

**פרויקט מסכם - קורס עיבוד תמונה**

תוכן עניינים

[מבוא 3](#_Toc443736575)

[מטרת הפרוייקט 3](#_Toc443736576)

[הנחות בסיס והגדרות 3](#_Toc443736577)

[חייכן תקין 3](#_Toc443736578)

[אלגוריתם 4](#_Toc443736579)

[כללי 4](#_Toc443736580)

[פירוט עיקרי האלגוריתם 4](#_Toc443736581)

[קלט פלט 7](#_Toc443736582)

[הוראות הפעלה 7](#_Toc443736583)

[סיכום ומסקנות 11](#_Toc443736584)

# מבוא

הרבה פעמים אנחנו נתקלים במהלך הגלישה שלנו באינטרנט, במסמך איכותי שנרצה לעשות בו שימוש כזה או אחר, לדוגמא:

* כאשר אנו קוראים מצגת באנגלית עבור הלימודים, ואנו נתקלים בשקופית לא ברורה כל כך, ע"מ לתרגם אותה עלינו להעתיק למילון מקוון את הטקסט של השקופית. הרבה פעמים השקופית אינה ניתנת לעריכה טקסטואלית שכן מדובר בקובץ תמונה שבתוכו טקסט, וללא הרעיון שלנו – נוכל רק להעתיק ידנית את השקופית למילון.
* כאשר נרצה לשכור דירה חדשה, נקבל חוזה כקובץ PDF שלא ניתן לערוך. בזכות הרעיון שלנו נוכל לצלם מסך, להכניסו לקוד וכך לערוך את החוזה לדרישותינו.

לאחר שהגשנו את המטלות בקורס ולמדנו את הנושאים המגוונים, החלטנו לנסות לזהות תמונות טקסט ולכתוב אותן בקובץ הניתן לעריכה. השתמשנו בהפיכת תמונה לבינארית , פקודות ועיצוב GUI כולל פונקצית save ובדיקה מעמיקה אודות יכולות MATLAB והצלחנו לכתוב , בזמן המוגבל שהיה לנו, קוד אופטימלי לרוב סוגי הקלטים העונים על ההנחות שלנו. נתקלנו בקשיים שעל חלקם התגברנו ועל חלקם פחות, אותם נפרט בהמשך.

# מטרת הפרויקט

ליצור קוד שיקבל כקלט תמונת טקסט (עם הנחות שעליהן נפרט בהמשך), ויוציא כפלט קובץ note pad שהמשתמש יבחר היכן לשמור על המחשב שלו, בתוך הקובץ יהיה הטקסט שהיה בתמונה.

מטרתינו היא להגיע לתוצאות מדוייקות בלפחות 70% מהמקרים שננסה.

אמנם נתבקשנו להניח מספר הנחות מקלות, אך החלטנו לאתגר את עצמנו בעיקר במשימה של קריאת קלט שמורכב באותיות בגדלים שונים (גודל אחיד לכל קלט נפרד, אך לא כמה גדלים שונים בקלט אחד).

# הנחות

**קלט תקין:**

1. קובץ תמונה.
2. כתוב בפונט arial.
3. הטקסט יהיה שחור על רקע לבן.
4. **עדיפות** ללא אות i וללא התו '!'

**פלט תקין:**

1. ייתכן ונקבל\_\_\_\_ כאשר בתמונת הקלט יש את האות i או התו !

# האלגוריתם

1. הכנו קובץ jpeg שמכיל בתוכו את התוים שאנו מעוניינים שהקוד שלנו יידע לזהות. התווים בגופן arial וכוללים את מלוא האותיות הלועזיות (ראשיות וקטנות) ומספר סימני פיסוק.
2. האלגוריתם שלנו הופך את התמונה לבינארית ומחלק אותה למספר מטריצות כמספר האותיות בזכות שיטת Bounding Box ושומר אותן במערך cell.

לפני השמירה:

* האלגוריתם הוא דואג לחבר בין 2 צורות שמהוות צורה אחת, כמו למשל האות i שמורכבת מ2 צורות – קו ונקודה. הוא עושה זאת בפונקציית comb שאנחנו הגדרנו.
* עוד הוא משנה את הגודל של האותיות ל 100 על 100, כדי שנוכל לזהות לאחר מכן קלטים בכל גודל שיגיעו (פירוט בהמשך).
* באופן ידני אנחנו כותבים Dictionary שבעזרתו הקוד מבין איזו מטריצה מייצגת איזו אות.

