Find-Forward

שם סטודנט: דבורי מונטג

ת.ז: 325277283

שם מכללה: "תפארת החיים"

מורה מנחה: רחל דרבקין

תאריך הגשה: 19/06/23

תוכן עניינים

1.....

1. הצעת פרויקט 4

2. תקציר / מבוא 6

2.1 הרקע לפרויקט.....6

2.2 תהליך המחקר.....7

2.3 סקירת ספרות.....12

2.4 אתגרים מרכזיים 13

2.4.1 הבעיה איתה התמודדתי.....13

2.4.2 הסיבות לבחירת הנושא.....13

2.4.3מוטיבציה לעבודה 14

2.4.4 הצורך עליו עונה הפרויקט.....14

2.4.5 הצגת פתרונות לבעיה 14

3 מטרות ויעדים 15

4 אתגרים.....15

5 מדדי הצלחה למערכת.....17

6 רקע תאורתי / ספרות מקצועית 17

7 תיאור מצב קיים 18

8 ניתוח חלופות מערכתי 18

9 תיאור החלופה הנבחרת 19

10 אפיון המערכת 20

10.1 ניתוח דרישות המערכת 20

10.2 מודול המערכת – תהליכים מרכזיים.....20

10.3 אפיון פונקציונלי – פונקציות עיקריות 21

10.4 ביצועים עיקריים.....21

10.5 אילוצים.....22

11 תיאור הארכיטקטורה 22

11.1 הארכיטקטורה של הפתרון המוצע בפורמט שלTop-Down level Design : 22

11.2 תיאור הרכיבים בפתרון.....22

11.3 תיאור פרוטוקולי התקשורת.....23

11.4 שרת – לקוח 23

12 ניתוח ותרשים UML / Use cases של המערכת המוצעת.....23

12.1 תיאור ה-UCהעיקריים של המערכת.....24
12.2 הצגת מקרה (use case) עבור כל הפונקציות העיקריות בפרויקט. 25
12.3 מבנה נתונים בהם השתמשתי.....26
12.4 חישוב יעילות האלגוריתם.....26
12.5 תרשיMUL.....27
12.6 תיאור המחלקות המוצעות.....27
13 רכיבי ממשק 28
14 תיכון המערכת 28
14.1 ארכיטקטורת המערכת 29
14.2 תיכון מפורט 29
15. תיאור התוכנה.....29
15.1 סביבת עבודה.....29
15.2 שפות תכנות.....29
16 תיאור מסכים: 30
17תרשים מסכים המתאר את היררכיית המסכים והמעברים ביניהם.....30
18 מה תפקידו של כל מסך / חלון עם צילום מסך של החלון הרלוונטי 31
19 תיאור מסך הפתיחה 34
20 הודעות למשתמש.....34
21 ממשק משתמש.....34
22 קוד התוכנית-על פי סטנדרטים בליווי תיעוד.....35
פונקציות הוספת קטגוריה חדשה לעץ אינדקס וכן חיפוש קטגוריה בעץ. 38
23 מדריך למשתמש 50
24 בדיקות והערכה.....51
25 מסקנות.....51
26 פיתוחים עתידיים.....52
27 בבליוגרפיה 52

1. הצעת פרויקט

סמל מוסד: 695064

שם מכללה :"תפארת החיים"

שם הסטודנט: דבורי מונטג

ת.ז הסטודנט :325277283

שם הפרויקט : Find-Forward

תיאור הפרויקט: חיפוש יעיל במצלמות אבטחה

הגדרת הבעיה האלגוריתמית:

שמירת הנתונים במבנה נתונים מתאים

מעבר יעיל על עץ ההסרטות

החלטה מהו אובייקט ובאיזה מבנה ייוצג

החלטה מהם גבולות התזוזה.

מעבר יעיל והשוואה מהירה ונכונה בין הפריימים.

שמירת התוצאות במבנה נתונים מתאים

רקע תיאורטי בתחום הפרויקט:

כיום כמעט בכל מקום מצויה מצלמת אבטחה המתעדת את כל המתרחש במקום.

במקרים רבים עולה הצורך לבדוק אירועים שזמן התרחשותם אינו ידוע במדויק.

חשבתי על זה ופתאום עלה לי רעיון איך ליעל את החיפוש במצלמה

לאפשר שליפת מידע לפי אובייקטים שונים בתמונה

נגדיר בדיוק את מה אנחנו רוצים לראות ע"י תמונה/ הגדרת החפץ כמו צבע, צורה, גודל וכו' , ונקבל וידאו קצר

תהליכים עיקריים בפרויקט:

בחירת תיקייה לחיפוש, סוג מצלמה, מספר מצלמה, וטווח זמן.

בחירת סוג החיפוש, הגדרת האובייקט המבוקש,

מעבר על עץ ההסרטות

פירוק כל הסרטה לפריימים

ניתוח סדרת תמונות והתזוזות ביניהם למציאת האובייקט הרצוי.

תיאור הטכנולוגיה : local system application

צד שרת :

שפת תכנות בצד השרת : python

צד לקוח :

שפת תכנות בצד הלקוח: python

מסד נתונים : File system

פרוטוקולי תקשורת : file read/write

לוחות זמנים :

1. חקר המצב הקיים – ספטמבר, אוקטובר
2. הגדרת הדרישות – נובמבר
3. אפיון המערכת – דצמבר
4. אפיון בסיס הנתונים – דצמבר
5. עיצוב המערכת – ינואר, פברואר
6. בנית התוכנה – פברואר, מרץ
7. בדיקות – אפריל
8. הכנת תיק פרויקט – מאי
9. הטמעת המערכת – יוני
10. הגשת פרויקט סופי - יולי

חתימת הסטודנט : פ.מוזלג

חתימת רכז המגמה :

אישור משרד החינוך :

2. תקציר / מבוא

2.1 הרקע לפרויקט

בבואי לבחור במה הפרויקט יעסוק, המטרה שעמדה לנגד עיני הייתה בחירת תחום מאתגר, חכם וצובר תאוצה, כזה שיקנה לי ניסיון חדש, וקשת מגוונת של אפשרויות.

בעשור האחרון, תחום מצלמות האבטחה עשה קפיצה טכנולוגית משמעותית. כיום מותקנות מצלמות אבטחה במגוון מקומות כמו בתי מגורים, עסקים, חניונים, אולמות אירועים, מוסדות ציבור ועוד, ובמקרה הצורך בקלות אפשר לצפות בהקלטות מהמצלמה.

אך עם זאת עדיין קיימים מוקשים בתחום וכל מעקב הוא תהליך מורכב ומייגע. בשל כך: תיקים רבים מתעכבים במשטרה עקב אי מציאת גנבים, מנהלי מפעלים נתקלים בקשיים לאיתור פועלים מתרשלים וכו' .

כל אלה חשפו את הצורך הגובר לאיתור מהיר ויעיל של אירועים חשובים וזיהוי מוקד של אובייקטים מסוימים בהקלטות.

ומכאן התעוררה בי תשוקה לחקור ולפתח פתרונות לחיפוש יעיל של אובייקטים בתוך ההקלטות הרבות שנרשמות.

חקרתי את הידע החשוב ביותר בנושא ולאחר מחשבה החלטתי לפתח תוכנה לחיפוש במצלמת אבטחה לפי אובייקט.

בעזרת החיפוש לפי אובייקט, ניתן לזהות אירועים חשובים ופרטיים מתוך כמויות גדולות של מידע.

לדוגמה, אם תרצה לחפש אדם מסוים שנכנס למבנה אתה תוכל להשתמש בחיפוש לפי אובייקט ולקבל את כל התמונות או הווידאו המתאימים לאדם המבוקש. זה מאפשר חסכון בזמן ומאמץ במהלך תהליך החיפוש.

במקום לבדוק ולחפש בצורה ידנית בכל ההקלטות והתמונות שנרשמו על ידי מצלמות האבטחה, המערכת תבצע את החיפוש באופן אוטומטי וממוחשב בשיתוף מינימלי של המשתמש תוך שימוש באלגוריתמים מתקדמים.

אני מקווה שאכן הפרויקט יספק מידע חשוב ומועיל לתחום האבטחה ויתרום להתקדמות טכנולוגיה נוספת בתחום.

חשוב לי לציין שבפיתוח התוכנה התבססתי על כך שמדובר במצלמות אבטחה פשטניות,

מצלמות במקומות פשוטים כמו חנויות/משרדים, בתי מגורים וכדו'.

2.2 תהליך המחקר

תהליך פיתוח הפרויקט הוביל אותי לניסוי ומחקר של מספר שיטות לזיהוי אובייקטים כשהתחלתי, התרשמתי מהאתגר העצום שמולי. לחפש ולזהות אובייקטים במאות אלפי תמונות והקלטות הייתה משימה שקשה להשתלב בתהליך אינטואיטיבי.

תחילה ניסיתי אלגוריתמים של עיבוד תמונה, אלגוריתם עיבוד תמונה מתבסס על מניפולציות ויזואליות שונות בתמונות ,דגימות צבע, חישובים בתחום האלגברה הלינארית על מטריצות של פיקסלים ועוד .אלגוריתמים אלו אומנם נותנים תוצאה טובה, אך לא אופטימלית. הדרך לכתיבתם משתמשת בהרבה ניסוי וטעיה, חקירה והתעסקות בפרטים שוליים וטכניים. דרך זו מסורבלת ובלתי נוחה לפיתוח.

המשכתי לחקור וגיליתי שקיימים מודולים הבנויים בצורה של למידת מכונה. מודולים אלה הם מבני תוכנה או רשתות עמוקות שמיועדים ללמד מערכות ממוחשבות ללמוד ולתפקד באופן עצמאי מתוך נתונים. (בהמשך הסביר מהי רשת עמוקה)

הראשון שבדקתי היה זיהוי אובייקט באמצעות חלון הזזה. תהליך חלון הזזה מתחיל על ידי עבירת חלון ריבועי בגודל קבוע על פני התמונה. בכל מיקום של החלון, מתבצע זיהוי של האובייקט באמצעות המודל. המודל מקבל את החלון ומשתמש בתהליך למידת מכונה כדי ללמוד דפוסים ומאפיינים של האובייקטים הרצויים.

אולם, טכניקה זו יכולה להיות מאוד מאכזבת, במיוחד כאשר ישנם אובייקטים במגוון גדלים, גדלי החלון גדולים מדי או קטנים ביחס לגודל האובייקטים בתמונה או במקרה של מספר רב של רעשים בתמונה. כך שזה לא סיפק אותי, רציתי להגיע לתוצאות מקסימליות אז עברתי הלאה לאלגוריתמים נוספים, מתוכם:

SSD – (Single Shot Detector) הוא אלגוריתם עמוק לזיהוי אובייקטים בתמונות ובווידאו בזמן אמת. המודל משתמש ב CNN ללמידה עמוקה של האובייקטים בתמונה ומיישם חיזויים על מספר רב של מסגרות בתמונה כדי לזהות את האובייקטים.

Faster R-CNN (Faster Region-based Convolutional Neural Network) - האלגוריתם משלב רשת עמוקה ללמידה והצעה של אזורי תפקוד (Region Proposal) בצורה מהירה ויעילה לזיהוי אובייקטים בתמונות.

Mask R-CNN- זהו עדכון של Faster R-CNN שמאפשר גם את זיהוי האובייקט וגם את סיווגו במסגרות של התמונה, כמו גם תצורת האובייקט.

