**סגמנטציה של קטעי מגילות מדבר יהודה**

מאת עדי מלאכי (207740846) ואילת ג'יבלי (208691675)

חלק א:

בחלק זה נדרשנו להקיף את כל אחד מקטעי המגילה במסגרת אדומה. קיבלנו 7 תמונות בגודל ממוצע של 5000x4000. על מנת לפתור את הבעיה פיתחנו אלגוריתם שמתחלק לשלושה שלבים עיקריים.

**חלק ראשון - הורדת רעשים**

1. טשטוש התמונה בעזרת **Bilateral Filter.** בחרנו בפילטר זה מתוך הפילטרים האפשריים (gaussian, median) מכיוון שהוא מאפשר לנו לנקות את הרקע ואת פנים המגילה תוך כדי שמירת הקצוות בצורה המיטבית. הערכים שהעברנו לפונקציה:

d=15 , sigma\_color=45, sigma\_space=45



1. מעבר לתמונה בינארית וסינון רעשים על ידי **ערך סף אדפטיבי**. בחרנו בערך סף אדפטיבי מכיוון שבניגוד לערך סף בינארי שהינו גלובלי לכל התמונה, ערך הסף האדפטיבי משתנה באזורים שונים ועוזר למצבים שבהם התאורה(במקרה שלנו צבע הקלפים על הרקע) לא אחידה. הערכים שהעברנו לפונקציה:

maxValue=255, adaptiveMethod=cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, thresholdType=cv2.THRESH\_BINARY\_INV, blockSize=601, C=2



1. ניקוי חלקים לא רלוונטיים מהתמונה הבינארית:
   1. **חיתוך המסגרת החיצונית** של התמונה. מצאנו קווים ישרים ארוכים בעזרת אלגוריתם hough lines, ומיינו אותם לאופקיים בחצי העליון, אופקיים בחצי התחתון, אנכיים בחלק הימני ואנכיים בחלק השמאלי. לאחר מכן צבענו בשחור עד לקו האופקי המקסימלי מהחלק העליון, עד לקו האנכי המקסימלי מהחלק השמאלי, החל מהקו האופקי המינימלי מהחלק התחתון והחל מהקו האנכי המינימלי מהחלק הימני. במילים אחרות, ניסינו למצוא את מסגרת הדף ולמחוק עד אליה.

הערכים שהעברנו לפונקציה:

image = edges, rho = 1, theta = np.pi/180, threshold = 600



* 1. **חיתוך סרגל קנה המידה.** חתכנו את קנה המידה מאחת התמונות (ניסינו לחתוך מכמה עד שמצאנו את האחד שעובד הכי טוב עם המספר הרב ביותר של התמונות. בסוף התמונה היחידה שבה לא נמצא הסרגל היא תמונה 5), וחיפשנו אותו בכל התמונות בעזרת template matching (המבוסס על fft). הערכים שהעברנו לפונקציה: template, cv2.TM\_CCOEFF\_NORMED

ערך הסף להתאמה טובה הינו 0.5. לאחר בחינה של תמונות רבות ראינו שהסרגל מופיע תמיד בתחתית התמונה, ולכן מיקדנו את החיפוש בחמישית התחתונה של התמונה וצבענו בשחור את הריבועים שנמצאו + מתחתם. (צריך לשים לב ובמידה וכן תגיע תמונה שבה הסרגל נמצא בחלק העליון, זה ישבש את האלגוריתם)



**חלק שני - מציאת קצוות וצביעת החתיכות**

1. מציאת הקצוות וצביעה ראשונית שלהם על גבי רקע שחור חלק בעזרת **findContours ו drawContours**. הערכים שהעברנו לפונקציה:

findContours: mode = cv2.RETR\_EXTERNAL, method = cv2.CHAIN\_APPROX\_NONE

drawContours: result =np.zeros(size),contours = findContours, contourIdx = -1, color=255, thickness=cv2.FILLED



1. הפעלה של פעולות מורפולוגיות על מנת לסגור חורים:
   1. מילוי של הצורות בעזרת **dilate** עם kernel בצורת אליפסה (כדי למלא חורים) של 5x5 ומס' איטרציות 4.



