גילוי שורות

על מנת לזהות כתב יד ישנו צורך בגילוי חלקי המסמך מתוך התמונה בסדר מסוים. בשלב הראשון בתהליך הגילוי (השלב בו נתמקד במסמך זה) נחפש אחר האלגוריתם האופטימלי לגילוי שורות הטקסט. כדי למצוא את האלגוריתם הטוב ביותר נחפש אחר הפתרונות המוכרים לבעיה כיום, נשתמש באלגוריתמים הקיימים ונפתח אלגוריתם חדש לגילוי שורות. לבסוף נבצע השוואה סטטיסטית בין תוצאות האלגוריתמים השונים.

תחילה בדקנו בספרות מה הם האלגוריתמים בהם משתמשים כיום לצורך הפרדת השורות בתמונה המכילה טקסט הכתוב בכתב יד. במהלך סקר השוק ראינו פתרונות רבים לבעיה ולבסוף החלטנו להתמקד בשלושת האלגוריתמים הנפוצים ביותר שראינו.

1. **MSER – maximally stable extremal regions**

MSER זוהי שיטה לזיהוי כתמים בתמונה, טכניקה זו פותחה במקור כדי למצוא התאמות בין רכיבי תמונה משתי תמונות עם נקודות מבט שונות, וכיום משתמשים בה גם לצורך גילוי טקסט. [1]

האלגוריתם משתמש בסף הזזה (threshold) המשלב פיקסלים לתוך כתמים גדולים יותר בתמונה. האזורים בתמונה נקבעים על ידי החלת סף ההזזה שבו הפיקסלים בעלי עוצמות גבוהות ומשולבים לאזורים קיצוניים.

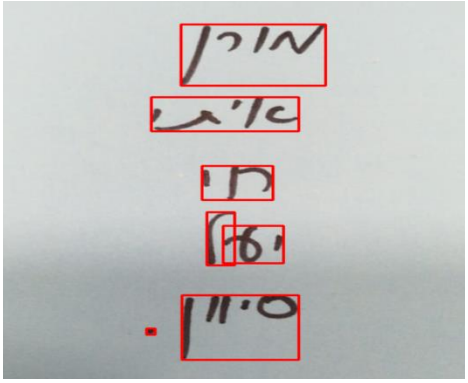
אזור קיצוני הוא אזור בו הפיקסלים הפנימיים של האזור בעלי עוצמה גבוהה יותר מהפיקסלים הנמצאים על גבול האזור, ואילו אזורים מקסימליים הם אותם אזורים שנותרו פחות או יותר באותו גודל עבור מספר ספים שונים.

לאחר מכן אנחנו מקבלים אזורים מקסימליים גדולים בתמונה ואזורים מקסימליים אחרים קטנים. כאשר האזורים המקסימליים הגדולים מקיפים את הקטנים, על ידי כך האלגוריתם יוצר עץ שהאזורים הקטנים הם הילדים של האזורים הגדולים ועל ידי כך לזהות אובייקטים בתמונה. במקרה שלנו אובייקט זוהי שורה.

האלגוריתם שמצאנו משתמש בשיטה MSER מהספרייה CV2 של openCV, באמצעות שיטה זו האלגוריתם מזהה את האזורים בתמונה שמהווים אובייקטים ומסמן אותם במלבנים.

לאחר מכן ממיין את המלבנים לפי ציר ה Y של כל מלבן על מנת לקבל סדר נכון של השורות.

<u>בעיות:</u> מספר שורות יכולות להיות מזוהות כשורה אחת. לעתים שורות או חלקים מהן מזוהים פעמיים.



דוגמה לתוצאה של ריצת האלגוריתם

2. **FindContours:**

contours הם קווי מתאר שהם עקומה סגורה של נקודות המייצגות את גבולות האובייקט בתמונה.

האלגוריתם משתמש בפונקציה findContours מתוך הספרייה cv2 של openCV כדי למצוא את האובייקטים בתמונה – במקרה שלנו שורות.

על מנת למצוא שורה שלמה בתור אובייקט ולא חלקים ממנה (שהם גם אובייקטים - מילים, אותיות) האלגוריתם קודם יבצע dilation לתמונה[2], כלומר התרחבות מורפולוגית - הוספה

של פיקסלים לגבולות האובייקטים בתמונה וכך נגרום לשורות להיות גוש של פיקסלים אחיד ללא רווחים בין האותיות והמילים.
לאחר ביצוע dilation ושימוש בפונקציה findContours האלגוריתם יזהה כל שורה בתור אובייקט ויסמן אותה במלבן.