1. מקבלים את תמונת הקלט ומבצעים עליה מניפולציות דומות על מנת להכניס גם אותה לתוך מערך cell ולחלק אותה לצורות בדיוק כמו שעשינו לתמונה הקודמת, כך שכעת יש לנו 2 מערכי cell – באחד יש את כל "בנק" התווים , ובשניה יש את רק את התווים שיש בתמונת הקלט. כאשר אנו עושים את זה אנו לוקחין בחשבון דבר נוסף שלא לקחנו קודם בחשבון, והוא מיקום האות בטקסט:

* השיטה שאנו משתמשים בה לצורך חלוקת האותיות (Bounding Box) קוראת את האותיות מלמעלה למטה, וזה משבש לנו את התוכניות, שכן עלינו לקרוא משמאל לימין. כאן נתקלנו באתגר לא פשוט שבסיומו הצלחנו להתמודד עם הבעיה באמצעות מיון : לפני שאנחנו מכניסים לתוך הטבלה את האותיות, אנו בודקים אם היא נמצאת בשורה הראשונה. אם כן – מכניסים אותה, אם לא – שומרים אותה להמשך האיטרציות. כך נגיע למצב שנכניס לתוך הטבלה את האותיות בדיוק לפי הסדר התקין לקריאה.

4 . לבסוף , אנו מגיעים לפונקציית ההשוואה. שם האלגוריתם קורא את אות אחר אות שיש במערך שנוצר מטבלת הקלט, ומשווה כל אות למטריצות שיש לנו בבנק. הוא עובר אינדקס אחר אינדקס של המטריצות וכותב לבסוף את המטריצה הזהה ביותר שמצא.

# שיטות מיוחדות

1. Bounding-Box
2. Label

# קשיים ואתגרים

לאחר קריאה מעמיקה חוזרת של כלל החומרים הרלוונטים שלמדנו בקורס, ומחקר באינטרנט, הבנו שאנחנו עומדים להתמודד עם מספר קשיים מרכזיים, והם:

1. בשל ההחלטה לא להשתמש בספריות קיימות (שכן אז לא היה לנו פרויקט) היינו צריכים "ללמד" את הקוד שלנו איך שנראית כל אות. היה עלינו לעבור אות-אות, לחבר אותיות בעיתיות (כאלו שבנויות משתי צורות, כדוגמת j), לזהות את הסדר הכרונולוגי בזיהוי האותיות של המחשב, ולכתוב ידנית כל צורה מהי.
2. קריאת קלט עם מספר שורות באופן תקין - שיטת ה Bounding-Box קוראת את הצורות בתמונה מלמעלה למטה, כלומר שאם קיים קלט עם 2 שורות, אזי השיטה לא תנתח את האותיות לפי הסדר הנכון (משמאל לימין), היינו צריכים למיין את הקלט לפי מיקום הצורות (=האותיות) וכך לכתוב בצורה תקינה את הפלט.
3. רווח –
4. קלט בגדלים שונים – מכיוון שהתמונה הראשונה שהכנסנו לקוד שלנו (עליה פורט בסעיף 1) מגיעה בגודל ספיציפי, ואחד האתגרים שלנו הוא שהקלט יוכל להיות במספר גדלים שונים, היינו צריכים ליצור צורה אחידה לכל אות. לכן החלטנו שכל אות מה"בנק אותיות" שלנו תשתנה לגודל 100/100 , וכן כל אות בקלט – וכך הצלחנו להשוות בצורה זהה ברוב המוחלט של המקרים בין האותיות.
   * שינוי הגודל גורם לתמונה הבינארית להיטשטש, מוסיף לה רעשים ונוספו לה עוד מספרים למטריצת התמונה, תיקנו זאת באופן ידני ובאמצעות השיטה medfilt2 לתיקון רעשים .

# קשיים שלא הצלחנו להתמודד איתם:

1. קריאה תקינה של האות i עדיין לא קוראת באופן מלא. ניסינו להשתמש בקריאת צורות באמצעות שיטת extractFeatures, אך ללא הועיל.
2. ניסינו לקרוא "." (נקודה) ובהתאם להחליט מה לכתוב לפי האות הבאה שנקרא (ואם זה קו אז i) אך גם ללא הועיל, בשל דמיון רב בין הקו של i לקו של l ולקו של !