* 1. ניקוי בעזרת **erode** עם kernel בצורת אליפסה של 7x7 ומס איטרציות 4 (הגענו לפרמטרים אלה לאחר ניסויים רבים).



1. הרחבה נוספת בעזרת **medianBlur על התמונה הבינארית** - מאחר וטשטוש זה משתמש בממוצע הוא עוזר מאוד לסגור חורים קטנים במיוחד שנוצרו מאותיות. הערכים שהעברנו לפונקציה: kernel\_size=3



1. לאחר מכן נדרש ניקוי נוסף ואחרון בעזרת **erode**, הפעם עם קרנל בצורת צלב (על מנת שינתק חתיכות שהתחברו) בגודל של 7x7 ומס איטרציות 4.

מצאנו שככל שמצליחים לסגור את החורים ולצבוע את המגילות בצורה יותר מדויקת, המלבנים יוצאים הרבה יותר מדויקים, ולכן ניסינו הרבה שילובים שונים ואלגוריתמים שונים לסגירת החורים מבלי לחבר את החתיכות הקטנות.

**חלק שלישי - מציאת המלבנים החוסמים, מיונם וסימונם בתמונה**

1. **מציאת המלבנים** החוסמים לכל הקצוות בתמונה בעזרת **findContours ו boundingRect** ושמירתם ברשימה. הערכים שהעברנו לפונקציה:

findContours**:** mode **=** cv2.RETR\_CCOMP, method = cv2.CHAIN\_APPROX\_NONE

1. **ניקוי וסידור רשימת המלבנים**:
   1. מחיקת מלבנים **קטנים מדי או גדולים מדי** לפי ערכי min = 40, max = 3000.
   2. מחיקת מלבנים **ארוכים וצרים** לפי abs(h//w) < 7.
   3. מחיקת מלבנים **קרובים מדי לקצוות** לפי border\_size = 150.
   4. מחיקת מלבנים **המכילים רק רקע לבן** (פחות מ10 פיקסלים כהים - מתחת ל180).
   5. מחיקת מלבנים **שמוכלים בתוך מלבן אחר**. (פעולה זאת לעיתים נדירות מוחקת סימון של מגילה הנמצאת בתוך ריבוע של מגילה אחרת, אך זהו סיכון שלקחנו על מנת להוריד סימנים של אותיות בתוך מגילות).
   6. **מיון המלבנים** לפי המרחק מהפינה השמאלית העליונה של התמונה לפינה השמאלית העליונה של המלבן.
2. ריצה על הרשימה המסודרת, **ציור של כל מלבן** בעזרת rectangle **וכתיבת מיקומו** בעזרת putText בצבע אדום על התמונה הצבעונית המקורית.



1. **ייצוא קואורדינטות המלבנים** לקובץ q1\_output.xlsx.
   1. כל טאב מייצג תמונה אחרת.
   2. מספר השורות בקובץ הוא כמספר קטעי המגילות שמצאנו.
   3. סדר השורות לפי המיון שנעשה בשלב 9f.
   4. העמודות הם: piece, rectangle coordinates כאשר בעמודה השניה ישנם ארבעה מספרים מופרדים בפסיק שהם הקואורדינטות של הפינה השמאלית העליונה והפינה הימנית התחתונה של המלבן החוסם את קטע המגילה.

**מסקנות:**

התהליך כלל המון ניסוי ותהיה, קריאה על מגוון האלגוריתמים הקיימים ומחשבה במה להשתמש, בחינה של כל שלב ושלב ובדיקה האם הוא הכרחי, מה צריך להיות השלב הבא, האם יש להוסיף עוד שלב בין לבין וכו'. בדרך, היינו צריכות להניח הנחות כלליות על התמונות (למשל: סרגל קנה המידה תמיד יהיה למטה, אין חתיכות שקטנות מ40x40 או גדולות מ3000x3000, אין חתיכה לגמרי לבנה וכו') ולהחליט מה יותר חשוב לנו.