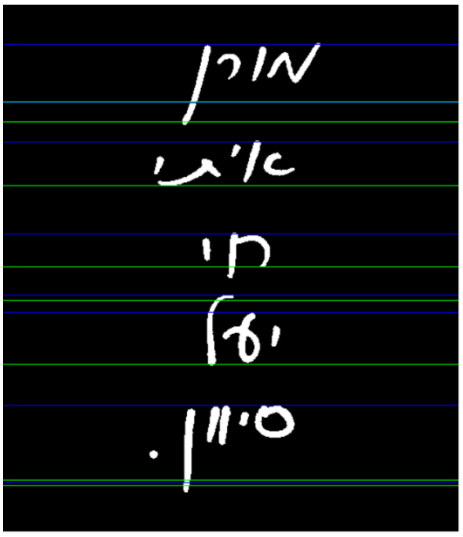
<u>בעיות:</u> בגלל שלב "מריחת" השורות, ייתכן מצב בו מספר שורות יתחברו וכך יזוהו כשורה אחת.

3. **<u>Reduce:</u>**

האלגוריתם מחולק לשלושה שלבים עיקריים שהם [3]:

- המרת התמונה לקנה מידה אפור, קביעת הסף הטוב ביותר (threshold), המרת התמונה לדימוי בינארי, הסרת רעש מהתמונה.
- מחולק למס' תהליכים אשר קובעים את הגבול העליון ואת הגבול התחתון של התמונה עצמה, קביעת גבול עליון עבור כל שורה וגבול תחתון על ידי הקרנה של הגבול העליון לתחתון, תיוג אובייקט ושינוי גודל, ואת מדידת הדמיון. שלב זה מתבצע על ידי פונקציה reduce מתוך הספרייה cv2 של openCV.
- ציור הגבולות (עליון- כחול ,תחתון- ירוק) עבור כל שורה על התמונה המקורית.

<u>בעיות:</u> זיהוי שורות שאינן שורות בפועל ולעתים זיהוי חלקי בלבד של שורה עקב רגישות גבוהה מדי של הפונקציה reduce.



תוצאת הרצת האלגוריתם, כאשר הקווים הכחולים הם גבול עליון של שורה, והקווים בירוק הם גבולות תחתונים של השורות

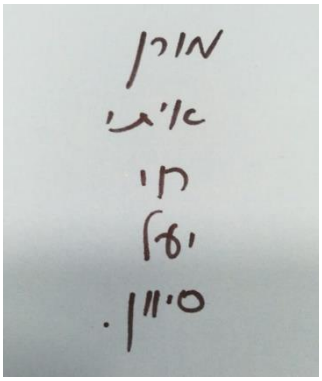
4. <u>סכימת פיקסלים (אלגוריתם שלנו):</u>

האלגוריתם מבוסס על ההנחה כי הקלט הוא תמונה המכילה טקסט (בכתב יד או מודפס), ועבור כל שתי שורות בטקסט ניתן לבצע הפרדה על ידי קו ישר אופקי אופטימלי המפריד ביניהן. גם כאשר שתי שורות מתחברות, כלומר ישנם פיקסלים חופפים בין השורות, המטרה היא למצוא את ההפרדה הטובה ביותר בין שורות הטקסט שבתמונה על ידי קו ישר אופקי.

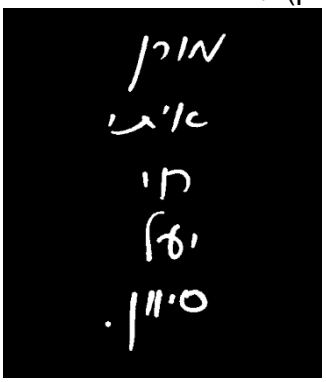
הרעיון הוא לסכום את הפיקסלים של כל שורה בתמונה, ולשמור את תוצאות הסכימה בוקטור שממנו ניתן יהיה להסיק היכן נמצאות בתמונה השורות בטקסט.

<u>שלבי האלגוריתם</u>

תחילה המרנו את התמונה לקנה מידה אפור, והעברנו את התמונה לדימוי בינארי (שחור-לבן) על מנת לפשט את התמונה ולהפחית רעש מיותר.

