**פרויקט סיום בעיבוד תמונה – ניתוח תמונות של ממברנה**

**שם המגיש: טל גורודצקי**

**רקע:**

בפרויקט זה התבקשנו לבנות 2 פונקציות, כאשר הפונקציה הראשונה מקבלת נתיב לתמונה של ממברנה ומחזירה רשימה של מילונים, כאשר כל מילון מידע על עיגול צבעוני (כחול, אדום, צהוב ושחור) שאותר בתמונה, המידע כולל את מרכז המעגל, רדיוס וצבע.

בפונקציה שנייה, נדרשנו לקבל רשימה של לפחות 10 נתיבים לתמונות, כאשר כל תמונה מהווה חלק מתצלום רצף של הממברנה. במטלה זו התבקשנו לנתח את השינוי של כל עיגול בכל תמונה, תוך כדי מתן זיהוי לכל עיגול ומעקב אחריו.

במסמך שלהלן אפרט כיצד ביצעתי את המשימות שאוזכרו לעיל, ואדגים תוצאות.

**חלק ראשון: זיהוי עיגולים בתמונה בודדת**

לשם ביצוע משימה זו, בניתי פונקציה בשם detect\_colored\_circles אשר מקבלת נתיב לתמונה, ומחזירה רשימה של מילונים, כאשר כל מילון מכיל מידע על עיגול שאותר בתמונה – צבעו, מרכז המעגל (x,y) ורדיוסו.

תחילה, טענתי את התמונה באמצעות cv2.imread אשר מקבלת נתיב לקובץ. לאחר מכן, הבנתי שעליי לבצע ניקוי לתמונה, שכן בחלק מהרצפים היה מעין רקע כתום אשר שיבש את ערכי הצבע בhsv. על מנת לנקות את התמונה, בניתי 3 פונקציות עזר והשתמשתי בהן בסדר המפורט להלן:

* Gray world white balance – פונקציה אשר עושה ממוצע משוקלל על כל ערוצי הrgb בתמונה ומאזנת את הרעשים בה, ע"פ פרמטר אלפא מסוים (בחרתי 0.7 משום שזה תיקון כמעט מלא לתמונה)
* Denoise image – פונקציה אשר מנקה רעשים על ידי טשטוש מסוג ביליטרל.
* Enhance contrast – פונקציה אשר עושה היסטוגרמה (clahe) על התמונה, כאשר היא במרחב הצבעים lab, לאחר ביצוע ההיסטוגרמה היא מחזירה את התמונה למרחב ה-rgb

לאחר ניקוי ואיזון התמונה מרעשי הרקע, העברתי את התמונה ממרחב הצבעים rgb למרחב hsv, משום שלפי התרגיל ניכר היה שיש כאן התעסקות בעכירות של צבעים (saturation, hue).

בשלב זה, הגדרתי רשימה של צבעים לפי התמונות שקיבלתי (התאמתי את הערכים ע"י פונקציה שמקבלת מרכז מעגל ורדיוס, ומחזירה מהם טווח הערכים והממוצע של הhsv בתחום זה), ועבור כל אחד מהצבעים ברשימה יצרתי מסכה.

לאחר יצירת המסכה לכל צבע, ביצעתי פעולות מורפולוגיות על מנת לייצר אליפסה בצורה של המסכה, כדי לטייב את המסכה.

לאחר מכן, השתמשתי בפונקציה cv2.findCounturs על המסכה, וזאת כדי למצוא את הקונטורים המתאימים עבור המסכה שנוצרה. כמו כן, עברתי בלולאה על כל הקונטורים שאותרו במסכה, ובעזרת הפונקציה cv2.minEnclosingCircles חילצתי מכל קונטור את העיגול המינימלי המתאים לכל קונטור (בעצם, כבר בשלב הזה אותרו במסכה [תמונה] מספר קונטורים [עיגולים] ולכל אחד מהם חולץ מרכז המעגל והרדיוס).