התפשרנו על רמה גבוהה וזהה של זיהוי בכל התמונות, מאשר התאמה של האלגוריתם והפרמטרים השונים לתמונה אחת ב 100%. הדבר בולט במיוחד בכוונון הפרמטרים של הפעולות המורפולוגיות, מאחר והם מחברים / חותכים חתיכות קיצוניות ייחודיות לכל תמונה, ולכן העדפנו שכמה חתיכות יתחברו תחת מלבן אחד מאשר שחתיכה תחתך באמצע. בנוסף, החלטנו שעדיף שהאלגוריתם יסמן רעשים בנוסף לחתיכות, מאשר שיפספס חתיכה. בסופו של דבר עדיין נשארו מעט רעשים ברקע שהאלגוריתם מסמן, אך ניסינו להוריד כמה שאפשר בעזרת הבנה של מכנים משותפים.

בשביל לנתח כל שלב בתהליך, שמרנו כל שלב בנפרד בתוך אובייקט מיוחד והצגנו אותם אחד לצד השני בסיום האלגוריתם. הסדר בקוד והקלות להוסיף/להוריד שלבים עזרו לנו מאוד בייעול תהליך הפיתוח והתמקדות במסקנות.

בסופו של דבר, האלגוריתם שלנו עובד היטב על תמונות עם המון חתיכות קטנות צפופות שחלקן בהירות/חלקן כהות, כי אלו התמונות שקיבלנו ומה שלמדנו מהתהליך זה כי יתכן והוא לא יעבוד היטב על תמונות אחרות שיהיו שונות מאוד.

**התוצאות הסופיות:**

עבור תמונה 1:

עבור תמונה 2:



עבור תמונה 3:



עבור תמונה 4:



עבור תמונה 5:

עבור תמונה 6:



עבור תמונה 7:

חלק ב:

בחלק זה נדרשנו למצוא contour צמוד ומדויק כמה שיותר לכל אחד מהקטעים שמצאנו בחלק א. על מנת לפתור את הבעיה, פיתחנו את האלגוריתם הבא:

1. עבור בלולאה על כל המלבנים שמצאת בחלק א בשביל התמונה(קראנו את קואורדינטות המלבנים מקובץ האקסל שנוצר בסעיף א'). עבור כל מלבן בצע:
   1. **חתוך את המלבן** עם תוספת של pad קטן **מהתמונה המקורית** (הpad מיועד על מנת שלא יחתך הקצוות במידה והמלבן יושב בדיוק על הקצה).
   2. הפעל **bilateral Filter** על מנת לטשטש את התמונה.

הערכים שהעברנו לפונקציה:

d=15 , sigma\_color=100, sigma\_space=100

* 1. הפעל **adaptive Threshold** על מנת להפוך את התמונה לבינארית.

הערכים שהעברנו לפונקציה:

maxValue=255, adaptiveMethod=cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, thresholdType=cv2.THRESH\_BINARY\_INV, blockSize=751, C=3

* 1. מצא ומלא את החורים בעזרת **findContours** ו **drawContours**.

הערכים שהעברנו לפונקציה:

findContours: mode = cv2.RETR\_EXTERNAL, method = cv2.CHAIN\_APPROX\_NONE

drawContours: result =np.zeros(size),contours = findContours,

contourIdx = -1, color=255, thickness=cv2.FILLED

המחשה של פעולת האלגוריתם עד שלב זה:



**בשונה מחלק א, לא הפעלנו אף פעולה מורפולוגית בחלק זה על מנת לא להרוס את הconture המדויק!**

* 1. מצא את הקצוות החיצוניות בעזרת **findContours**.

הערכים שהעברנו לפונקציה:

findContours**:** mode **=** cv2.RETR\_EXTERNAL, method = cv2.CHAIN\_APPROX\_NONE

* 1. **סנן קצוות לא רלוונטיות** על ידי חישוב המלבן החוסם, ומחיקה של קצוות קטנות מדי (רעשים) או צמודות למסגרת (חלק קטן מחתיכה אחרת שנכנסה למלבן).שלב זה הוא מאוד בעייתי ועדין. יתכן וימחק ה contour של החתיכה אם בשלב הקודם המלבן חתך את החתיכה באמצע. אבל אם משאירים את כל ה contour בתוך המלבן, בשורת האקסל של אותה חתיכה יכנסו חתיכות אחרות. בסופו של דבר העדפנו להוריד רעשים ולקבל contour סופי יותר מדויק.