YOLO (You Only Look Once) – הוא אלגוריתם המשתמש ברשת נוירונים עמוקה ומתקדמת שמפרקת את התמונה לרשתות, מבצעת זיהוי אובייקטים בכל תא ומשתמשת בטכניקות ואלגוריתמים מתקדמים כדי להביא לתוצאות מדויקות ויעילות בזיהוי האובייקטים.

קראתי על השיטות השונות בחנתי יתרונות למול חסרונות כולם התגלו כטובים אבל יותר מכולם אלגוריתם YOLO הרשים אותי מאד, גיליתי את עוצמתו והחלטתי להשתמש דווקא בו.

אלגוריתם YOLO שונה מאלגוריתמים אחרים לזיהוי אובייקטים בכך שהוא מבצע זיהוי בשלב בודד, ולא במספר שלבים. זה הופך אותו למהיר ויעיל מאוד, ומאפשר לו לעבד תמונות בזמן אמת. ל-YOLO יש גם דיוק גבוה והוא יכול לזהות אובייקטים מרובים באותה תמונה.

יישומים:

אתחיל בכמה מצבי שימוש באלגוריתם כדי לתת קצת השראה לפני שאעמיק בפרטים הטכניים. מקרי השימוש הם אינסופיים וניתן להשתמש ב – YOLO בתחומים רבים ושונים כמו:

נהיגה אוטונומית: ניתן להשתמש באלגוריתם YOLO במכוניות אוטונומיות כדי לזהות חפצים סביב מכוניות כגון כלי רכב, אנשים ואותות חניה. זיהוי עצמים במכוניות אוטונומיות נעשה כדי למנוע התנגשות מאחר שאף נהג אנושי לא שולט במכונית.

חיות בר: אלגוריתם זה משמש לאיתור סוגים שונים של בעלי חיים ביערות. סוג זה של זיהוי משמש שומרי חיות בר ועיתונאים לזיהוי בעלי חיים בסרטונים (הן מוקלטים והן בזמן אמת) ובתמונות. חלק מהחיות שניתן לזהות כוללות ג'ירפות, פילים ודובים.

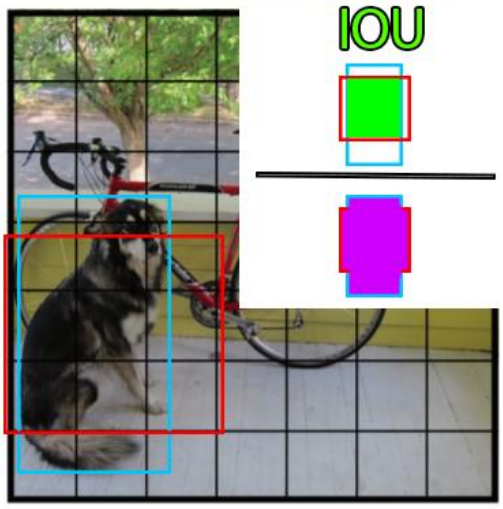
אבטחה: ניתן להשתמש ב-YOLO גם במערכות אבטחה כדי לאכוף אבטחה באזור. נניח שאנשים הוגבלו לעבור באזור מסוים מסיבות ביטחוניות. אם מישהו עובר באזור המוגבל, אלגוריתם YOLO יזהה אותו/ה, מה שידרוש מאנשי האבטחה לנקוט בפעולות נוספות.

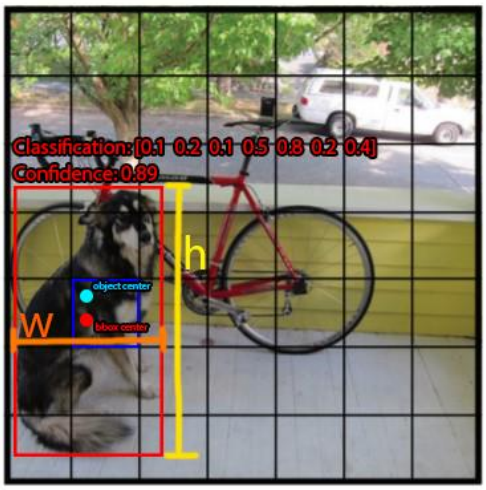
איך YOLO עובד?

YOLO מבוסס על הרעיון של פילוח תמונה לתמונות קטנות יותר. התמונה מפוצלת לרשת מרובעת של מידות SXS כך:

התא שבו שוכן מרכזו של עצם, למשל, מרכז הכלב, הוא התא האחראי על זיהוי האובייקט הזה .כל תא יחזה תיבות תוחמות B וציון ביטחון עבור כל תיבה .ברירת המחדל של ארכיטקטורה זו היא שהמודל יחזה שתי תיבות תוחמות .ציון הסיווג יהיה מ-`0.0` ל-`1.0`, כאשר `0.0` היא רמת הביטחון הנמוכה ביותר ו-`1.0` היא הגבוהה ביותר ;אם לא קיים אובייקט באותו תא, ציוני הביטחון צריכים להיות `0.0`, ואם המודל בטוח לחלוטין בתחזית שלו, הציון צריך להיות `1.0 .`רמות ביטחון אלו לוכדות את הוודאות של המודל שקיים אובייקט באותו תא וכי התיבה התוחמת מדויקת .כל אחת מהתיבות התוחמות הללו מורכבת מ – 5 מספרים: מיקום x , מיקום y , רוחב, גובה וביטחון .הקואורדינטות (x , y) מייצגות את המיקום של מרכז התיבה התוחמת החזויה, והרוחב והגובה הם שברים ביחס לכל גודל התמונה. הביטחון מייצג את ה – IOU בין התיבה התוחמת החזויה לבין התיבה התוחמת בפועל, המכונה תיבת האמת הקרקעית. ה – IOU מייצג Intersection Over Union והוא אזור ההצטלבות של תיבות האמת החזויה והקרקע חלקי שטח האיחוד של אותן תיבות אמת חזויה וקרקעית.

הנה דוגמה ל – IOU: אזור ההצטלבות של אמת הקרקע והתיבה החזויה בירוק חלקי שטח האיחוד של שתי התיבות, בסגול.

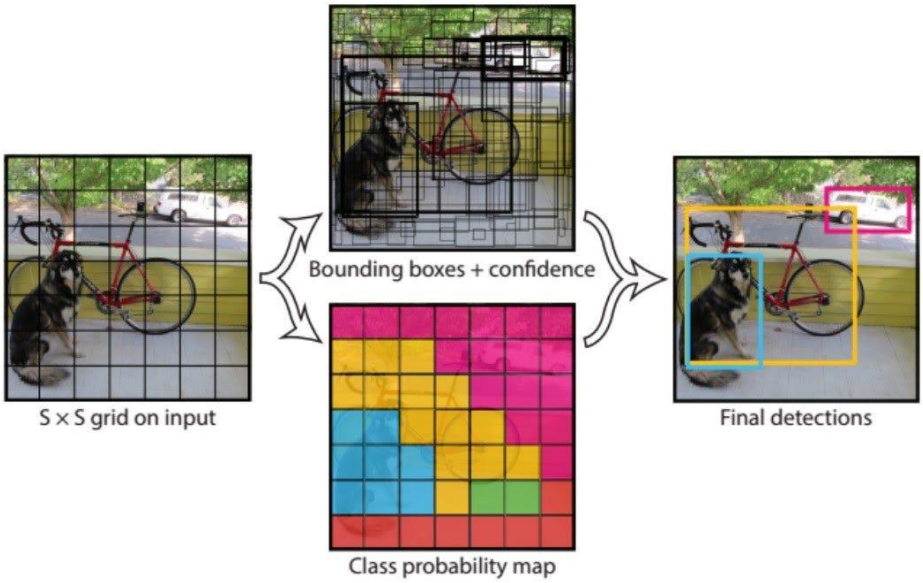




S × S grid on input

התמונה לעיל היא דוגמה לפלט של המודל כאשר רק חיזוי תיבה תוחמת אחת לתא .בתמונה זו, המרכז האמיתי של הכלב מיוצג על ידי עיגול שכותרתו 'מרכז אובייקט' ככזה, תא הרשת שאחראי לזיהוי ותחומה של התיבה הוא זה שמכיל את נקודת הציאן, המודגשת בכחול כהה .התיבה התוחמת שהתא חוזה מורכבת מ-4 אלמנטים .הנקודה האדומה מייצגת את מרכז התיבה התוחמת ***(x, y)*** , והרוחב והגובה מיוצגים על ידי הסמנים הכתומים והצהובים בהתאמה .

להלן תמונה נוספת של כל התיבות התוחמות ותחזיות הכיתה שיבוצעו בפועל והתוצאה הסופית שלהן בדוגמה שנתתי.

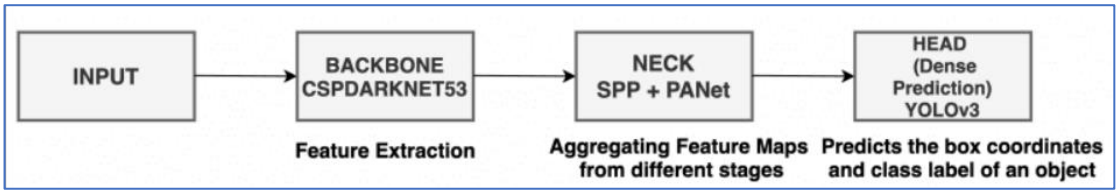


ארכיטקטורת YOLO

דגם YOLO מורכב משלושה מרכיבים מרכזיים: הראש, הצוואר ועמוד השדרה .עמוד השדרה הוא החלק של הרשת העמוקה המורכב משכבות קונבולוציוניות כדי לזהות תכונות מפתח של תמונה ולעבד אותן.

השכבות הקונבולוציוניות משתמשות במסגרת קונבולוציה, שהיא פעולה מתמטית שבה מתבצעת הכפלה של מספרים בין קלט (input) ליחידות ליבה (kernel) במיקום מסוים והוספת התוצאות הכפולות. השכבות הקונבולוציוניות משמשות לזיהוי מאפיינים מקומיים ומודלים את היחידות הליבה כדי לזהות דפוסים ביחידות המידע. כל שכבת קונבולוציה מחשבת את המידע בחלונות קטנים על פני התמונה ומגדירה תכונות מקומיות ברמת הפיקסלים.
שכבות הקונבולוציה מאפשרות לרשתות עמוקות ללמוד אוטומטית את המאפיינים החשובים ביותר בתמונות ולהפעיל ייבוש מרחבי על נתונים גדולים ביעילות. זאת מאפשרת להם לבצע זיהוי והבנה של תכונות מורכבות בתמונות, כגון זיהוי אובייקטים, צורות ומאפיינים מקומיים אחרים

עמוד השדרה מאומן תחילה על מערך נתונים של סיווג, כגון ImageNet ובדרך כלל מאומן ברזולוציה נמוכה יותר ממודל הזיהוי הסופי, מכיוון שזיהוי דורש פרטים עדינים יותר מאשר סיווג .הצוואר משתמש בתכונות משכבות הקונבולציה בעמוד השדרה עם שכבות מחוברות במלואן כדי ליצור תחזיות על הסתברויות וקואורדינטות של תיבה תוחמת .הראש הוא שכבת הפלט הסופית של הרשת אותה ניתן להחליף עם שכבות אחרות בעלות אותה צורת קלט.
זה אומר שאם כרגע אני משתמשת במודל שאומץ לזיהוי אובייקטים, ניתן להחליף את שכבת הפלט הסופית שלו בשכבות אחרות עם צורת קלט דומה (לדוגמה, לתרגם את המודל למודל לזיהוי פנים על ידי החלפת הראש המקורי בראש המאומן לזיהוי פנים)



שלושת החלקים הללו של המודל עובדים יחד כדי לחלץ תחילה מאפיינים ויזואליים מרכזיים מהתמונה ולאחר מכן לסווג ולקשר אותם.