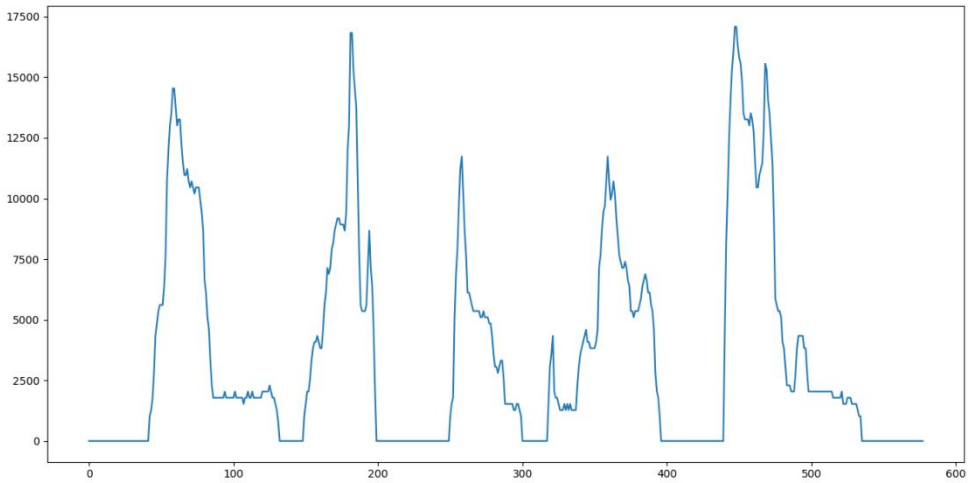


התמונה המקורית



התמונה לאחר המרה לבינארי

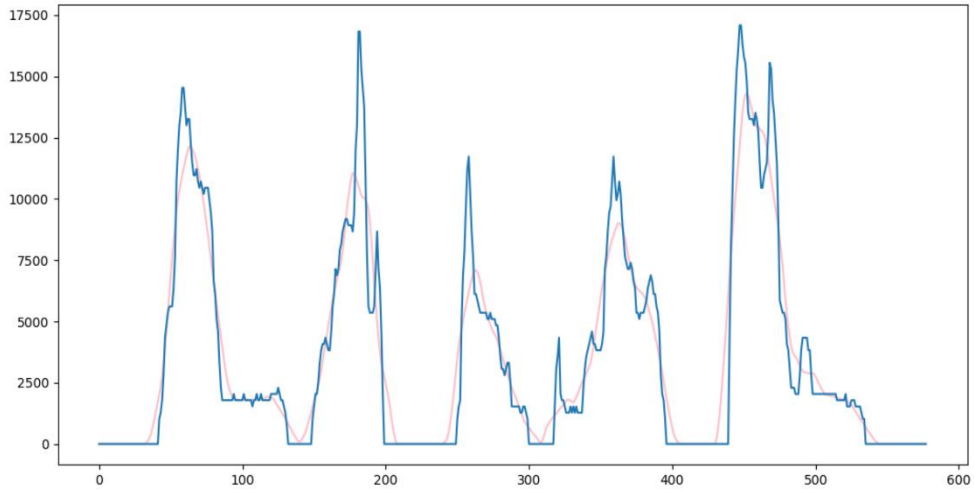
לאחר מכן עברנו על אורך התמונה וביצענו סכימה של כמות הפיקסלים בכל שורה, כך שבסוף קיבלנו מבנה נתונים המחזיק את כל ערכי הסכימה עבור כל שורת פיקסלים.



שרטוט סכימת הפיקסלים, כאשר הציר האופקי הוא מספר השורה (בפיקסלים) לאורך התמונה, והציר האנכי הינו ערך סכימת הפיקסלים עבור אותה שורה

על ידי התוצאות שקיבלנו מסכימת השורות נוכל להסיק היכן בתמונה נמצאים גבולות השורות. המטרה כאן היא למצוא באופן אוטומטי את מיקומי גבולות ה"הרים" בגרף שהתקבל - שהם גם מיקומי גבולות השורות לאורך התמונה. כדי למצוא אותם נרצה למצוא נקודות קיצון מסוימות, אך מהתבוננות בגרף, ניתן לשים לב כי יש הרבה "רעש" המקשה למצוא את נקודות אלו בתמונה.

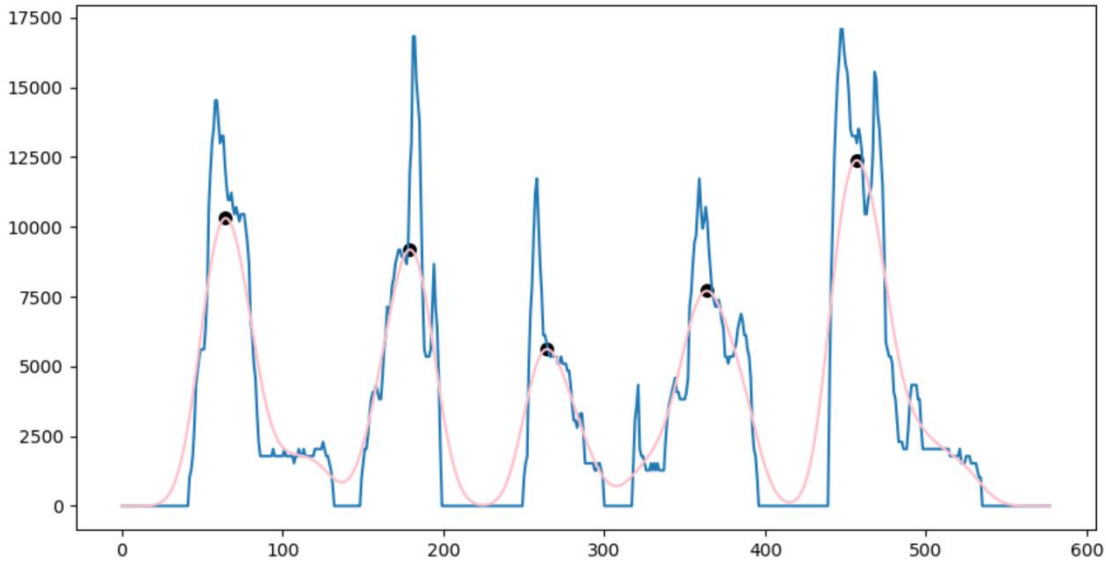
על מנת להתמודד עם בעיה זו ביצענו החלקה לגרף על ידי שימוש בפילטר savitzky-golay. השימוש בפילטר זה נעשה על ידי פונקציה מהספרייה scipy המקבלת מספר שלם אי זוגי כפרמטר למידת ההחלקה.



הפונקציה בורוד מתארת את הגרף לאחר ההחלקה.

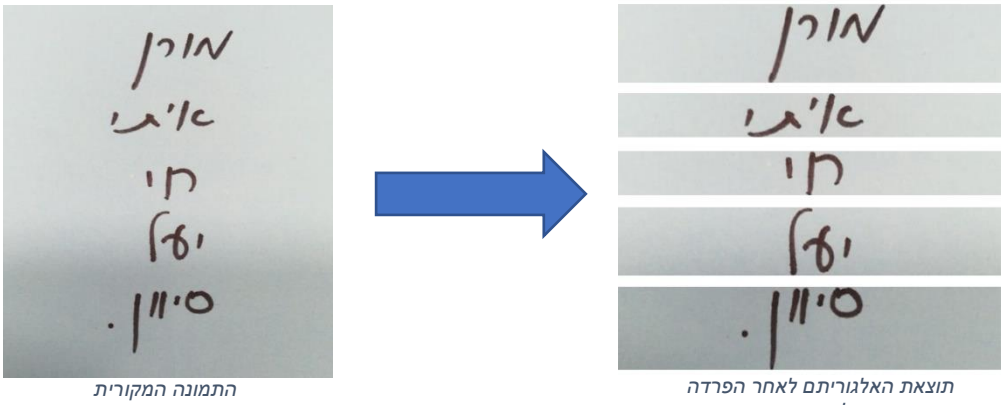
ניתן לראות כי למרות שיפור המצב כתוצאה מההחלקה שביצענו, עדיין קיים "רעש" מיותר בחלק מהשורות, ואנו נזהה בהם נקודות קיצון מיותרות שיגרמו לזיהוי שגוי של מיקומי השורות. במקרים אחרים ההחלקה שבוצעה הייתה "חזקה" מדי- דבר שעלול לגרום לאיבוד של שורות, ולכן שאלנו את עצמנו באיזה החלקה להשתמש כך שהאלגוריתם יתאים לקלט ארקאי.

כדי למצוא את ההחלקה הנכונה, ביצענו החלקות לגרף באיטרציות. התחלנו בהחלקה חלשה ובכל איטרציה הגברנו את פרמטר ההחלקה. בכל שלב מצאנו את נקודות המקסימום בגרף המתקבל כתוצאה מההחלקה, ושמרנו את חישוב המרחקים בין כל שתי נקודות מקסימום סמוכות. לאחר מכן חישבנו את החציון של מרחקים אלו, והשווינו אותו לחציון המרחקים בין נקודות המקסימום שהיו באיטרציה הקודמת. כאשר ההפרש בין החציונים של שתי איטרציות החלקה הוא מספיק קטן (על ידי סף שהגדרנו) נדע כי הגענו להחלקה הרצויה.