**בחלק זה ישנה הנחה – אם רדיוס העיגול קטן מ-35 וגדול מ100, ניתן להתעלם ממנו שכן לא רלוונטי (הנחה זו העיפה לי עיגולים קטנים)**

עבור כל עיגול שחולץ מהקונטור, הכנסתי את ערכיו ואת ערכי הצבע שלו (לפי המסכה) במילון, ואת נתוני העיגול הזה הכנסתי לרשימת העיגולים.

לבסוף, ביצעתי מעבר על הרשימה וזאת כדי לסנן עיגולים באופן הבא – אם המרחק בין 2 עיגולים קטן מהרדיוס של אחד מהם (כלומר אחד העיגולים מוכל בשני) – מדובר באותו העיגול, ולכן נכניס רק אחד מהם לרשימת העיגולים המסוננים.

לבסוף, אני מחזיר את רשימת העיגולים המסוננים.

**(הערה: יש בקובץ ההגשה עוד פונקציה, בשם count and mark circle. פונקציה זו מבצעת את אותו הדבר שתיארתי לעיל, רק שהיא משתמשת בhough circle . יש לציין שהפונקציה עובדת ומזהה את (רוב) העיגולים הרלוונטים, אך לא באופן מדויק (רדיוס גדולים מדי, מעגלים סוטים לצדדים לפעמים וכו'). התחלתי את העבודה בדרך זו, אך עברתי לדרך שצוינה לעיל כי הבחנתי בכך שזיהוי מסכות וקונטורים עובד טוב יותר בתרגיל זה.)**

**חלק שני: מעקב אחר שינויים לאורך זמן**

על מנת לבצע את המעקב אחר רצף תמונות, בניתי פונקציה בשם track\_circles over\_time אשר מקבלת רשימה של נתיבים לתמונות ומחזירה טבלה (מילון) שמציין מהם הערכים של כל עיגול וזיהוי שלו בכל תמונה מ10 התמונות ברצף.

תחילה, הגדרתי רשימה ריקה בשם table ומונה אשר מציין את מס' הזיהוי של כל עיגול (מאותחל ל-1).

כעת, עברתי בלולאה על כל אחת מנתיבי התמונות שמופיעות ברשימה, ושלחתי כל נתיב לפונקציה detect\_colored\_circles (הפונקציה של החלק הראשון), שתפקידה לזהות עיגולים בכל תמונה.

בשלב זה, עברתי על כל אחד מהעיגולים שהפונקציה החזירה, הוספתי לו מזהה לפי המונה שהוגדר בהתחלה ובדקתי האם יש לו התאמה עם אחד מהעיגולים שהופיעו בתמונות הקודמות, כאשר התאמה מוגדרת להיות המרחק הקטן ביותר בין 2 עיגולים (אם העיגול הופיע בתמונה הקודמת, ובתמונה הנוכחית הוא גם מופיע, לפי נוסחת דיסטנס המרחק ביניהם אמור להיות זעום ואף קטן מהרדיוס של אותו עיגול, ולכן זו התאמה מיטבית – כלומר זהו אותו העיגול מהתמונה הקודמת).

אם לא נמצאה ההתאמה הטובה ביותר עבור עיגול, כנראה שהוא לא הופיע בתמונה הקודמת/לא זוהה בתמונה הנוכחית (לא אמור לקרות ברצפים שלי!), ולכן נגדיר עבור עיגול זה מס' זיהוי חדש, בהתאם לערך המונה הנוכחי.

לבסוף, נכניס את העיגול (מילון אשר מעודכן כעת עם מספר זיהוי) אל הרשימה table, לצורך תיעוד המעקב ובנוסף נגדיר את רשימת העיגולים הקודמת להיות רשימת העיגולים הנוכחית (על מנת שבאיטרציה הבאה, העיגולים שיאותרו יושוו על פי המרחק עם העיגולים הנוכחים).