כדי להשתמש ב YOLO עם מערך הנתונים של COCO עליו אומן, הייתי צריכה להשתמש בקבצי התצורה המתאימים ולטעון את קבצי המשקולות המאומנות בעת הפעלת המודל.

קצת אסביר עליהם:

COCO (Common Objects in Context) הוא מאגר נתונים שפותח במיוחד עבור משימות ראיה מתקדמות, ובעיקר לזיהוי אובייקטים והבנה של הקשר הקונטקסטואלי ביניהם. המאגר כולל כ-320,000 תמונות של 80 קטגוריות שונות, כאשר כל תמונה מסופקת עם תיבות תוחמות של האובייקטים הנמצאים בתוכה ועם העלמה של מאפיינים נוספים כמו מסלולים ומספרים עבור האובייקטים. מטרתו העיקרית של COCO היא לתמוך במגוון של משימות ראיה, כולל זיהוי אובייקטים, סיבוב מודלים, סיבוב מונחיות ועוד.

קבצי התצורה (configuration files) של YOLO מכילים את הפרמטרים וההגדרות של המודל. בפרט, הם מגדירים את הארכיטקטורה של הרשת הנוירונים העמוקה, הפרמטרים של הרשת, המערכת היחסית לתהליך ההדרכה ,ההתקנים החומריים שישמשו להרצת המודל ועוד. קובץ התצורה גם מגדיר את התוויות (labels) המשמשות לזיהוי העצמים השונים.

משקולות (weights) הן הערכים הנמצאים בפרמטרים המודל, שנלמדו בתהליך ההדרכה. כאשר תהליך ההדרכה מסתיים, המשקולות נשמרות בקובץ שלמד על ידי YOLO.

2.3 סקירת ספרות

על מנת למצוא חומרים שונים הנצרכים לי בפרויקט נעזרתי רבות במנוע החיפוש של google.

בין האתרים שנמצאו שימושיים עבורי ביותר לכתיבת הפרויקט הן ברקע התאורטי של הנושא והן בנושאים המקצועיים והתכנותיים:

ברקע תיאורטי

https://reolink.com/blog/ip-camera-footage-storage/

https://viso.ai/deep-learning/object-tracking/

https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/

ברקע התכנותי

https://stackoverflow.com/

https://www.w3schools.com/

https://reshetech.co.il/python-tutorials/all-the-tutorials

http://www.github.com

2.4 אתגרים מרכזיים

2.4.1 הבעיה איתה התמודדתי

זיהוי נכון של האובייקטים כדי לאפשר חיפושים מדויקים לפי אובייקט

קידוד תקין של האובייקטים בעץ אינדקס

מנגנון חיפוש מתקדם, אחרת עשוי להיות קשה לבצע חיפושים ספציפיים ומדויקים לפי אובייקט וזה עשוי להתבטא בכך שהמערכת מחזירה תוצאות רבות ולא רלוונטיות עבור החיפוש

פעולות חיפוש מיטביות המביאות שיפור משמעותי ביעילות זמן הריצה

2.4.2 הסיבות לבחירת הנושא

בבואי לבחור נושא לפרויקט הגמר, רציתי לחקור נושא בתחום התכנה אשר יעשיר אותי , ובמקביל יביא לידי ביטוי את הידע והכלים שרכשתי וההשקעה הרבה לאורך כל תקופת הלימודים.

סקרתי ביסודיות נושאים שונים נעצרתי כשהגעתי לתחום הבינה המלאכותית **AI** - Artificial intelligence
תחום שהצית את דמיוני ומצאתי בו התעניינות גדולה והתחלתי לקרוא מאמרים העוסקים בתחום, מתוכם:

בינה מלאכותית היא ענף של מדעי המחשב העוסק ביכולת לתכנת מחשבים לפעול באופן המציג יכולות המאפיינות את הבינה האנושית.

הגדרה רחבה יותר לתחום זה ניתנה על ידי מרווין מינסקי:
"לגרום למכונה להתנהג בדרך שהייתה נחשבת לאינטליגנטית לו אדם התנהג כך."

המבחן המקובל ביותר לבינה מלאכותית הוטבע בשנת 1950 על ידי אלן טיורינג, וידוע בשם "מבחן טיורינג":

מכונה תחשב לתבונית, אם יינתן לאדם היושב בחדר סגור, לנהל שיחה באמצעות ממשק מחשב (Console) עם שתי ישויות שנמצאות בחדר השני, כאשר אחת מהן תהיה מכונה והשנייה אנושית, והמשוחח לא יוכל לזהות מי משתי הישויות היא מכונה או אדם, או ־ אז המכונה תחשב לתבונית.

במהלך החשיבה סביב הנושא, רציתי להתמקד בנושא שאף יביא תועלת לעולם הטכנולוגיה.

אז הגעתי להחלטה לפתח תוכנה לחיפוש במצלמת אבטחה לפי אובייקט. שבפרויקט כזה גם אחשף לתחום הבינה מלאכותית לצורך זיהוי האובייקטים והמעקב אחריהם, וגם אשתמש בכלים שרכשתי במהלך הלימודים להתמודד עם אלגוריתמים מורכבים במבנה נתונים

2.4.3 מוטיבציה לעבודה

פיתוח בתחום החיפוש במצלמת אבטחה והבינה מלאכותית מציע אתגרים מרתקים והזדמנויות ליצירת פתרונות חדשניים ויצירתיים.

כיום התחום של AI הוא התחום המרכזי בעולם הטכנולוגיה והמדעים המחשביים. החשיבות של התפתחות בתחום זה היא בלתי מעורערת, והשקעה בו יכולה להביא לחידוש טכנולוגי מהפכני.

אני רואה עניין גדול ללמוד נושאים חדשים ואקטואליים בעולם ההייטק, להתפתח ולהכיר בטכנולוגיות בתחום הבינה מלאכותית ורשתות נוירונים עמוקות, ובפיתוח פרויקט כזה ניתנת לי הזדמנות להתעסק בטכנולוגיות מתקדמות ולהיות בעיקר החדשנות הטכנולוגית.

כמו"כ אני רואה את היתרונות שיכולים להביא פתרונות חדשים בתחום החיפוש במצלמה ורוצה לתרום לשיפור הביטחון והמערכות האבטחתיות.

2.4.4 הצורך עליו עונה הפרויקט

הפרויקט מסייע בשיפור טכנולוגי בתחום מערכות האבטחה והמצלמות. במקום שימוש בשיטות חיפוש מורכבות ומייגעות הפרויקט מביא פתרון חדשני ומתקדם שיכול להביא לשיפור רב ביכולת הזיהוי והחיפוש במצלמות, כמו כן הפרויקט מטמיע טכנולוגיות מתקדמות היכולות לבצע את תהליך הזיהוי בצורה אוטומטית ומהירה באופן משמעותי, חוסכות משאבים ומגוון מאמצים.

2.4.5 הצגת פתרונות לבעיה

מסלול קצר על עץ ההסרטות – מעבר רק על הרמה האחרונה של העץ

מודל מאומן לזיהוי אובייקטים המבוסס על רשת הנוירונים

יצירת אינדקס של אובייקטים בהסרטות לחיפוש מהיר ויעיל

3 מטרות ויעדים

המטרה שעמדה לנגד עיני היא רכישת ידע והכרה נרחבת הכולל גם את תחום הבינה מלאכותית שלאחרונה צברה תאוצה בתחום ההייטק וכן ידע מקיף בטכנולוגיות וספריות של פיתון.

מטרת העל:

חיסכון במשאבים, כגון: כוח אדם וזמן.

מטרות נוספות:

- פיתוח תוכנה יעילה ונוחה לשימוש
- שמירת הנתונים באופן מסודר ויעיל
- תכנון המערכת תוך שימת דגש על כתיבה נכונה, מאורגנת ומקצועית של הקוד

יעדים:

- קבלת תמונה או הגדרות האובייקט וטווח הזמן לחיפוש
- חיפוש האובייקט תוך מעבר יעיל על עץ ההסרטות
- ניתוח תזוזות של האובייקט
- אינדוקס תוצאות מהחיפוש
- אפשרות צפיה בחיפושים אחרונים

4 אתגרים

במהלך כתיבת הפרויקט עמדתי בפני אתגרים שונים אשר לימדו אותי והעניקו לי ניסיון רב ומיומנויות חדשות, ביניהן:

- לימוד חומר חדש
- התמצאות בשפת פיתון
- חקירת נושא הפרויקט
- העשרת ידע בתחום ה AI
- פתירת באגים
- לימוד והבנה של קודים והשימוש בהם.

חקר עצמאי של הנושא ולימוד חומרים רבים וספריות שונות ומובילות
ב Python שהפרויקט משתמש בהן,כגון:

- **tkinter** - ספריית גרפיקה לממשק משתמש (GUI) בשפת פיתון. היא מאפשרת לפתח יישומים גרפיים מתקדמים באמצעות מרכיבים גרפיים ותכניתים מובנים לניהול אירועים. בשימוש ב Tkinter- ניתן ליצור חלונות, מסגרות, כפתורים, תיבות טקסט, רשימות, תמונות ועוד.
  היא מציעה ממשק פשוט ונוח לשימוש, עם אפשרויות רבות להתאמה אישית של מראה והתנהגות היישומים. היא מספקת מגוון מרכיבים גרפיים שאפשר לשלב יחד כדי ליצור ממשקים משוכללים יותר, וניתן להשתמש בתכונות ובפעולות מובנות כדי לטפל באירועים כמו לחיצות עכבר והקלדה.

- **PIL** - (Python Imaging Library) ספריה לעיבוד תמונה בשפת פיתון. היא מציעה אפשרויות רבות לעבודה עם תמונות כולל טעינה, שמירה, שינוי גודל, חיתוך, סיבוב, שינוי צבעים ועוד.
  עם ספרית PIL ניתן לטעון תמונות מקבצים במגוון פורמטים כמו: jpeg , png ,bmp, gif, ולבצע שינויים על התמונות. ניתן גם לבצע פעולות מתקדמות כמו הפקת תת-תמונות, שינוי רמת האור והניגודיות, והוספת אפקטים כמו פיקסליזציה וטשטוש.

- **cv2**- ספריה מתקדמת לעיבוד וניתוח תמונות ווידאו. היא מציעה אפשרויות רחבות לעבודה עם תמונות, כולל טעינה, עיבוד, ניתוח ושינויים מתקדמים בתמונות. עם ספרית CV2 ניתן לטעון תמונות ווידאו ממקורות שונים ולבצע עליהן פעולות כמו חיתוך ,שינוי גודל, שינוי צבעים, הפיכת תמונות לשחור-לבן, שימור או שינוי פורמטים ועוד. ניתן לעבוד על פיקסלים יחידים ולגשת לערכים של הערכי RGB , HSV ועוד. בנוסף היא מציעה פונקציות מתקדמות כמו זיהוי ומעקב אובייקטים בתמונה, הפעלת

  פילטרים ומסננים, זיהוי פנים ותכונות בפנים, זיהוי קווים וקשתות, התאמה צבעים , התאמה גאומטרית, וזיהוי וניתוח מופעים בווידאו.

- **numpy**– ספריה המאפשרת לעבוד עם מערכים רב-ממדיים, ומספקת פונקציות מתמטיות לעבודה עם מערכים וכן מספקת יכולות לעיבוד ושדרוג תמונות באמצעות מערכים.

- **os** - (Operating System) ספריה שמספקת גישה לפונקציות מערכת הפעלה, מציעה פונקציות ליצירת, קריאה, כתיבה, מחיקה ושינוי שמות של קבצים ותיקיות במערכת הפעלה. ניתן לנווט במבנה הקבצים והתיקיות, לבדוק את קיומם ותכולתם ,ולבצע פעולות מתקדמות כמו גישה להרשאות ופרסום מבני קבצים. כמו"כ שינוי משתני סביבה, קריאת פרמטרים של סביבת הפעלה, יצירת תהליכים חדשים ועוד.

- **matplotlib** – היא ספרית גרפיקה בפיתון שמאפשרת ליצור תרשימים וגרפים בצורה פשוטה וגמישה. היא מציעה את היכולת ליצור תרשימים בתצורות שונות, כולל גרפים קוויים, עמודות, מאגרים, פיצוץ, תרשימי פיתגורס, דיאגרמות בתורת הגרפים ועוד. הספרייה מציעה אפשרויות רבות להתאמה אישית של התרשימים, כולל אפשרויות לשנות את הצבעים, הסגנונות, המרכיבים הגרפיים ותוויות הצירים.

- **imageio** – ספריה התומכת בעיבוד וקריאה של קבצי וידאו בפיתון. היא מספקת את אותן יכולות של קריאה, כתיבה ועיבוד כמו במקרה של תמונות ויכולות נוספות כגון חיבור וחיתוך של וידאו, שינוי צבעים ואפקטים ,השמת כתובים ועוד.

- **Moviepy** – גם היא ספרית פיתון לעיבוד וידאו, המספקת מגוון יכולות לעיבוד ועריכת וידאו. הספרייה מאפשרת ליצור, לערוך, לפתוח, לשמור ולהציג סירטונים וידאו מתוך קוד פיתון.

- **Webbrowser** - היא ספרית מובנית שמאפשרת פתיחת דפדפן ותצוגת דפים אינטרנטיים ישירות מתוך קוד פיתון.

5 מדדי הצלחה למערכת

- הצגת תוצאות תואמות לפרטי הבקשה של המשתמש
- מעקב אחר התרחשויות שזמנם אינו ידוע במדויק
- מצב שבו המערכת מזהה בהקלטות אובייקטים והתזוזות שלהם ברמה שעין של בן אדם ממוצע לא יכולה לקלוט.

6 רקע תאורתי / ספרות מקצועית

הבעיה שלה נדרשתי למצוא פתרון היא לאתר אירוע ממצלמות אבטחה שזמן התרחשותו אינו ידוע במדויק.

נגדיר בדיוק את מה אנחנו רוצים לראות ע"י תמונה ולבסוף נקבל וידאו קצר הכולל את האירוע אותו רצינו לאתר.

תחילה מפעילים פונקציה המזהה מה החפץ אותו רוצים למצוא, לאחר מכן חיפוש החפץ ע"י האינדוקס שנעשה לאובייקטים בחיפושים הקודמים, במידה ולא נמצא – חיפוש החפץ תוך מעבר על עץ ההסרטות.

כאשר מוצאים את האובייקט אותו חיפשנו, נפעיל פונקציית מעקב אחר האובייקט עד ההחלטה איפה בדיוק זז ואז אותו חלק בהסרטה בו זז - יחתך ויוצג על המסך.

בנוסף, תוצאות החיפוש יכתבו לקובץ לפי דרישת המשתמש.

7 תיאור מצב קיים

למיטב ידיעתי אופן החיפוש היום בהקלטות ממצלמות אבטחה נעשה בצורה שדורשת זמן ממושך בצפייה בהקלטות וריכוז ומאמץ רב מצד הבנ"א למציאת האובייקט שמחפש , מה שגם מונע מעניינים חשובים להסתדר מהר כמו איתור גנבים לדוגמה.

8 ניתוח חלופות מערכתי

בכדי להגיע לפתרון הסופי הוצרכתי לערוך השוואה בין האובייקטים המופיעים בהסרטות לבין תמונת האובייקט שהמשתמש העלה.

ב-OpenCV קיימים מספר אלגוריתמים וכלים שיכולים לשמש להשוואת אובייקטים בין תמונות ישבתי ימים שלמים וניסיתי מספר רב של אלגוריתמים.

הנה כמה מהאלגוריתמים הנפוצים:

Template Matching (התאמת תבנית): האלגוריתם מחפש תבנית קטנה או אובייקט בתוך התמונה המקורית. ניתן לסרוק את התמונה על ידי הזזת ריבוע התבנית או האובייקט על פני התמונה ולמצוא את המיקום הטוב ביותר עבור ההתאמה.

Histogram Comparison (השוואת היסטוגרמה): האלגוריתם משווה את ההיסטוגרמה של התמונות בכדי למצוא דמיון בין האובייקטים. השוואה יכולה להתבצע על יסוד ההתפלגות של ערכי הפיקסלים בין התמונות.

Edge Detection (זיהוי קווים): האלגוריתם מזהה ומדגים את הקווים והמתארים של האובייקטים בתמונה. זיהוי זה מאפשר להשוות את הקווים של האובייקט בין התמונות.

זיהוי תכונות באמצעות אלגוריתמים כמו ORB, SURF, SIFT :

זהו תהליך שבו מתבצעת חילוץ ואיתור של מאפיינים מיוחדים בתמונות. זיהוי התכונות מאפשר למערכות מחשוב לזהות ולהתאים בין תצורות, אובייקטים או אזורים בתמונות שונות.

התהליך כולל שני שלבים עיקריים:

חילוץ מאפיינים:

בשלב זה, האלגוריתם מחפש אזורים מיוחדים בתמונות שהם ייחודיים ומיוחדים במבניהם. האזורים המיוחדים הללו נקראים מאפיינים. הם יכולים להיות נקודות מסוימות, שיפועים, קווים או צורות מורכבות יותר, תלוי באלגוריתם הספציפי שמשמש.

התאמה וחישוב תיאור:

בשלב זה, המאפיינים שנחשפו בתמונות נשווים וניתן למצוא התאמות ביניהם. זה מתבצע על ידי חישוב תיאורים אינפורמטיביים לכל מאפיין. התיאורים נוצרים באמצעות פרמטרים מאובטחים ומיוחדים לכל מאפיין ומשמשים להשוואה והתאמה בין מאפיינים בתמונות שונות.

ORB, SIFT ו-SURF הם אלגוריתמים שימושיים לחילוץ וזיהוי תכונות בתמונות. כל אלגוריתם מציע גישות ייחודיות ואלגוריתמים מתקדמים יותר כמו SIFT ו-SURF משתמשים במתמטיקה מורכבת וחישובים מתקדמים על מנת לחלוץ ולתאר את המאפיינים. ORB, מצד שני, מתאפיין ביכולת מהירות ויעילות גבוהה והוא נמצא ביניהם מבחינת יעילות ודיוק בזיהוי התכונות.

תחילה פילחתי את האובייקט מהתמונה כדי שההשוואה תהיה רק בין האובייקטים בלי קשר לרקע בו צולמו שבטח הוא שונה ולאחר שבדקתי כל אחד מהאלגוריתמים על מספר תמונות ואף אחד לא הביא לי תוצאות מקסימליות ומדויקות מסקנתי הייתה כי שינויים בתנאי התאורה, בזוויות הצילום, ברעש תמונה ובתפוקת המצלמה עשויים להשפיע על איכות התמונה ולגרום לקושי בזיהוי והשוואה בין האובייקטים.

הבנתי שכרגע לפרויקט שלי לא נמצא אלגוריתם רלוונטי להשוואה בין האובייקטים שהרי התמונה שהמשתמש יעלה קרוב לוודאי שצולמה בתנאי תאורה אחרים ובטח שאם עוקב אחר גניבה מסתבר שהאזור חשוך, כמו"כ האובייקטים מופיעים מול המצלמה מכיוון שונה.

יוצא שהאלגוריתמים המתוארים לא מספקים את מה שדרוש לפתרון הסופי.

9 תיאור החלופה הנבחרת

תחליף לאלגוריתם השוואה יעיל שלעת עתה לא נמצא שהחלטתי לעשות הוא הצגת הפריים המכיל את האובייקט (שיכול להיות רלוונטי) למשתמש, ועליו לאשר בלחיצה על הכפתור כן / לא אם התכוון לאובייקט הזה או לא.

אני חושבת שבאופן כזה כשהמשתמש מאשר את האובייקט אנחנו בטוחים במאה אחוז שאת זה חיפש

10 אפיון המערכת

10.1 ניתוח דרישות המערכת

סביבת פיתוח:

חומרה – מעבד i5-11357 8 GB

עמדת פיתוח – מחשב Intel

מערכת הפעלה – windows 10

שפות תוכנה – Python

כלי תוכנה לפיתוח המערכת – vs code , Jupyter notebook

מסד נתונים - File system

חיבור לרשת – לא נדרש

עמדת משתמש מינימאלית:

חומרה: מעבד i5 8 GB

מערכת הפעלה: windows 10 ומעלה

תוכנות: vs code

דרישות בהם המערכת צריכה לעמוד:

- כתיבה בסטנדרטים מקצועיים.
- מחשוב השרות ללקוח.
- כתיבת הקוד בסיבוכיות היעילה ביותר.
- ממשק נוח וידידותי למשתמש.
- תגובה מהירה ככל שניתן למשתמש.

10.2 מודול המערכת – תהליכים מרכזיים

- קליטת פרטי המצלמה וטווח הזמן לחיפוש
- העלאת תמונה של האובייקט לחיפוש
- זיהוי הקטגוריה אליה שייך האובייקט

- מעבר על עץ אינדקס לבדיקה אם כבר נערך בעבר חיפוש של אותה קטגוריה
- במידה וזו לא פעם ראשונה שעורכים חיפוש של הקטגוריה יהיה מעבר על הרשימה מקושרת השייכת לקטגוריה והצגת תמונות של האובייקטים המעודכנים שם בטווח הזמן המבוקש
- מעבר על עץ ההסרטות בטווח הרצוי פחות הזמנים שהופיעו ברשימה של הקטגוריה בעץ אינדקס
- פירוק כל הסרטה לפריימים ושליחת הפריימים לזיהוי האובייקטים המופיעים בו עד מציאת האובייקט המבוקש תוך אינדוקס האובייקטים המזוהים במשך ההסרטה.
- מעקב אחר האובייקט וניתוח התזוזות שלו
- החזרת תגובה למשתמש: קטע מההסרטה בו נראתה תנועה של האובייקט
- עדכון תוצאות החיפוש בקובץ

10.3 אפיון פונקציונלי – פונקציות עיקריות

- () Identify_the_category_to_search_for – הפונקציה מזהה לאיזו קטגוריה שייך האובייקט אותו מחפשים.
- () One_box – הפונקציה קובעת את המסגרת האמיתית של האובייקט.
- () Check_in_index – הפונקציה עוברת על העץ אינדקס ובודקת אם כבר קיימת בו הקטגוריה. אם כן, עוברת על הרשימה ובודקת אם האובייקט הרצוי מופיע שם ואם לא, אז מוסיפה את הקטגוריה לעץ ועוברת לבצע מעבר על עץ ההסרטות.
- ()show <– ()Display_specific_frame – הפונקציות מציגות למשתמש פריים המכיל אובייקט לאישור כן/לא
- () Iterate_folders – הפונקציה מבצעת מעבר על עץ ההסרטות השייכות לטווח הזמן הרצוי.
- () Video_detect – הפונקציה מפרקת כל הסרטה לפריימים ומזהה את האובייקטים בכל פריים ומוסיפה כל אובייקט חדש הנראה בהסרטה לעץ אינדקס.
- ()Object_track - מעקב אחר האובייקט הרצוי
- ()Save_results – שמירת תוצאות החיפוש לקובץ

ועוד פונקציות נוספות...