* 1. **הוסף לערך הקואורדינטות** של ה contour שנוצר במלבן היחסי **את המיקום של המלבן** בתמונה הכללית.
  2. הוסף את הconture של **חתיכה זו לרשימה הכללית**.

1. עבור על הרשימה הסופית **וצייר** כל אחד מה contour על התמונה המקורית.
2. עבור על הרשימה הסופית **וכתוב** את ה contour בקובץ q1\_output.xlsx
   1. כל טאב מייצג תמונה אחרת.
   2. מספר השורות בקובץ הוא כמספר קטעי המגילות שמצאנו.
   3. סדר השורות הוא לפי המיון שנעשה בחלק א'.
   4. העמודות הם: piece, contour כאשר בעמודה השניה ישנם כל הנקודות הנמצאות על הקצוות כתובות בפורמט (x,y) ומופרדות בפסיק.

**מסקנות:**

שמנו לב שמאחר ואנחנו עובדות על המלבנים שמצאנו בסעיף קודם, מלכתחילה איננו מתייחסות להמון רעשים וקטעים לא רלוונטים בתמונה שכבר נוקו בחלק א'! אך החיסרון הוא שחתיכות שלא זוהו טוב בחלק א' לא יזוהו בצורה המיטבית שוב. בנוסף, היה צורך בכוונון קטן נוסף של הפרמטרים מכיוון שהפעולות מתבצעות מקומית על ריבוע קטן אך זה הוביל לכך שהאלגוריתם עובד יותר מדויק ומהר. **התוצאות הסופיות:**

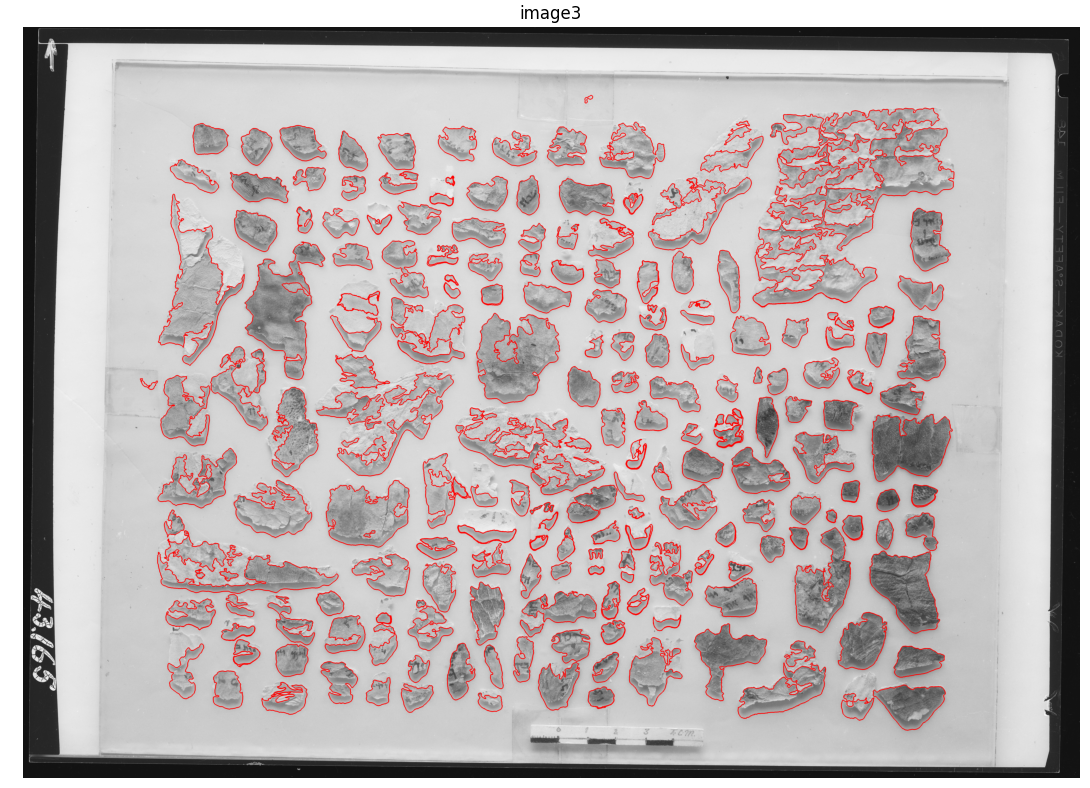
עבור תמונה 1:



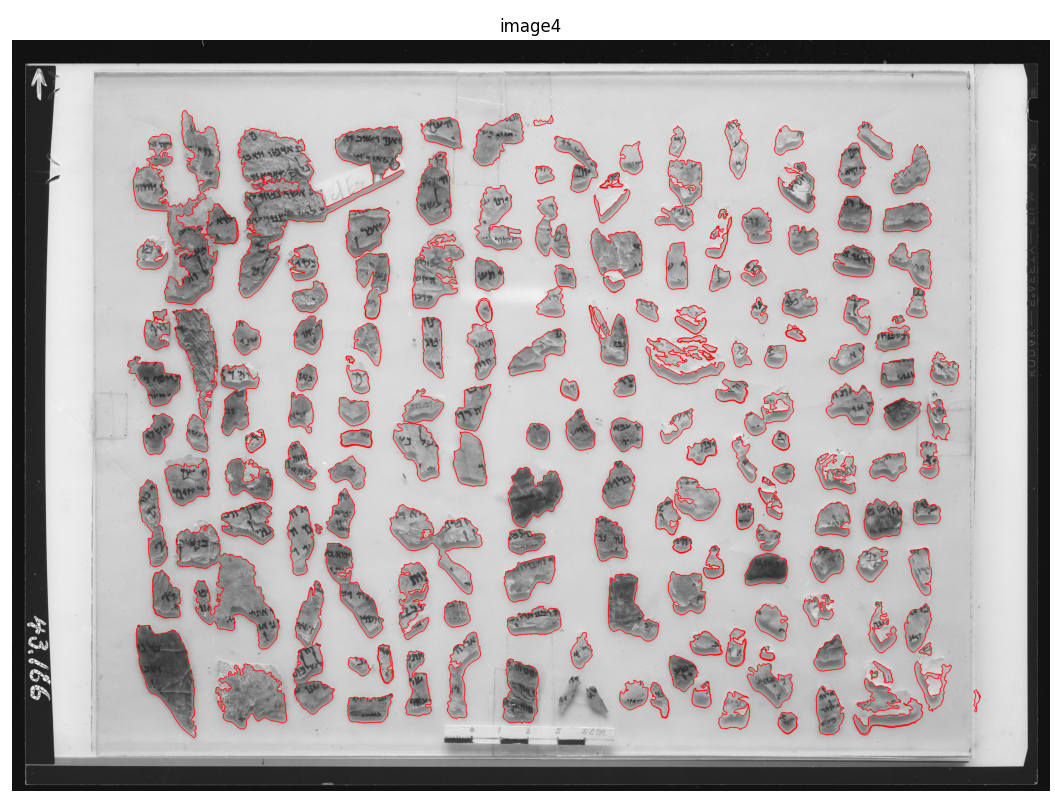
עבור תמונה 2:



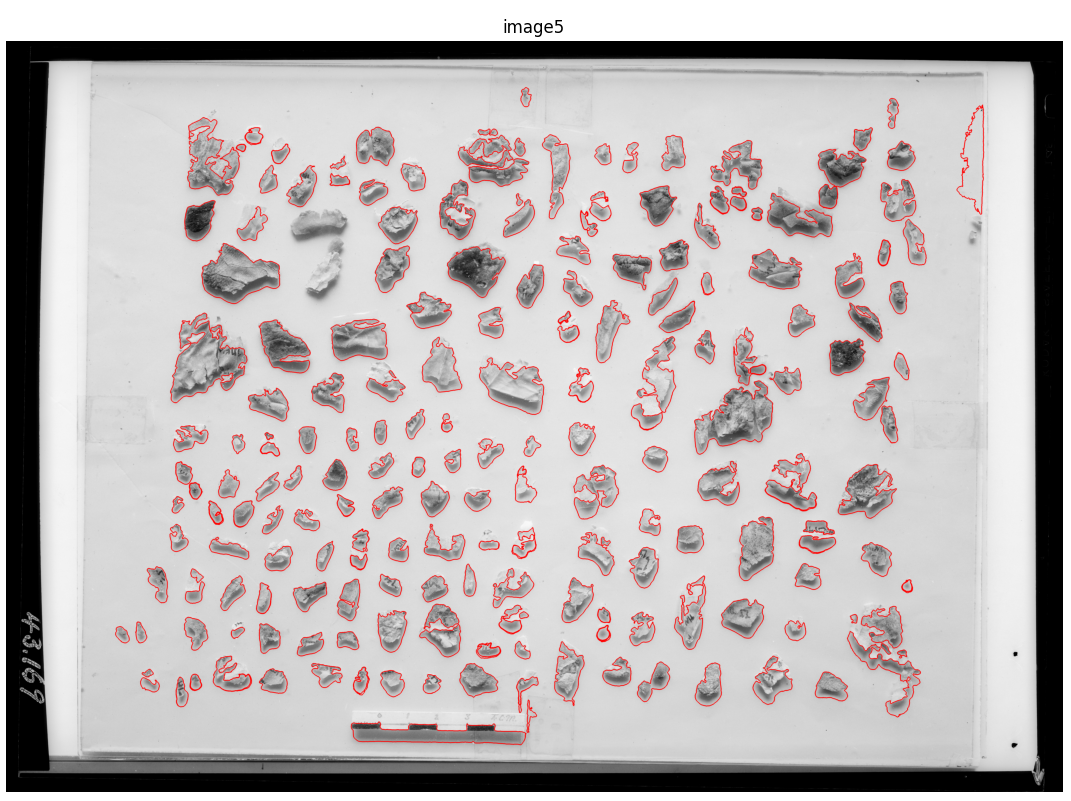
עבור תמונה 3:



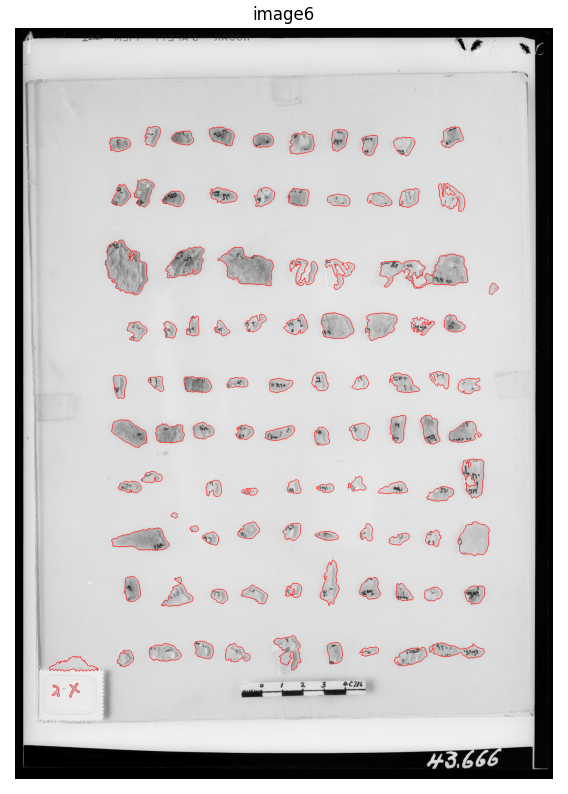
עבור תמונה 4:



עבור תמונה 5:



עבור תמונה 6:



עבור תמונה 7:

