כללי: ראשית בדקנו אלו פרמטרים הם האופטימליים לפונקציית canny edge detection. לטובת בדיקה זו כתבנו קוד בקובץ main.py שבו יצרנו שני track bars, אחד לכל פרמטר.

הוראות:

* יש לוודא כי החבילות cv2, numpy מותקנות על pycharm (או כל פלטפורמה אחרת)
* בפונקציית load\_img() להחליף את שם התמונה לתמונה אותה נרצה להטעין (מתוך תיקיית jpg).
* להריץ את הקוד.
* להחליף את הברים לmax=191, min=1.
* קבענו את פרמטרים אלו לאחר שהרצנו את הקוד על 10 תמונות, ובדקנו אלו פרמטרים נותנים את התמונה בעלת קווי המתאר הכי ברורים.
* לאחר בחירה זו יש "להפעיל" את הבר האחרון ל-1.

הסבר: הקוד טוען תמונה, הופכת אותה לגווני אפור ומעבד אותה (מטשטש רעשים) ע"י פונקציית Gaussianblur. לאחר מכן יש לבחור את הפרמטרים הנ"ל שמוגדרים בתוך פונקציית canny edge detection ומבצעים עיבוד נוסף לתמונה.

לבסוף הקוד מריץ את אלגוריתם האף לאיתור מעגלים על התמונה

לאחר מכן אפשר לעבור לקובץ RealAlgo:

הוראות:

* יש לוודא כי החבילות cv2, numpy,os,openpyxl,glob מותקנות על pycharm (או כל פלטפורמה אחרת)
* יש לדאוג כי תיקיות masks,npy,jpg נמצאות כולן בתיקייה של הקוד.
* יש לדאוג שגם קובץ האקסל מוגדר באותו מקום.
* במידה ולא, יש לשנות את המיקומים:

הגדרת מיקום תיקיית jpg- בשורה שמתחילה בלולאת for filepath in global. להחליף במקום "jpg".

הגדרת מיקום תיקיית masks- לתוך משתנה mask. להחליף את המיקום במקום "masks".

הגדרת מיקום תיקיית npy- לתוך משתנה number. להחליף את המיקום במקום "npy".

הגדרת מיקום קובץ אקסל- לתוך משתנה workbook. להחליף את המיקום במקום "results.xlsx".

* לבסוף אפשר להריץ את הקוד

הסבר: הקוד קורא כל תמונה בנפרד, ובהתאם את קובץ הnpy ואת המסיכה המתאימה לה.

כשלב ראשוני זיהינו את מספר האשכולות בכל תמונה וזאת בעזרת קובץ npy ותמונות המסכות שהתקבלו מהפרויקט הקודם. באמצעות זיהוי זה התמקדנו בכל אשכול בנפרד ע"י זיהוי הגוון הייחודי של כל מסכה, ולבצע מניפולציה על התמונה המקורית על מנת לייצג כל אשכול כתמונה נפרדת כך ששטחו של האשכול היה בגווני RGB המקוריים ושאר התמונה הומרה לשחור.

לאחר מכן זיהינו את קווי המתאר של המסכה ע"י חיפוש הצבע הייחודי שלה בעזרת קובץ npy של התמונה. זיהוי קווי המתאר הקל על הגעה לנתוני קואורדינטות המסכה וזיהוי השטח שלה בתוך התמונה, וזאת בכדי לזהות את גודלו של האשכול.

בשלב הבא הכנו את התמונה על מנת שתתאים כקלט לאלגוריתם וזאת באמצעות הפיכתה לגווני אפור, החלקת רעשים ע"י פונקציית Gaussian Blur והמרתה לבינארית (שחור-לבן) ע"י פונקציית canny edge detection.

לטובת זיהוי הענבים, השתמשנו בפונקציה קיימת בספריית openCV ב-Python המשמשת לזיהוי עיגולים בתמונות ע"י אלגוריתם האף.

התוצאות נכנסות לתוך קובץ אקסל לפי מספר האשכול והתמונה (המספור הוגדר בקובץ npy).