10.4 ביצועים עיקריים

המשתמש מזין פרטים לצורך החיפוש.

המערכת טוענת את המודל ולאחר שזיהתה מה האובייקט לחיפוש היא מחפשת אותו תחילה בעץ אינדקס ולאחר מכן בעץ ההסרטות פחות הזמנים שהופיעו בעץ אינדקס, היא מפרקת כל הסרטה לפריימים ומבצעת בהם שינויים קלים להשיג תוצאות מדויקות יותר בזיהוי.

במהלך הזיהויים היא מבצעת אינדוקס לאובייקטים. ולאחר שזיהתה בהסרטה את האובייקט הרצוי היא מפעילה מעקב אחר האובייקט לזיהוי תנועה שלו.

לבסוף מציגה למשתמש קטע חתוך מההסרטה בו נראתה תנועה של האובייקט, ולאחר שהמשתמש יאשר שהמידע שנמצא מספק אותו המערכת תשמור את תוצאות החיפוש בקובץ

כמו כן יש אפשרות לצפות בתוצאות החיפושים הקודמים

10.5 אילוצים

המערכת מסתמכת על כך שהתמונה שהלקוח מעלה לחיפוש מכילה אובייקט אחד בלבד.

וכן התבססתי על היררכיה מסוימת של ההקלטות השמורות בתיקיות במחשב.

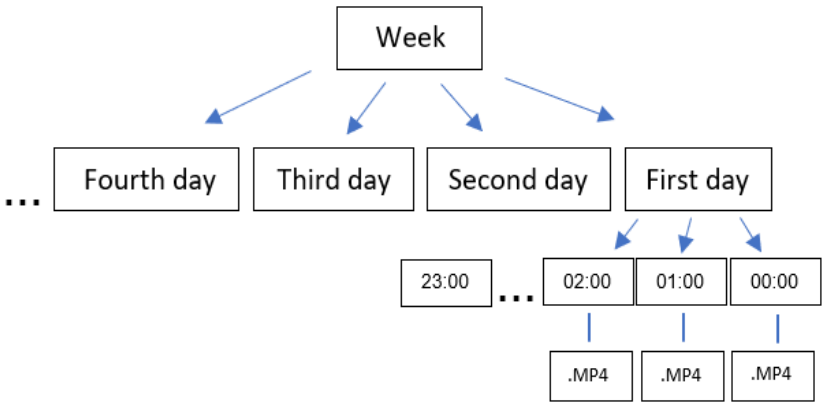
11 תיאור הארכיטקטורה

11.1 הארכיטקטורה של הפתרון המוצע בפורמט של: Down level Design-Top

תכנון הפרויקט נעשה מן הכלל אל הפרט. בתחילה הפרויקט תוכנן באופן כללי ובכל שלב ירדתי פנימה יותר לפירוט ולחשיבה מעמיקה עד לביצוע מלא של הפרויקט.

11.2 תיאור הרכיבים בפתרון

1. **קריאת הסרטות מתיקיות השמורות בצורה כזו:**



2. קובץ המתאר את כל המידע ותוצאות החיפושים שהיו.

קובץ pickle המכיל את המידע והתוצאות עבור החיפושים שהיו במבנה של אובייקטים, כאשר נרצה לצפות בחיפושים קודמים – תוכן הקובץ יטען למערך ועבור כל איבר המערכת תיצור כפתורים לתצוגת המידע על המסך

מסד נתונים: File system

פרוטוקולי תקשורת: file read/write

11.3 תיאור פרוטוקולי התקשורת

file read/write

11.4 שרת – לקוח

צד שרת: הנכתב בשפת Python

צד לקוח: הנכתב בשפת Python

12 ניתוח ותרשים UML / Use cases של המערכת המוצעת

רשימת ה Use cases:

- הזנת פרטי מצלמה ופרטים נוספים לצורך החיפוש
- לחיצה על yes or no אם התכוון אל האובייקט המוצג בחלון
- צפיה בתוצאות החיפוש

- צפיה בחיפושים קודמים

12.1 תיאור הUC-העיקריים של המערכת

Use Case 1:

Identifier: UC1

Name : חיפוש אובייקט

Description: המשתמש מכניס את פרטי המצלמה, טווח זמן לחיפוש ומעלה תמונה של האובייקט לחיפוש או מגדירו.

Actors: משתמש

Frequency: לאחר הפעלת התוכנה ובחירה בחיפוש חדש

Pre-conditions: הזנת פרטים לצורך החיפוש

Post-conditions: הפרטים נקלטים במערכת והמערכת מתחילה לחפש את האובייקט.

Assumptions : המשתמש מורשה לעיסוק במצלמות האבטחה

Decisions : אנו מסתמכים על נכונות הפרטים והתמונה שהועלתה מכילה את האובייקט בלבד

Use Case 2:

Identifier: UC2

Name : אישור האובייקט המוצג בחלון

Description: במהלך החיפוש מוצגות תמונות של אובייקטים למשתמש ועליו לאשר yes/no אם התכוון לאובייקט המוצג בחלון

Actors: משתמש

Frequency: תוך כדי שהמערכת מבצעת את החיפוש

Pre-conditions: לחיצה על כפתור yes or no

Post-conditions: המערכת בודקת את תגובת המשתמש וממשיכה בפעולות התואמות לתגובתו

*** Extended use cases:** קיימים הרבה אובייקטים עד שנמצא האובייקט המבוקש

Assumptions : קיים אובייקט ברשימה התואמת לפרטי החיפוש או זוהה בהסרטה

Decisions : אנו מסתמכים על נכונות התגובה

Use Case 3:

Identifier: UC3

Name : צפיה בחיפושים קודמים

Description: המשתמש יצפה בחיפושים קודמים הכוללים את: התאריך בו בוצע החיפוש, האובייקט, המצלמה, טווח הזמן שהתבקש, וסרטון התוצאה הסופית

Actors: משתמש

Frequency: לאחר הפעלת התוכנה ובחירה בצפייה בחיפושים קודמים

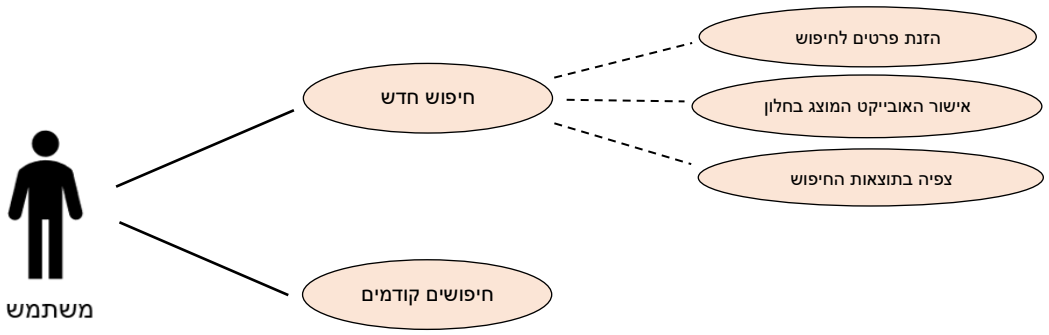
Pre-conditions: היה כבר חיפוש בעבר

Post-conditions: המשתמש יוכל לצפות בכל החיפושים שהתרחשו בעבר

Assumptions : המשתמש מורשה לעיסוק במצלמות האבטחה

Decisions : אנו מסתמכים על נכונות הפרטים

12.2 הצגת מקרה (use case) עבור כל הפונקציות העיקריות בפרויקט.



12.3 מבנה נתונים בהם השתמשתי

Array, TST- Trenary Search Tree, linked list

מבנה נתונים בשם ObjectsTree שיצרתי המשלב את תכונותיהם של העץ הטרינארי ועץ החיפוש וכן כולל בתוכו גם רשימות מקושרות באופן הזה:

עץ טרינארי – לכל צומת בעץ TST יש שלושה בנים לכל היותר: שמאלי, ימני ואמצעי.

עץ חיפוש – הערך השמור בכל צומת עץ TST גדול מכל ערך שנמצא בתת העץ השמאלי של הצומת, וקטן מכל ערך שנמצא בתת העץ הימני של הצומת – כמו בעץ חיפוש בינארי.

לכל צומת בעץ תהיה גם רשימה מקושרת המאותחלת ל None.

מבנה זה שומר את שמות הקטגוריות של האובייקטים שכבר חיפשו בהסרטות וכן את האינדקסים של האובייקטים שזוהו, כלומר: תאריך, שעה , מספר פריים, ומיקומו המדויק בפריים.

כמתואר להלן:

התו הראשון של שם הקטגוריה יימצא בשורש העץ, או בצומת אחר שניתן להגיע אליו משורש העץ, תוך כדי מעבר דרך בנים שמאליים או ימניים **בלבד**.

התווים הבאים של המחרוזות יימצאו רק בתת העץ - בבן האמצעי של צומת תחילת המחרוזת.

התו האחרון של המחרוזת יימצא בצומת כלשהו אשר יכיל סימן מיוחד המציין את סוף המחרוזת, והוא יכונה 'צומת סוף המחרוזת' ומשם תתחיל רשימה מקושרת.

כל איבר ברשימה מייצג אינדקס ויכלול: תאריך ושעה המייצגים את נתיב ההסרטה בה מופיע האובייקט.

את מספר הפריים בו הוא מופיע.

מיקומו בפריים – { x, y, w, h }

וכן מצביע לאיבר הבא ברשימה.

חשוב לי לציין שהרשימה תהיה תמיד ממוינת לפי התאריך והשעה.

12.4 חישוב יעילות האלגוריתם

בבואי לתכנן את האלגוריתם בפרויקט נתקלתי רבות בשאלות על היעילות. ביצועי הפרויקט חייבים להיות יעילים שכן זה קשור באופן ישיר לזמן התגובה למשתמש: ברגע שהביצועים גרועים, זמן התגובה למשתמש מתארך, המערכת 'חושבת' הרבה זמן מה שפוגע באחוזי ההצלחה של התוכנה.

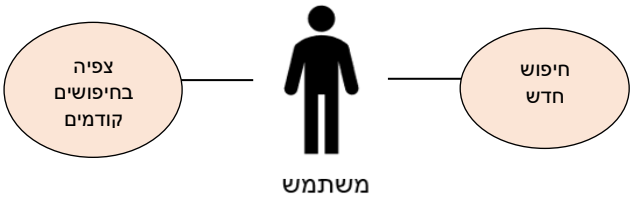
כמובן שאחרי כל פונקציה שיצרתי וראיתי שיש אפשרות לייעל, השתדלתי לייעל כמה שניתן.

להלן סיבוכיות זמני ריצה של הפונקציות העיקריות בפרויקט.

One_box	אם נסמן את מספר התיבות המקוריות ב n, ואת מספר התיבות המסוננות ב k, אזי סיבוכיות הזמן של האלגוריתם one_box היא בערך O(n*k)
Identify_the_category_to_search_for	O(n)

searchTST, insert	כללית, ניתן לומר שסיבוכיות הפונקציה searchTST היא O(h), כאשר h הוא גובה העץ (או גובה המסלול הארוך ביותר מהשורש לעלה)
search_linked_list	צמתים ברשימה המקושרת, הפונקציה n באופן כללי, כאשר יש פעולות לפני שמוצאת את הצומת המתאים או n/2 תבצע ממוצע O(n). מסיימת את החיפוש. לכן, סיבוכיות הפונקציה היא
add_sorted (הוספה לרשימה)	O(n)כאשר n הוא מספר הצמתים ברשימה המקושרת.
display_specific_frame	סיבוכיות הפונקציה display_specific_frame תלויה בגודל הווידאו, גודל התמונות בפריים, ובמקרה של עיצוב הפריים גם בגודל האובייקט המזוהה בתמונה.
video_detect	O(n), כאשר n הוא מספר הפריימים בווידאו
iterate_folders	O(n)
Object_track	סיבוכיות הפונקציה היא O(N), כאשר N הוא מספר הפריימים בוידיאו בהם מתבצע המעקב

12.5 תרשים UML



12.6 תיאור המחלקות המוצעות

יחידות הפרויקט

1. **מודל זיהוי אובייקטים**

קלט: תמונה

פלט: כל תוצאת זיהוי (Detection) מכילה מספר מאפיינים המשתמשים לתיאור האובייקט הנמצא בתמונה. המאפיינים העיקריים הם:

"id" או "label": מזהה האובייקט או תווית האובייקט, המציינת את הקטגוריה או השם של האובייקט שזוהה (לדוגמה: אופניים, רכב, כלב, וכו').

"confidence" או "score": הביטחון או הסיכוי לגבי הזיהוי הנכונות של האובייקט.
ערך גבוה מציין גיבוי חזק לזיהוי המדויק של האובייקט.

"bbox" או "bounding box": מיקום האובייקט בתמונה מוגבל באמצעות מלבן הגבולות (bounding box), שמציין את הגבולות האובייקט בתמונה. המלבן מוגדר על ידי קואורדינטות הפינה העליונה השמאלית והתחתונה הימנית (x_min, y_min, x_max, y_max).

2. **DB**

הקלטות מהמצלמה השמורות בדיסק המקומי בעץ תיקיות עם חותמות התאריך והשעה

3. **TST- Trenary Search Tree**

קובץ המתאר מבנה של עץ חיפוש הכולל רשימות מקושרות הוא מכיל את האינדקסים של האובייקטים שזוהו בהסרטות.

4. **search results**

קובץ המתאר את כל המידע והתוצאות של החיפושים שהיו.
הקובץ שמור בדיסק המקומי

5. **Algorithm**

כתוב בשפת פיתון

13 רכיבי ממשק

המערכת מורכבת משני חלקים:

1. תוכנית הלקוח
2. תוכנית השרת

14 תיכון המערכת

חיפוש תנועתו של אובייקט רצוי בעץ ההסרטות שצולמו במצלמת אבטחה

14.1 ארכיטקטורת המערכת

local system application

14.2 תיכון מפורט

צד שרת: python

צד לקוח: python

מסד נתונים: File system

15. תיאור התוכנה

סביבת פיתוח:

חומרה: מעבד i7

Ram: 16.0 GB

עמדת פיתוח: מחשב Intel

מערכת הפעלה:Windows 10

15.1 סביבת עבודה

- Jupyter notebook
- Visual studio code

15.2 שפות תכנות

- Python

16 תיאור מסכים:

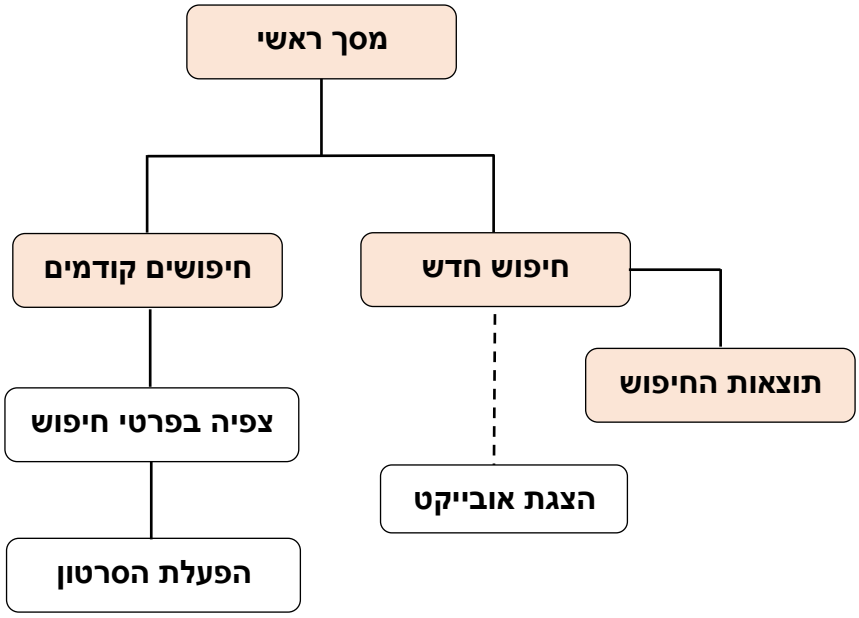
מסך ראשי

מסך חיפוש

מסך תוצאות החיפוש הנוכחי

מסך חיפושים קודמים

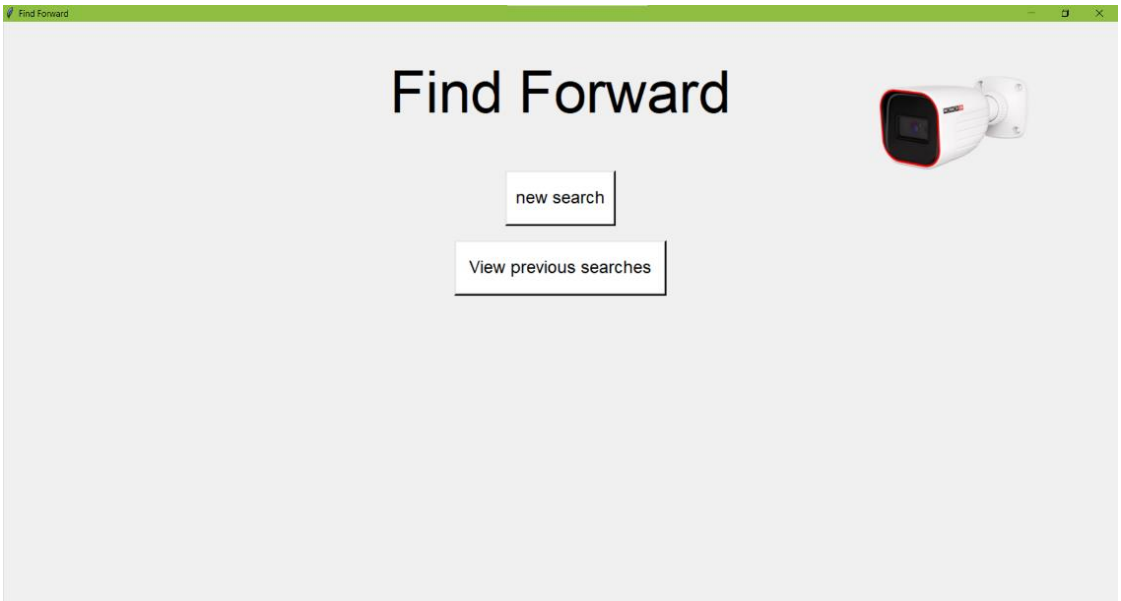
17 תרשים מסכים המתאר את היררכיית המסכים והמעברים ביניהם



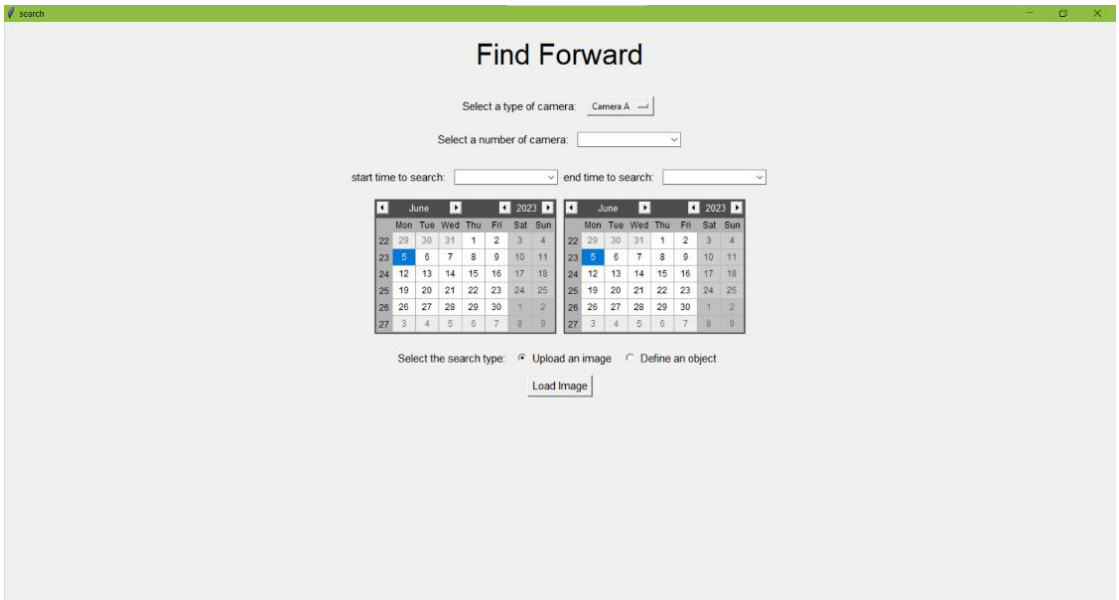
Screen flow diagram

18 מה תפקידו של כל מסך / חלון עם צילום מסך של החלון הרלוונטי

מסך ראשי: כפתור ראשון – מעבר לחיפוש חדש וכפתור שני לצפייה בחיפושים קודמים.



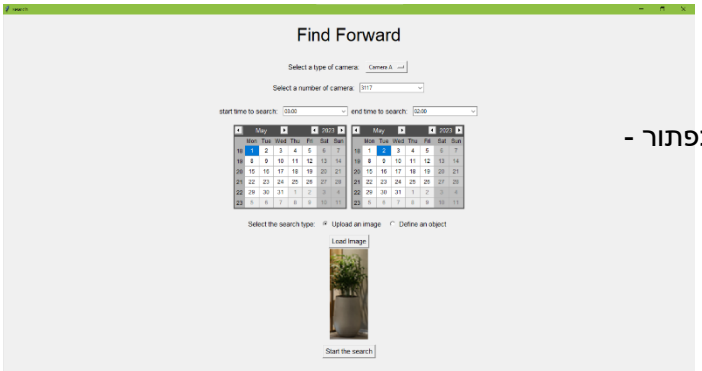
מסך חיפוש: תיבות טקסט להזנת פרטי המצלמה, תאריך ושעה התחלתיים וסופיים, בחירת סוג החיפוש (radio button) וכפתור להעלאת תמונה של האובייקט.



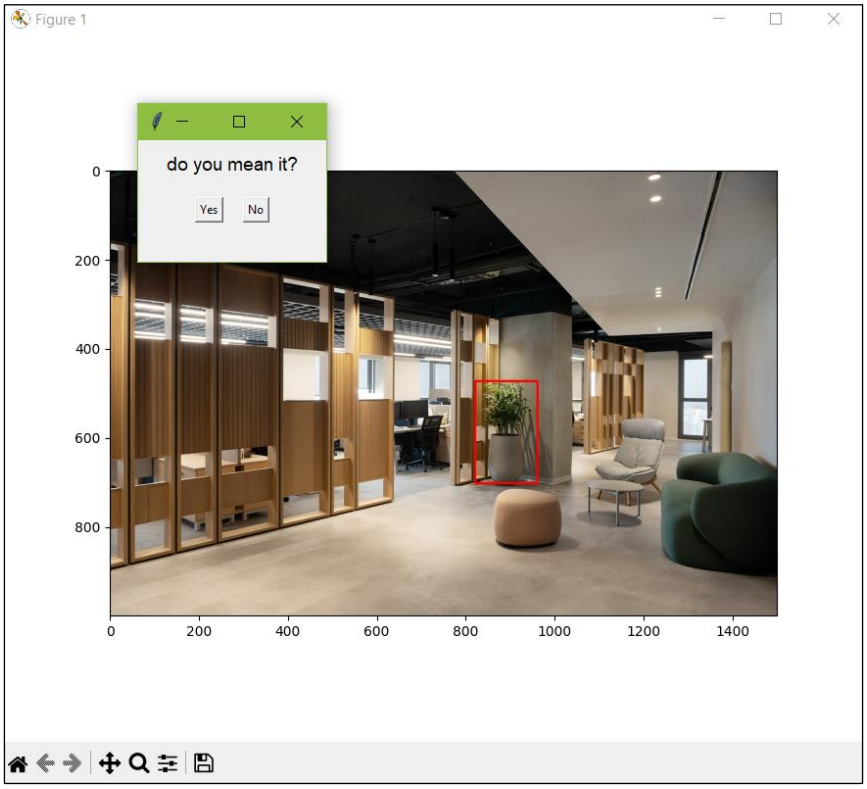
לאחר העלאת התמונה:

מוצגת התמונה ונוסף הכפתור -

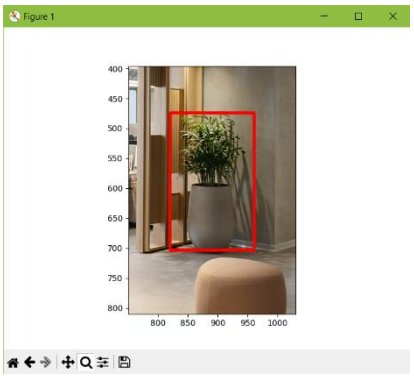
Start the search



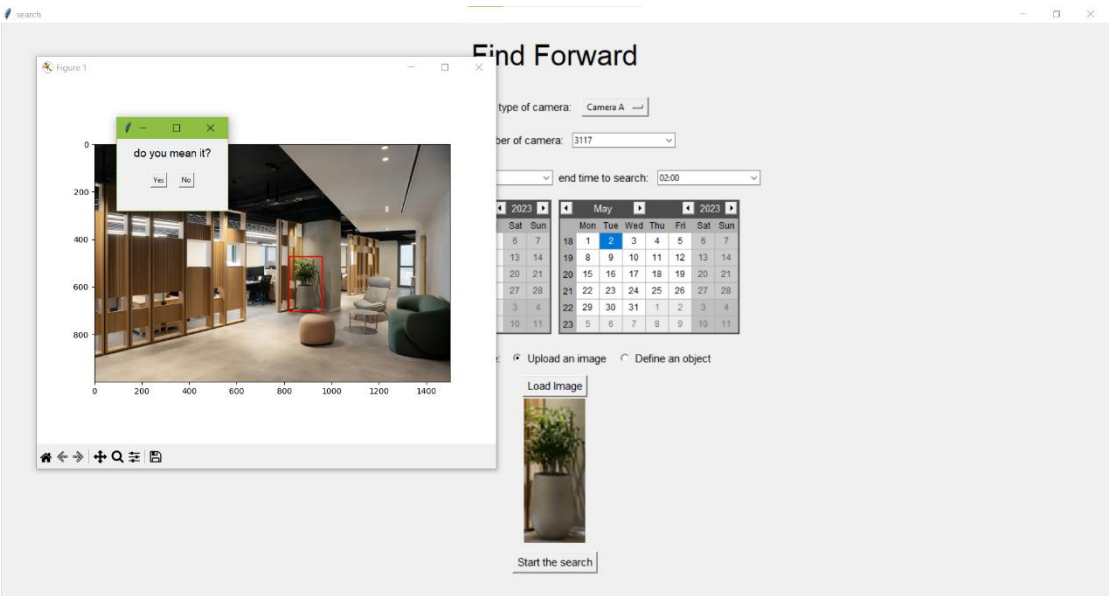
חלון matplotlib להצגת אובייקטים מהקטגוריה המחופשת המופיעים ברשימה בטווח הזמן הרצוי או תוך המעבר על ההסרטות עצמן. וחלון נוסף עם שני כפתורים לאישור המשתמש yes or no.



הצגת התמונה היא באופן נוח וידידותי למשתמש, המציעה אפשרויות כמו הגדלת התמונה וכד'.



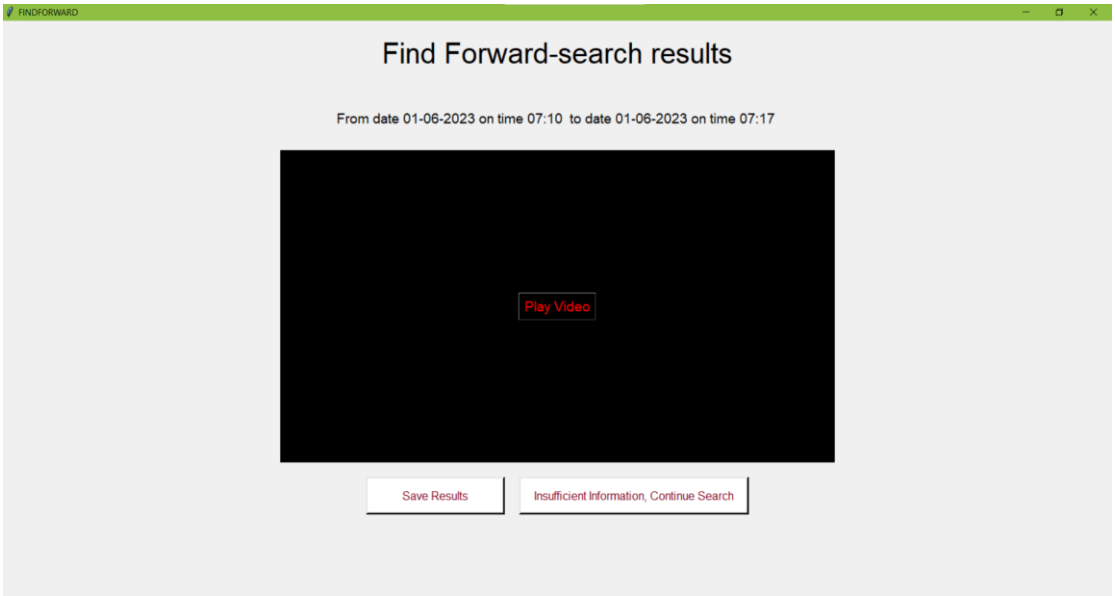
להלן הדגמת התהליך:



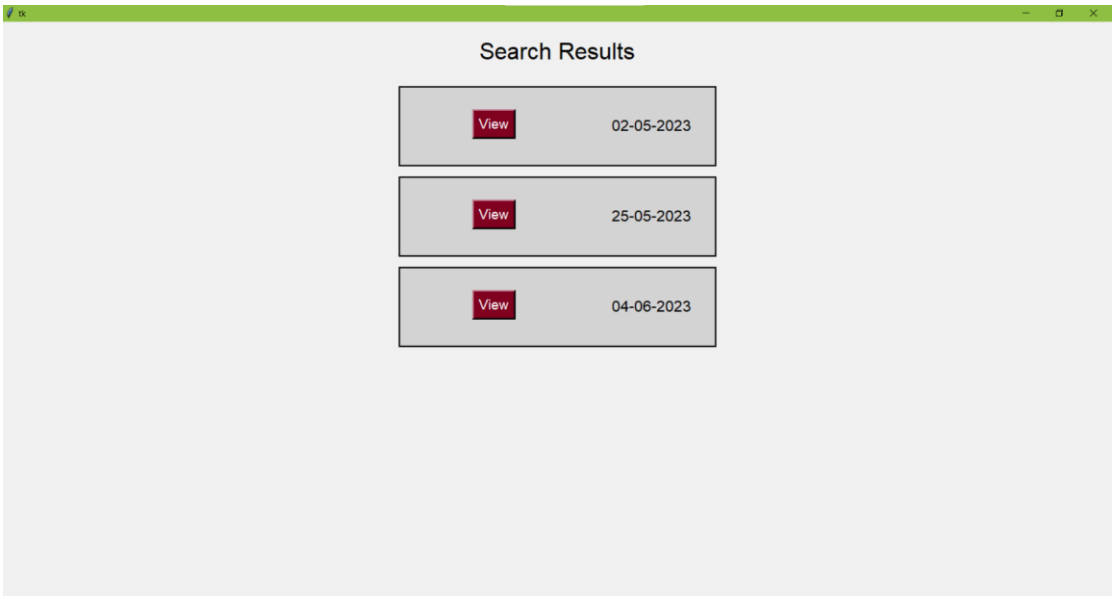
מסך תוצאת החיפוש: על המסך מוצג הזמן המדויק בו נמצא האובייקט בתנועה וכן כפתור Play Video להפעלת הסרטון.