הגרף בורוד מתאר את ההחלקה בסיום האלגוריתם- ההחלקה הרצויה. ניתן לראות כי קיימת נקודת מקסימום אחת בכל פיק (מסומן בשחור). בטווח הרוחבי שבין כל שתי נקודות שחורות, נחפש נקודות מינימום בגרף המקורי (בכחול). נקודות אלו הן גבולות השורות.

כעת כשמצאנו את ה"פיק" של כל שורה (בדרך כלל באיזור מרכז השורה), נרצה למצוא את הגבולות של השורות, כלומר היכן כל שורה מתחילה והיכן היא מסתיימת. כדי לעשות זאת חיפשנו בטווח שבין כל שתי נקודות קיצון בגרף החלק את הנקודות הנמוכות ביותר בגרף המקורי (הגרף הכחול בתמונות). וכך לפי נקודות אלה, הפרדנו את השורות שבתמונה



תוצאת האלגוריתם לאחר הפרדה לשורות

התמונה המקורית

<u>בעיות:</u>

לעתים מזוהות שורות ריקות שאינן שורות בפועל. בנוסף ישנם קלטים בהם הסף (threshold) לפיו אנו ממירים את התמונה לבינארי אינו מתאים לתמונה ועקב כך גורם לחיתוך שורות במיקומים לא רצויים.

<u>בחירת האלגוריתם בו נשתמש</u>

על מנת לבחור את האלגוריתם הטוב ביותר נצטרך להשוות בין האלגוריתמים השונים. לשם כך אספנו מאגר של 40 תמונות המכילות טקסט- 30 מתוכם כתובים בכתב יד ו-10 בכתב מודפס. מאגר זה ישמש ללמידה והחלטה לצורך השוואה בין האלגוריתמים השונים.

כדי שנוכל להגדיר מדד מהי תוצאה טובה, עברנו על כל המסמכים ובדקנו היכן נמצאות השורות בטקסט: מצאנו בכל שורה בצורה ידנית היכן בתמונה נמצא הפיקסל של הגבול העליון של כל שורה, והיכן הגבול התחתון שבו אותה שורת טקסט מסתיימת.

לאחר ששמרנו את התוצאות הרצויות, אספנו את אותם הנתונים גם מתוך האלגוריתמים השונים, כלומר מהם הגבולות של השורות אותם האלגוריתמים מצאו.

כעת כדי לבצע את ההשוואה נצטרך להגדיר פונקציית מחיר לפיה נוכל לקבוע איזה אלגוריתם יותר טוב.

בהינתן שורה אופטימלית L1, ושורה L2 אותה מצא אלגוריתם מסויים, רצינו לקבוע נוסחה לקביעת ציון התאמה עבור כל שורה שאלגוריתם מצא, כך שעבור איבוד של פיקסלים או עבור מידע עודף, ציון ההתאמה ירד בהתאם. הנוסחה שנקבעה בסופו של דבר:

$$Line\ score = \frac{\mathrm{L1} \cap L2}{L1\ \cup L2}$$

מנוסחה זו נקבל מספר בטווח [0,1], שיהווה את אחוזי הדיוק עבור השורה שהאלגוריתם מצא.

עבור כל מסמך (המכיל N שורות), מצאנו את ממוצע ציוני השורות שכל אלגוריתם קיבל עבור אותו המסמך. התוצאה שהתקבלה היא ציון התאמת האלגוריתם עבור גילוי השורות באותו מסמך:

$$\text{Document score} = \frac{1}{N}\sum_{i=0}^{N} \text{Line score}_i$$

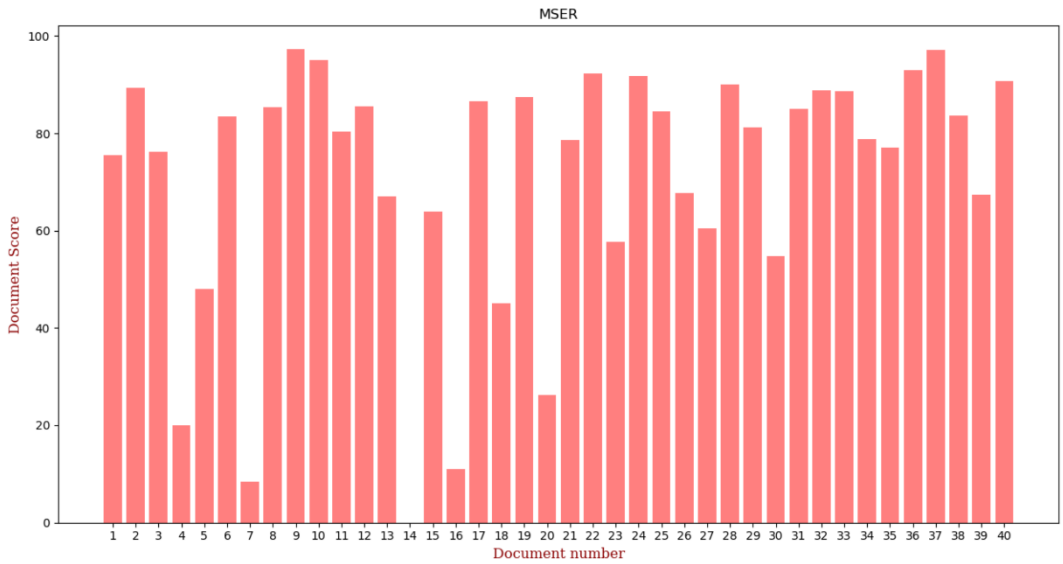
לבסוף, הציון של כל אלגוריתם נקבע לפי ממוצע הציונים שהוא קיבל עבור כל 40 המסמכים במאגר:

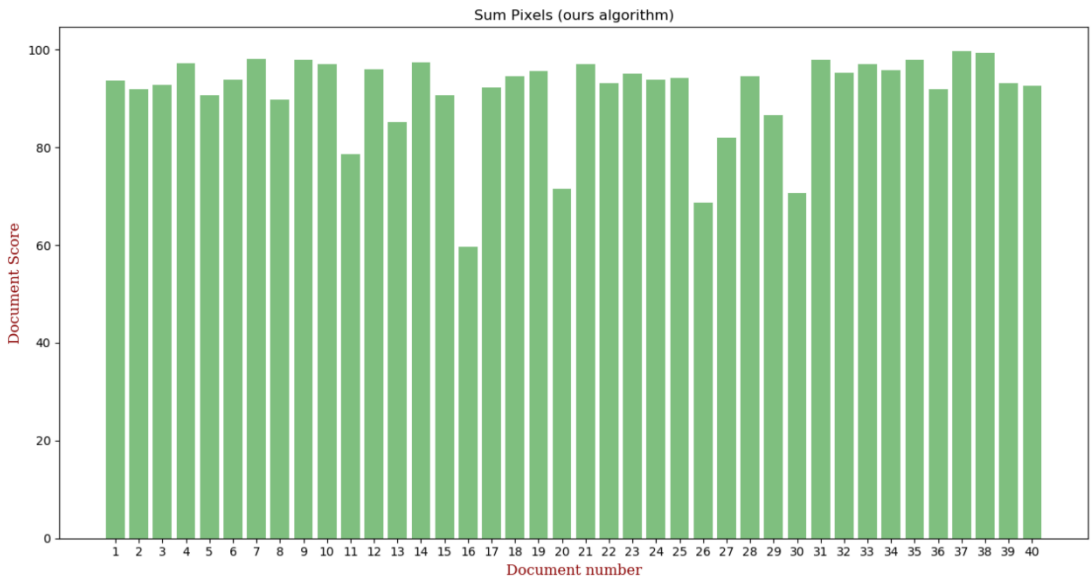
$$\text{Algorithm score} = \frac{1}{40}\sum_{j=0}^{40} \text{Document score}_j$$

