע, שורה 154 בקובץ ה-Main של Hough Transform)

לאחר שצברנו את כל ההצבעות נבצע non-maximal suppression לטבלת הצבירה, ובכך נשמור את ה-peaks בטבלה ונצמצם אותה לכדי הנקודות (בגרדיאנט ספציפי) שקיבלו את מירב ההצבעות גם בשלב הצבירה ולאחר מכן גם על ידי החישוב המרחבי בגודל חלון נתון (משתנה בין תמונה לתמונה)

בשלב האחרון נחשב את ערכי 3 הנקודות A, B, C שמרכיבים את משולש שווה הצלעות בכל נקודת מרכז מסה שצברנו בטבלה (כמובן שנקודות A B C משתנות בין נקודת מרכז מסה לאחרת), גם זאת על ידי חישובים טריגונומטריים.



**Triangels\_1/ image002**

אורך הצלע – 10 פיקסלים, steps - -5 עד 5.

ניתן לראות את זיהוי המשולשים בתמונת הקלט מצד שמאל ומתחתיה את תמונת ה-edges.

ישנו זיהוי ברמה יחסית טובה כאשר מחצית מהמשולשים זוהו בצורה טובה מאוד, רבע

מהמשולשים זוהו בצורה לא טובה (צוירו כמה משולשים/אוריינטציה לא נכונה של המשולש/מיקום

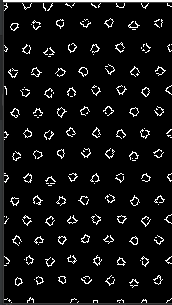
שגוי) ועוד רבע לא זוהו כלל.

הדבר נובע ככל הנראה מכך שהתמונה מכילה מספר רב של משולשים במרחקים לא גדולים

במיוחד האחד מהשני – מה שגורם לעיתים לזיהוי שגוי ולעיתים למעבר לא נכון של אלגוריתם

ה- non-maximal suppressionמה שגורם למחיקת מספר נקודות מרכז מסה פוטנציאליות

נוספות. היה לנו מאוד קשה להתגבר על הבעיה הנ"ל במימוש האלגוריתם ועל כן התוצאות הללו.

לדעתנו ניתן להתגבר על הבעיה על ידי הקטנת הצעד שעושים בין כל מעבר באלגוריתם, בצורה

הנוכחית אנחנו קופצים לפי גודל החלון בכל איטרציה ולא פיקסל פיקסל (כלומר לא stride = 1),

אכן ניתן לעשות מעבר שכזה אך הוא יגרור חישוביות גבוהה מאוד שלדעתנו לא כדאית בהתחשב

בתוצאות הכלל לא רעות שהתקבלו ללא מעבר שכזה.

**Triangels\_1/ image003**

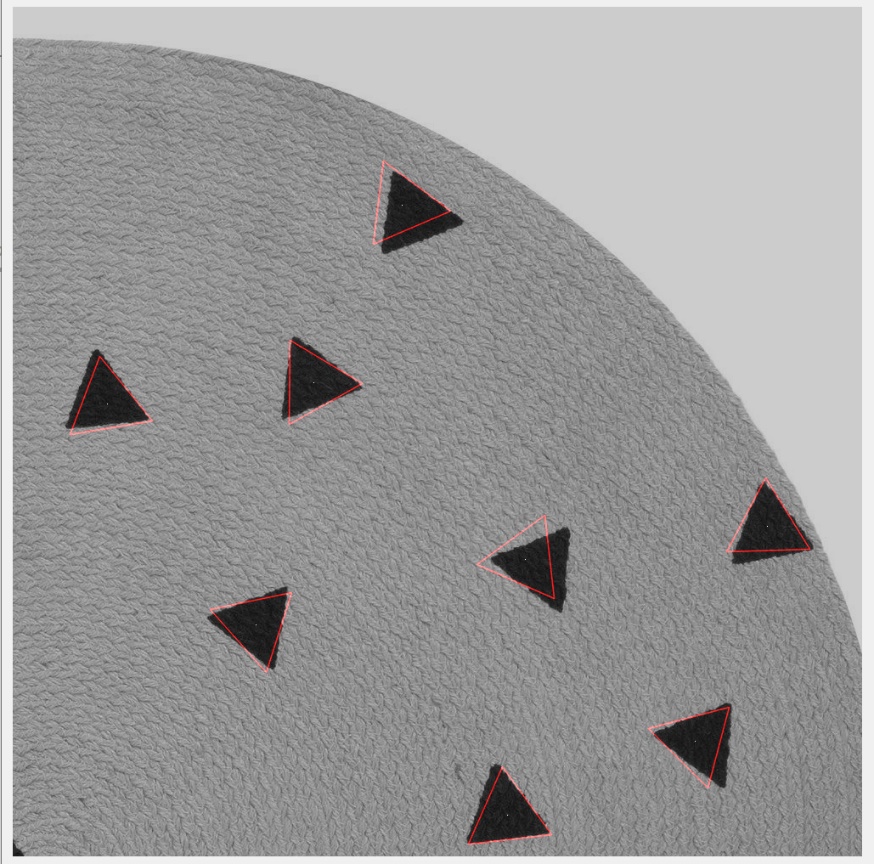
אורך הצלע – 99 פיקסלים, steps – 49- עד 49.

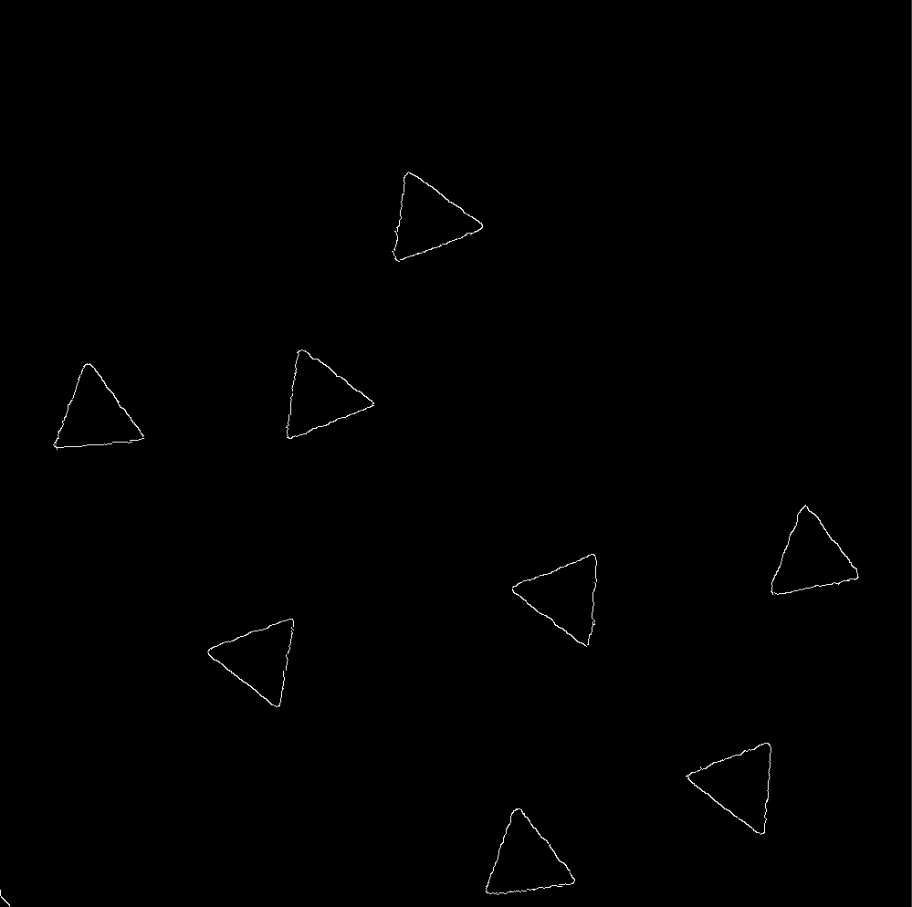
ניתן לראות זיהוי משולשים מושלם, כל המשולשים באורך הצלע הנתון זוהו בהצלחה (המשולש הקטן בקצה השמאלי התחתון לא באורך הנתון ועל כל לא זוהה) הבעיות שניתן לראות הן בחלק מהמשולשים הן אוריינטציה לא נכונה של המשולש המצויר, אבל בכמה מעלות בלבד ולא יותר, ובחלק אחר במיקום לא נכון של מרכז המסה וכתוצאה מכך מיקום לא נכון של המשולש המצויר עצמו.