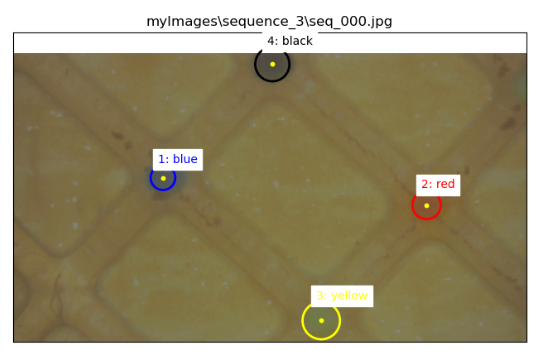
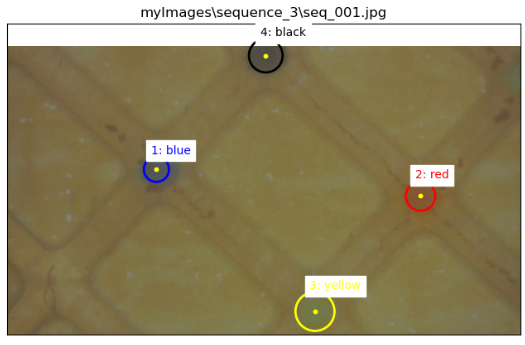
נשים לב שעלינו להחזיר מילון בפונקציה זו, אך יש ברשותנו כעת את הרשימה table. לכן, נחזיר את המילון באופן הבא:

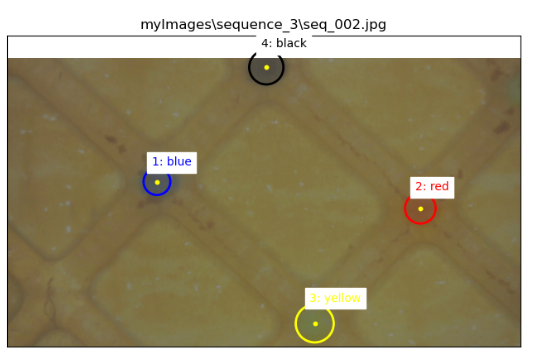
{"table" : table}

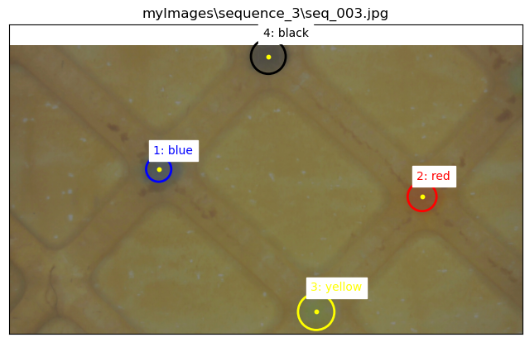
**חלק שלישי: הצגת תוצאות**

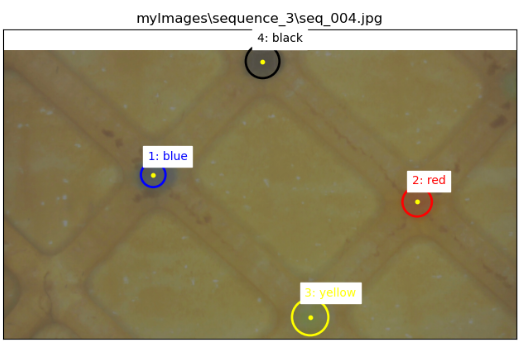
מכיוון שפרויקט זה דורש עיסוק רב בתמונות ובצבעים, עלה הצורך ליצור פונקציה אשר תציג לי את העיגולים שמצאתי בצורה ויזואלית. לשם כך, בניתי פונקציה בשם show\_results אשר מקבלת נתיב לתמונה ורשימה של עיגולים, ומציגה אותם על המסך באמצעות plt. הפונקציה מקיפה כל אחד מהעיגולים שאותרו במעגל עם צבע, לפי הצבע שאותר ברשימה circle. לנוחות הבודק, יצרתי main קצר אשר מדגים את הרצת הפונקציה הראשונה ואת הפונקציה השנייה, ומדגים תוצאות ויזואליות על המסך, כמו כן אצרף כאן מספר דוגמאות:

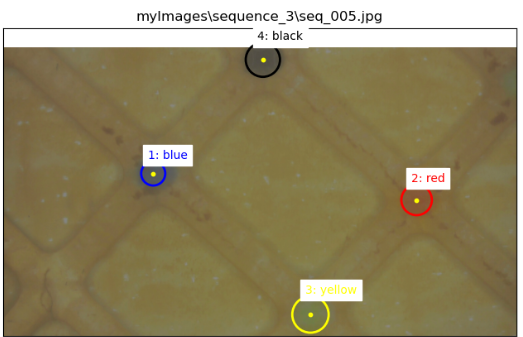
* הרצת הפונקציה השנייה על רצף 3 (רצף בעל לכלוך כתום – עכירות מים גבוהה):

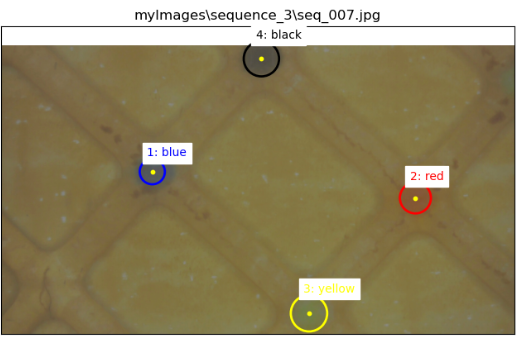




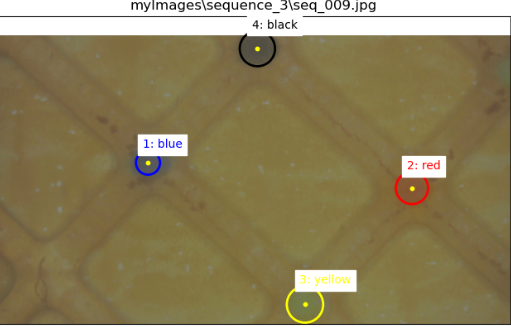






תמונה שמכילה צילום מסך

התיאור נוצר באופן אוטומטי



תמונה שמכילה צילום מסך, תכונות מולטימדיה

התיאור נוצר באופן אוטומטיכמו כן, פלט הפונקציה (מילון):

{'image': 1, 'circle\_id': 1, 'x': 560, 'y': 465, 'radius': 46, 'color': 'blue'}

{'image': 1, 'circle\_id': 2, 'x': 1546, 'y': 567, 'radius': 53, 'color': 'red'}

{'image': 1, 'circle\_id': 3, 'x': 1152, 'y': 1000, 'radius': 70, 'color': 'yellow'}

{'image': 1, 'circle\_id': 4, 'x': 969, 'y': 40, 'radius': 64, 'color': 'black'}

{'image': 2, 'circle\_id': 1, 'x': 558, 'y': 462, 'radius': 47, 'color': 'blue'}

{'image': 2, 'circle\_id': 2, 'x': 1546, 'y': 562, 'radius': 55, 'color': 'red'}

{'image': 2, 'circle\_id': 3, 'x': 1151, 'y': 994, 'radius': 73, 'color': 'yellow'}

{'image': 2, 'circle\_id': 4, 'x': 967, 'y': 36, 'radius': 63, 'color': 'black'}

{'image': 3, 'circle\_id': 1, 'x': 560, 'y': 461, 'radius': 50, 'color': 'blue'}

{'image': 3, 'circle\_id': 2, 'x': 1545, 'y': 561, 'radius': 57, 'color': 'red'}

{'image': 3, 'circle\_id': 3, 'x': 1150, 'y': 992, 'radius': 71, 'color': 'yellow'}