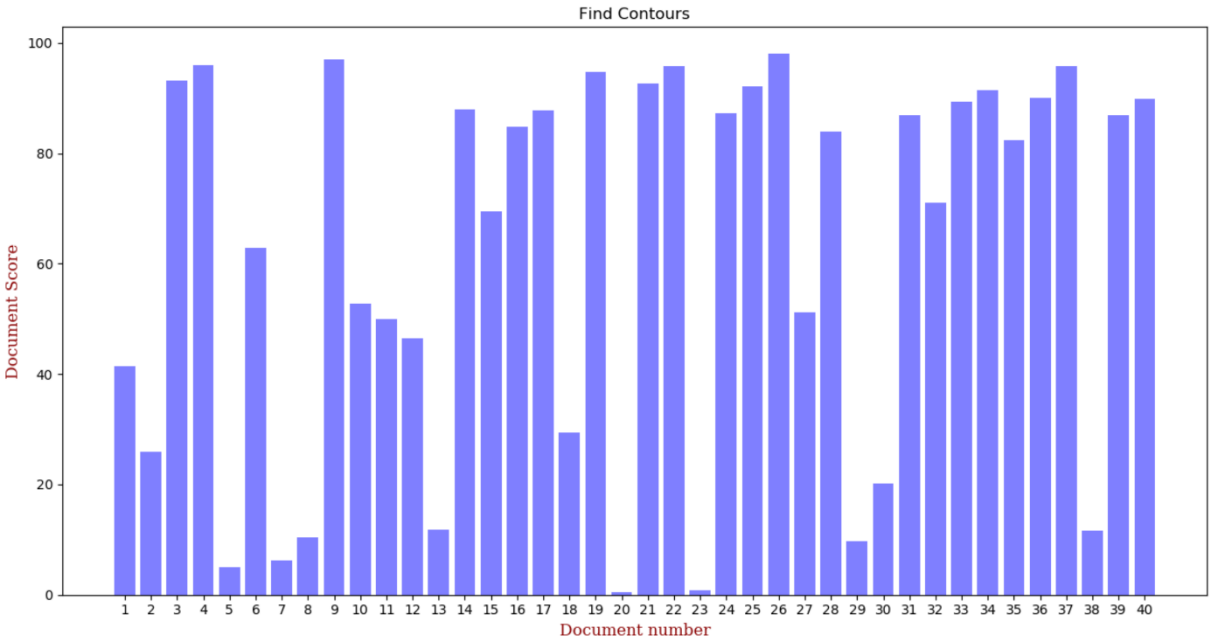
<u>תוצאות</u>

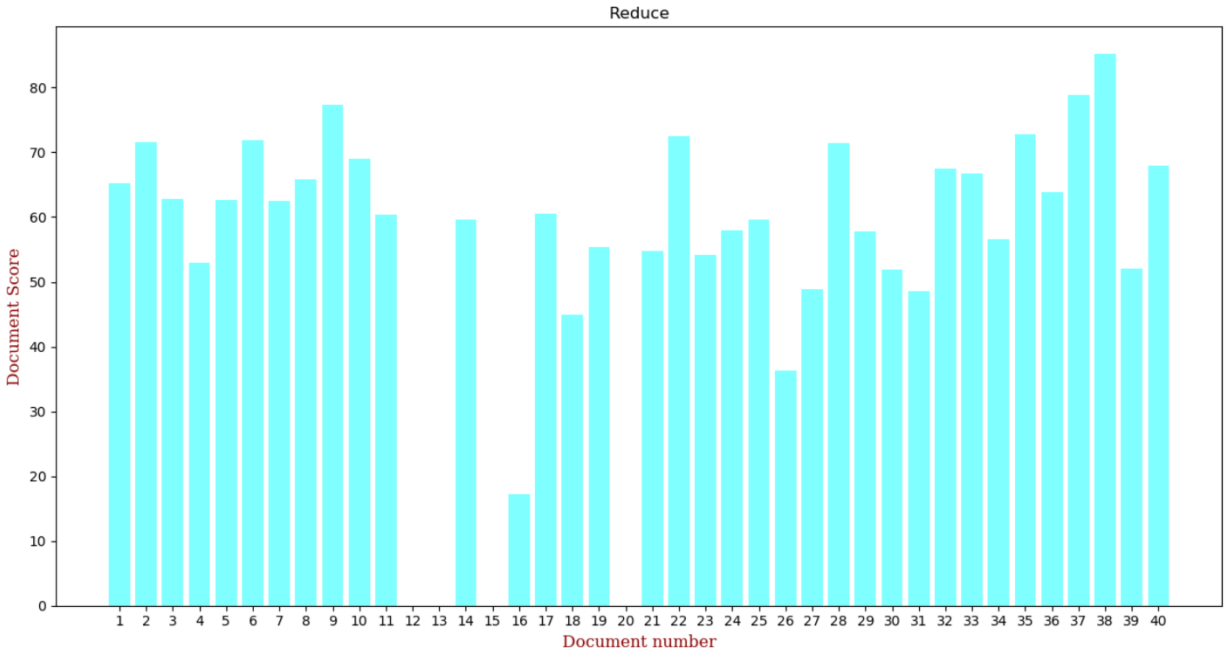
ביצענו את המדידות שתוארו לעיל על האלגוריתמים השונים.