הדבר נובע ככל הנראה שוב מאלגוריתם ה-non-maximal Suppression שנתפסו בו כמה נקודות מסה עם הצבעה שווה

אך לא המתאימות ביותר למשולש הנתון בתמונה.

הפתרון לבעיה הוא שוב מעבר על כל פיקסל ופיקסל, בדומה לפתרון בתמונה הקודמת, אך כאמור זהו פתרון עם עלות חישובית גבוהה הרבה יותר (זמן ריצה שעלול להגיע לשעה של ריצה אל מול הפתרון שהנוכחי שמאפשר זמן ריצה של דקה)



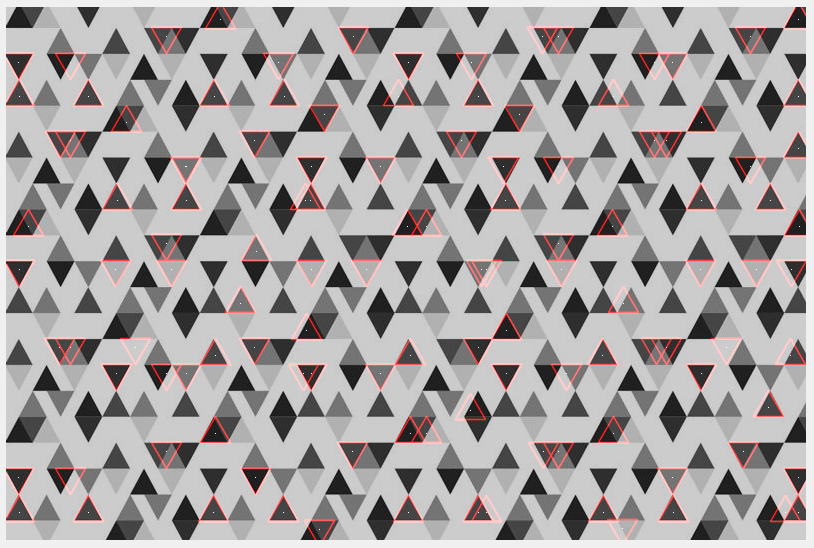


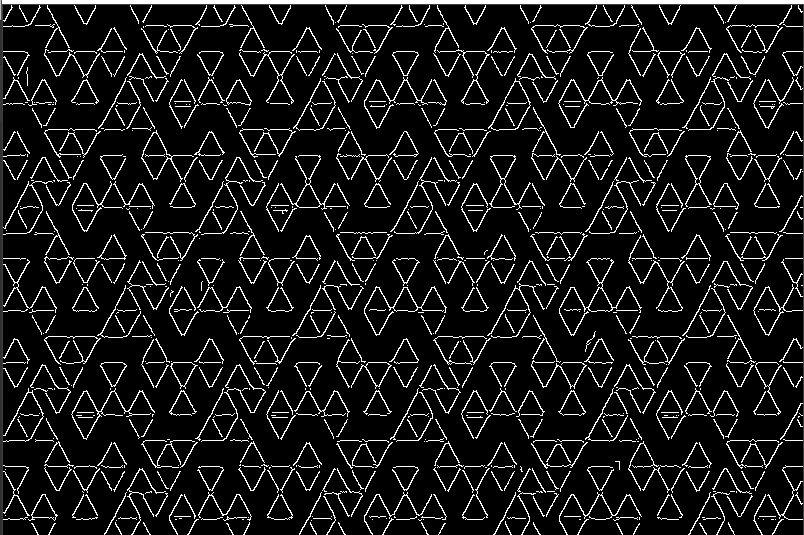
**Triangels\_2/ image001**

אורך הצלע – 10 פיקסלים, steps – 5- עד 5.

בדומה לתמונה הראשונה, גם כאן ישנם משולשים רבים וצפופים אך גם באורנטציות דומות מאוד. ניתן לראות שעל אף המשולשים הרבים מאוד לפחות חצי זוהו ורוב אלה שזוהו – זוהו היטב. במשולשים מעטים ניתן לראות משולשים כפולים, קצת בריחה במיקום אך לא מעבר.

הבעיה העיקרית שזיהינו בתמונה היא מספר המשולשים המועט יחסית שהתקבל, הסיבה היא שילוב של Threshold גבוה יחסית אל מול הבעיה החוזרת של ה-non-maximal suppression. מצד אחד היינו חייבים להגדיל את ה-Threshold בכדי למנוע כפילויות רבות וריבוי זיהויים לא נכונים, מצד שני הדבר הוריד מספר לא מועט של זיהויים נכונים אמנם שקיבלו הצבעות נמוכות יותר (אולי בגלל השפות הרבות שזוהו על ידי קני). בנוסף מציאת ה-peaks באזורים צפופים גרמה גם היא למחיקת נקודות מרכז מסה טובות – בעיה ששוב ניתן היה לפתור בתשלום של זמן חישוביות רב.



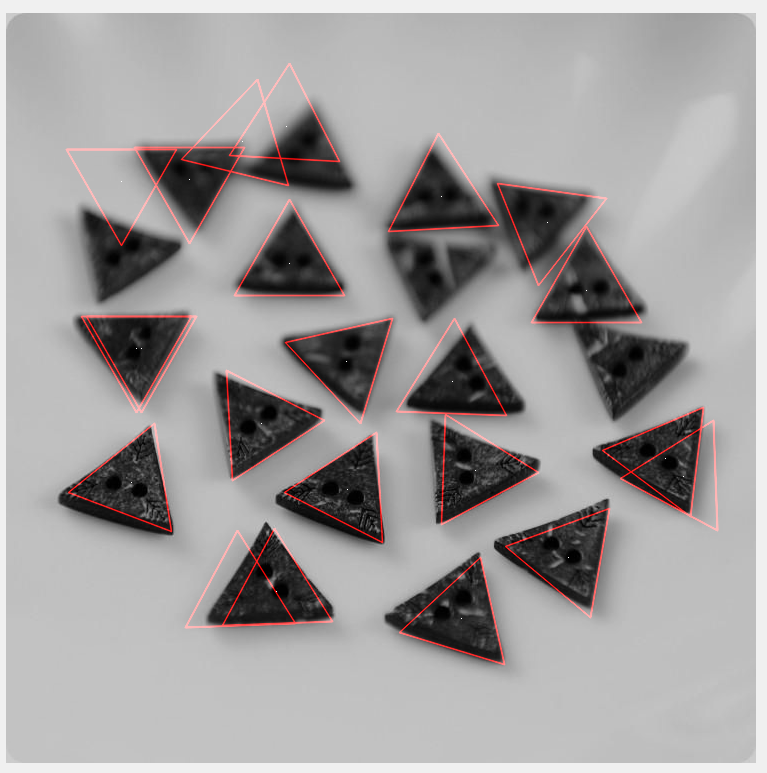


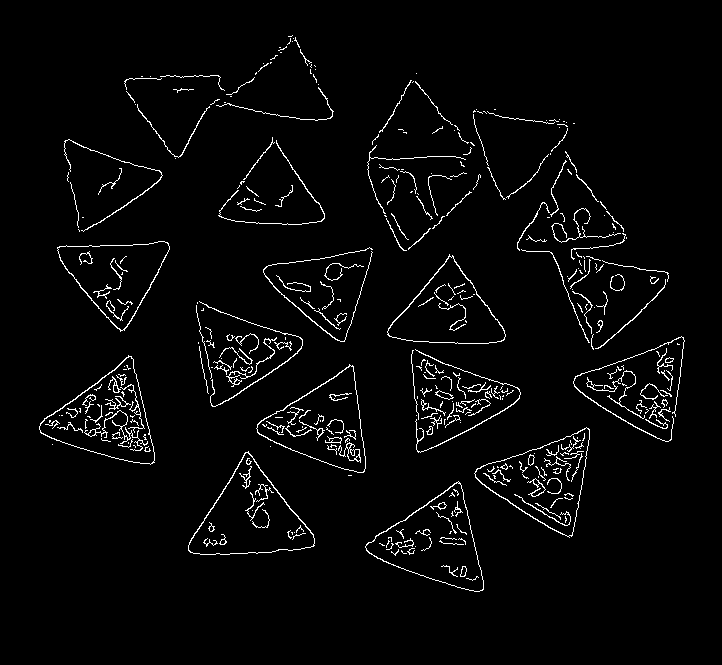
**Triangels\_2/ image006**

אורך הצלע – 110 פיקסלים, steps – 55- עד 55.

בתמונה זו התוצאות טובות מאוד; 80%-90% מהמשולשים מזוהים ואף בצורה טובה מבחינת מיקום ואוריינטציה.

במקרים בודדים יש כפילות של משולשים מזוהים ומיקומים לא נכונים, דבר זה נובע כלל הנראה מהטשטוש בחלק העליון של התמונה ושוב מגודל החלון והצעד (Stride) שנעשה בתהליך ה- non-maximal suppression.

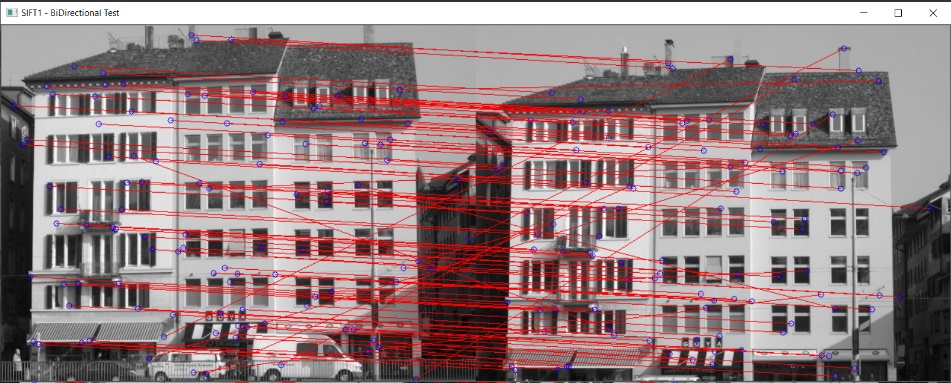
הבעיות המינוריות הללו היו יכולות להיפתר באותה צורה שהוסברה בסעיפים הקודמים – מעבר ב-Strideים קטנים יותר – אך הדבר היה עולה בהרבה זמן ריצה, וכמו במקרים קודמים, ואף יותר במקרה זה שהזיהוי כל-כך מוצלח; זמן הריצה הרב לא שווה את העלאת הדיוק.



**תרגיל 2 – SIFT-based correspondences :**

1. מצורף הקוד exeA.py אשר הרצתו ממחישה את אותה ויזואליזציה של key points על גבי התמונה UoH.jpg. כפי שניתן לראות בקוד לאחר חילוץ נקודות העניין, עבור כל נקודה חושבה הזווית ברדיאנים, חושב גודל וקטור ה- key point ובאמצעות שני הפרטים הנ"ל, הצלחנו למצוא את הנקודה שאליה נוכל למתוח את החץ.   
   לאחר מכן, עבור כל נקודה נמתח מעגל, כאשר נקודת העניין היא מרכז המעגל, ורדיוס המעגל הוא בגודל שחושב קודם לכן.  
   **לשים לב:**