{'image': 3, 'circle\_id': 4, 'x': 969, 'y': 33, 'radius': 64, 'color': 'black'}

{'image': 4, 'circle\_id': 1, 'x': 559, 'y': 463, 'radius': 47, 'color': 'blue'}

{'image': 4, 'circle\_id': 2, 'x': 1544, 'y': 566, 'radius': 54, 'color': 'red'}

{'image': 4, 'circle\_id': 3, 'x': 1148, 'y': 997, 'radius': 68, 'color': 'yellow'}

{'image': 4, 'circle\_id': 4, 'x': 969, 'y': 41, 'radius': 65, 'color': 'black'}

{'image': 5, 'circle\_id': 1, 'x': 561, 'y': 467, 'radius': 46, 'color': 'blue'}

{'image': 5, 'circle\_id': 2, 'x': 1548, 'y': 568, 'radius': 55, 'color': 'red'}

{'image': 5, 'circle\_id': 3, 'x': 1148, 'y': 1000, 'radius': 68, 'color': 'yellow'}

{'image': 5, 'circle\_id': 4, 'x': 970, 'y': 42, 'radius': 63, 'color': 'black'}

{'image': 6, 'circle\_id': 1, 'x': 561, 'y': 470, 'radius': 45, 'color': 'blue'}

{'image': 6, 'circle\_id': 2, 'x': 1546, 'y': 569, 'radius': 57, 'color': 'red'}

{'image': 6, 'circle\_id': 3, 'x': 1149, 'y': 998, 'radius': 68, 'color': 'yellow'}

{'image': 6, 'circle\_id': 4, 'x': 971, 'y': 44, 'radius': 64, 'color': 'black'}

{'image': 7, 'circle\_id': 1, 'x': 565, 'y': 472, 'radius': 48, 'color': 'blue'}

{'image': 7, 'circle\_id': 2, 'x': 1549, 'y': 570, 'radius': 58, 'color': 'red'}

{'image': 7, 'circle\_id': 3, 'x': 1152, 'y': 1002, 'radius': 68, 'color': 'yellow'}

{'image': 7, 'circle\_id': 4, 'x': 973, 'y': 43, 'radius': 65, 'color': 'black'}

{'image': 8, 'circle\_id': 1, 'x': 565, 'y': 473, 'radius': 47, 'color': 'blue'}

{'image': 8, 'circle\_id': 2, 'x': 1549, 'y': 571, 'radius': 58, 'color': 'red'}

{'image': 8, 'circle\_id': 3, 'x': 1151, 'y': 1003, 'radius': 68, 'color': 'yellow'}

{'image': 8, 'circle\_id': 4, 'x': 973, 'y': 50, 'radius': 66, 'color': 'black'}

{'image': 9, 'circle\_id': 1, 'x': 562, 'y': 471, 'radius': 45, 'color': 'blue'}

{'image': 9, 'circle\_id': 2, 'x': 1548, 'y': 571, 'radius': 57, 'color': 'red'}

{'image': 9, 'circle\_id': 3, 'x': 1152, 'y': 1004, 'radius': 69, 'color': 'yellow'}

{'image': 9, 'circle\_id': 4, 'x': 971, 'y': 47, 'radius': 65, 'color': 'black'}

{'image': 10, 'circle\_id': 1, 'x': 564, 'y': 476, 'radius': 45, 'color': 'blue'}

{'image': 10, 'circle\_id': 2, 'x': 1551, 'y': 571, 'radius': 60, 'color': 'red'}

{'image': 10, 'circle\_id': 3, 'x': 1151, 'y': 1007, 'radius': 68, 'color': 'yellow'}

{'image': 10, 'circle\_id': 4, 'x': 973, 'y': 49, 'radius': 66, 'color': 'black'}