להלן תוצאות ההרצה של ארבעת האלגוריתמים על כל המסמכים שבדקנו:

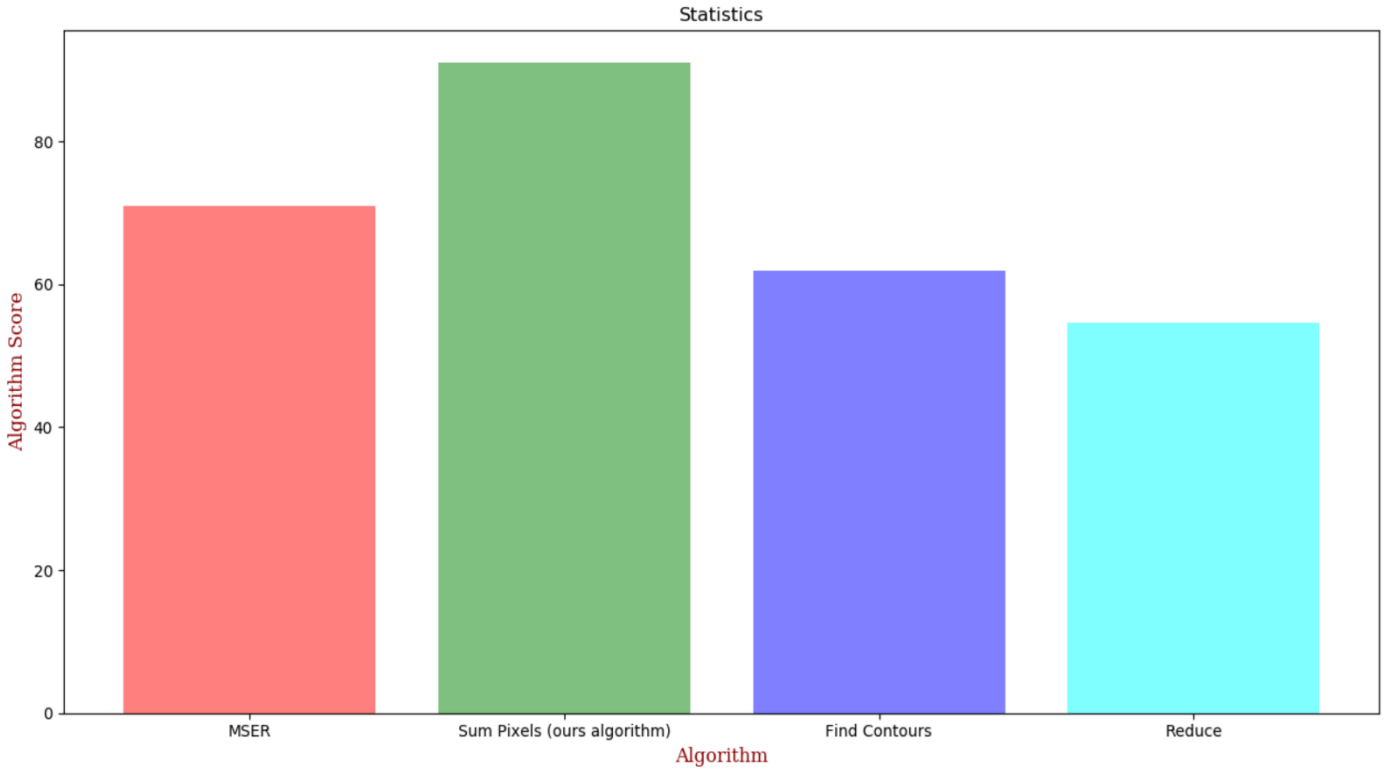





Find Contours
Document Score
Document number


Reduce
Document Score
Document number

מהשוואה בין תוצאות האלגוריתמים הסופיות, קיבלנו את התוצאות הבאות:



ניתן לראות כי ניכר פער בין תוצאות האלגוריתם שלנו לבין תוצאותיהם של שאר האלגוריתמים, וציון האלגוריתם שלנו הוא הגבוה ביותר מבין כולם. לכן הוחלט שנמשיך ונשתמש באלגוריתם זה לצורך גילוי השורות.

---

[1] Text detection in natural images using convolutional neural networks

[2] Separating Lines of Text in Free-Form Handwritten Historical Documents

[3] Line Detection Model and Adaptive Threshold Based Image Segmentation For Handwriting Word Recognition