* עקב ריבוי key points שמילאו את התמונה בנקודות אדומות וכתוצאה מכך לא היה ניתן לראות כמעט דבר, נלקחה תת קבוצה של נקודות עניין לתצוגה, ולשם כך נבחרו לתצוגה אלו בעלי גודל וקטור מספיק משמעותי.
* תמונת התוצאה המקורית מצורפת בשם Key Points UoH.jpg

1. מצורף הקוד exeB.py שמכיל בתוכו כלל ההתאמות שצלחו את מבחן ה- Ratio Test, והקובץ exeBi-directionalTest.py שמכיל בתוכו את כלל ההתאמות שעברו בהצלחה את ה- Bi-Directional Test.  
   את ההתאמות ניתן לראות באמצעות קווים המתוחים בצבע אדום, ואת הנקודות המותאמות מוקפות בעיגול בצבע כחול. כמובן, עקב ריבוי התאמות בחרנו להציג מדגם מייצג מתוכן (כעשירית).   
   את התמונות ניתן למצוא בתוך תיקיית ההגשה.
2. ניתן לראות כי ב- Bi-Directional Test יש לנו יותר Outliers. אותם outliers באים לידי ביטוי באמצעות התאמות מהחלק התחתון של התמונה השמאלית אל החלק העליות בתמונה הימנית.  
   התמונה הזו משקפת היטב את הייתכנות הפוטנציאלית ל- outliers מפני שישנן נקודות רבות בתמונה (כמו החלונות), שעלולות לגרום לאותו mismatching.  
   ה- ratio test מביא לנו תוצאות טובות יותר במקרה הזה מפני שהוא מסנן התאמות שעשויות להשתמע לכמה פנים. כלומר הוא מסנן את אלו שלא ניתן לקבוע איתן באופן חד משמעי לאן נקודת העניין בתמונה 1 תותאם בתמונה 2 בגלל קירוב או דמיון חזק בין מספר descriptorים שונים. ה- Bi-Directional Test לעומת זאת, לא עושה את האבחנה הזו אלא רק מתאים בין נקודות עניין שהן שכנות אחת של השנייה, ללא הקשר לכך שעלולות מספר נקודות שעשויות להיות שכנות של אותה נקודת עניין. לכן אנחנו נקבל כאן יותר outliers. דבר שכזה גם משליך באופן ישיר על כמות ההתאמות, שכן ב- Ratio Test נמצאו פחות התאמות לעומת ה- Bi-Directional Test (ניכר בעיקר בסט התמונות של pair2)

**תרגיל 3 – Image Warping and RANSAC-based fitting:**

**חלק 1:**

1. תמונת ה-frame החדשה שאליה הועתקה Dylan נמצאת בתוך התיקייה. הקובץ המבצע זאת נקרא Warping.py

**חלק 2:**

1. כלל התמונות נמצאות בתיקיית ההגשה. הקבצים שביצעו זאת נקראים AffineRANSAC.py ו- PerspectiveRANSAC.py.
2. ניתן לראות כי עבור התמונות בעלות הרזולוציות הגדולות כמעט ולא ניתן לשים לב בין ההבדלים אבל warping אפיני והומוגרפי וזאת משום שנמצאו הרבה התאמות בין זוגות התמונות, כך שהמיפוי התבצע בצורה טובה בכל מקרה. לעומת זאת עבור סט התמונות השלישי ההבדל ניכר מאוד. תחילה קיבלנו 77 התאמות בלבד מה שגרר כי עבור ה-warping ההומוגרפי קיבלנו תמונה יחסית טובה מפני שלטרנספורמציית ההומוגרפיה יש יותר דרגות חופש (8) מאשר לזו האפינית (6). לכן בהומוגרפיה אנחנו מקבלים תוצאה טובה שכן מצליחה ליצור את אפקט הפנורמה, ואילו באפיני ה-fitting לא מתבצע כמו שצריך בגלל המיעוט היחסי בדרגות חופש של הטרנספורמציה ומספר נמוך של התאמות.