כפתור Save Results לשמירת התוצאות

כפתור Insufficient Information, Continue Search להמשך החיפוש במידה והתוצאה לא סיפקה את המשתמש

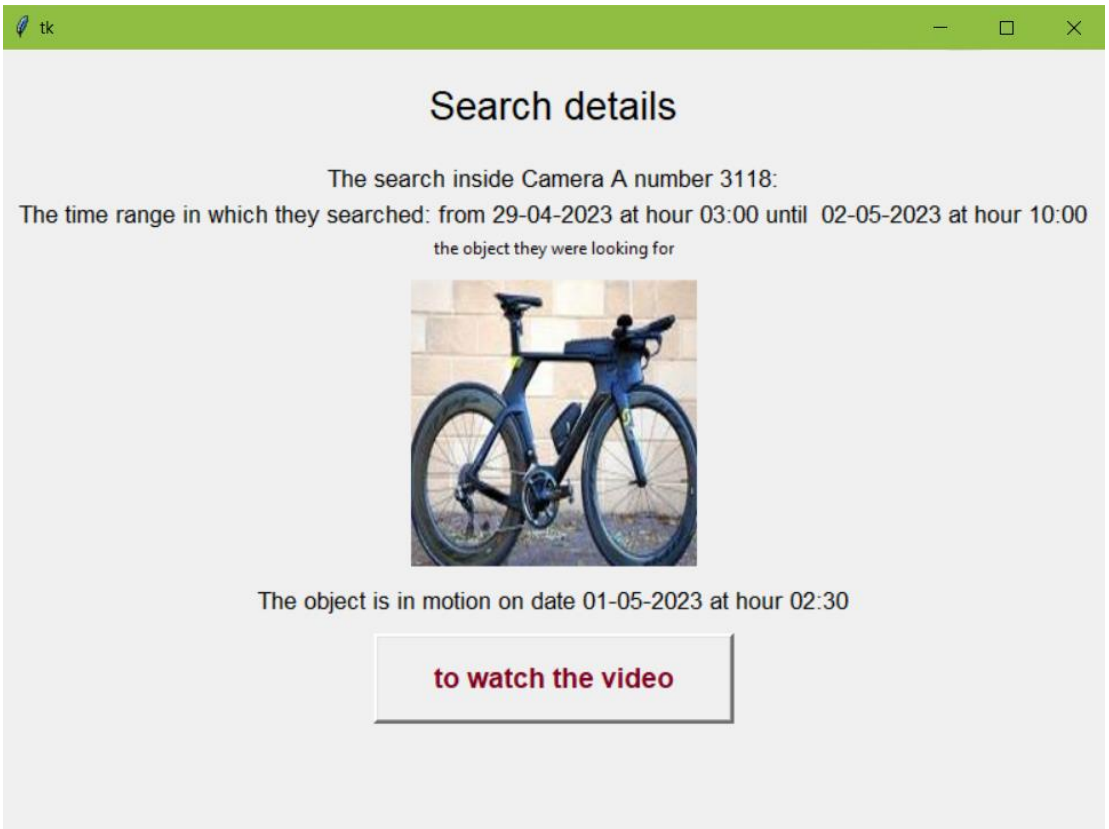


מסך צפיה בחיפושים קודמים: החיפושים מוצגים עם התאריך בו נערך החיפוש, לכל חיפוש כפתור view שבלחיצה עליו יפתח חלון המכיל את פרטי החיפוש התואם.



מסך פרטי חיפוש הנפתח בלחיצה על הכפתור view הוא כולל את פרטי החיפוש: המצלמה, מספר מצלמה, טווח התאריכים לחיפוש האובייקט, תמונת האובייקט וכן באיזה תאריך ושעה נמצא האובייקט בתנועה.

בנוסף יש כפתור to watch the video להפעלת הסרטון הכולל את תנועת האובייקט.



19 תיאור מסך הפתיחה

כפי הנכתב לעיל, מסך הפתיחה מכיל כפתור ניווט למסך של ביצוע חיפוש חדש וכן כפתור ניווט לצפייה בחיפושים קודמים.

20 הודעות למשתמש

במקומות מסוימים כמו בלחיצה על הכפתור "התחל חיפוש", במידה ולא כל הפרטים הוזנו, תוצג למשתמש הודעת alert שצריך למלא את כל הפרטים.

21 ממשק משתמש

נכתב בשפת python

22 קוד התוכנית-על פי סטנדרטים בליווי תיעוד

טעינת המודל לזיהוי אובייקט והשמת שמות המחלקות במערך.

```
import cv2
import numpy as np


# opencv DNN
net = cv2.dnn.readNet("resource/model/dnn_model/yolov4-tiny.weights",
                      "resource/model/dnn_model/yolov4-tiny.cfg")

# מודל זהוי אוביקט
model = cv2.dnn_DetectionModel(net)
model.setInputParams(size=(320, 320), scale=1/255)

# load classes list
classes = []
with open("resource/model/dnn_model/classes.txt", "r") as file_object:
    for class_name in file_object.readlines():
        class_name = class_name.strip()
        classes.append(class_name)
```

הפונקציה מקבלת את הנתיב לתמונה המכילה את האובייקט ובעזרת המודל מזהה לאיזו קטגוריה שייך האובייקט.

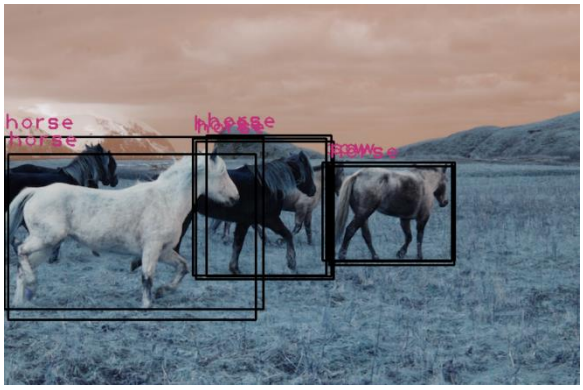
```
# זיהוי הקטגוריה לחיפוש
def Identify_the_category_to_search_for(filename):
    img = Image.open("{}".format(filename))
    img = img.convert("RGB")
    np_img = np.array(img)
    (class_ids, scores, bbox) = model.detect(np_img)
    final_boxes =One_bounding_box.one_box(bbox, scores,np_img.shape[0],np_img.shape[1])

    return class_ids[final_boxes[0]]
```

כשבצעתי את הקריאה לזיהוי האובייקטים בתמונה

```
(class_ids, scores, bbox) = model.detect(np_img)
```

התוצאות שקיבלתי לא היו לי ממש מדויקות, היו פעמים שראיתי שאובייקט זוהה כביכול מספר פעמים, נקבעו לו יותר מתיבה תוחמת אחת. וזאת למרות שדגם YOLO פועל באמצעות הטכניקה של IOU - צומת מעל איחוד (הוסבר לעיל).

להלן דוגמת הרצה:



תחילה לא הבנתי כיצד יתכן אך לאחר שניסיתי את הזיהוי על תמונות שונות הגעתי לכמה סיבות שעשויות לגרום לתוצאות אלה, מתוכם:

- במקרים בהם האובייקטים בתמונה נמצאים במצב מסתובב.
- התאמות חלקיות: אזורים של התוצאה המתקבלת מהמודל יכולים להיות דומים לאובייקטים אמיתיים בתמונה, אך לא לחלוטין זהים. זאת יכולה להתרחש כתוצאה מהתמודדות המודל עם אתגרים כמו חוסר וידוי בתמונה, עומס רעשים או תנאי תאורה מסובכים. במקרים כאלה, התוצאה המקובלת יכולה לכלול מספר תיבות תוחמות עבור האובייקט הממוקם בקרבת האזור.

חשבתי רבות איך אוכל להתגבר על הבעיה ולאחר מאמצים עלה לי רעיון –

התוצאה המתקבלת מהזיהוי כוללת:

bbox - רשימה של קואורדינטות המלבן המתאר את מיקום האובייקט המזוהה בתמונה. המלבן מוגדר על ידי הפינה השמאלית העליונה וכן הרוחב והגובה שלו (x,y,w,h)

Scores - זוהי רשימה של סבירויות לזהות נכונה של האובייקט המזוהה. סכום גבוה מציין את הסבירות הגבוהה יותר לזיהוי נכון. בכל זיהוי האובייקט מקבל סכום סבירות משלו.

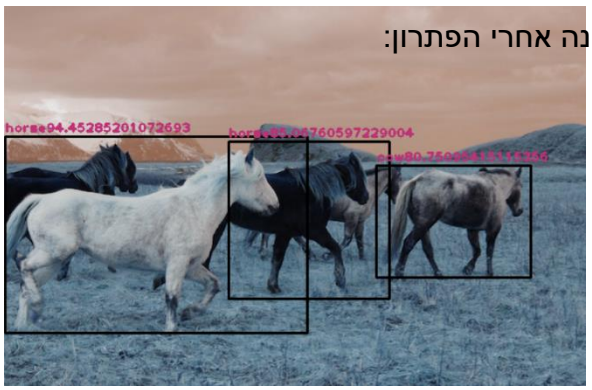
להלן הפתרון:

```
# הפרש בין 2 מספרים כערך מוחלט
def Difference(num1,num2):
    return abs(num1-num2)

# חישוב מרחק בין 2 נקודות
def distance(x1, y1, x2, y2):
    dist = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
    return dist

# רוחב המסגרת באחוזים מתוך כל התמונה
def width(w_rectangle, w_frame):
    return w_rectangle/w_frame*100

# גובה המסגרת באחוזים מתוך כל התמונה
def heigth(h_rectangle, h_frame):
    return h_rectangle/h_frame*100
```

```
# מסננת תיבות מיותרות
def one_box(boxes, scores, w_frame, h_frame):

    #  מיין את התיבות לפי ציון הביטחון בסדר יורד
    ids = np.flip(np.argsort(scores))
    i = 0
    while i < len(ids)-1:
        w=width(boxes[ids[i],2],w_frame)
        h=width(boxes[ids[i],3],h_frame)
        j=i+1
        while j < len(ids):
            if(Difference(w,width(boxes[ids[j],2],w_frame))<8 and
                Difference(h,heigth(boxes[ids[j],3],h_frame))<8 and
                distance(boxes[ids[i],0],boxes[ids[i],1],boxes[ids[j],0],boxes[ids[j],1])<60):
                ids = np.delete(ids, j)
            else:
                j=j+1
        i=i+1
    return ids
```

הפונקציה one_box מקבלת את bbox , Scores , רוחב התמונה וגובה התמונה. תחילה ממיינת את התיבות לפי ציוני הבטחון שלהם בסדר יורד תוך שמירת האינדקסים המקוריים ועבור כל אחד, על ידי חישובים באחוזים בודקת אם קיים אחריו ברשימת התיבות תיבה שהפינה השמאלית העליונה שלה לא רחוקה מהפינה שלו וכן הרוחב והגובה של התיבה מתקרבים לגדלים שלו. אם כן, מסתבר שמדובר באותו אובייקט ולכן מוחקת אותו עם הציון הנמוך מהרשימה. לבסוף היא מחזירה רשימה של האינדקסים של האובייקטים האמיתיים הסופיים עם ציוני הבטחון הגבוהים.

וזו התוצאה לאותה תמונה אחרי הפתרון:



מחלקות היוצרות את עץ האינדקסים

מחלקת Node – צומת בעץ

```
class Node:
    def __init__(self, data):
        self.data = data
        self.isEndOfString = False
        self.left = None
        self.eq = None
        self.right = None
        self.next = None  # מצביע לרשימה הבאה
```

פונקציות הוספת קטגוריה חדשה לעץ אינדקס וכן חיפוש קטגוריה בעץ.

```
# הוספת מילה חדשה
def insert(root, word):
    if not root:
        root = Node(word[0])
    if word[0] < root.data:
        root.left = insert(root.left, word)
    elif word[0] > root.data:
        root.right = insert(root.right, word)
    else:
        if len(word) > 1:
            root.eq = insert(root.eq, word[1:])
        else:
            root.isEndOfString = True
    return root


# חיפוש מילה בעץ ובמקרה ונמצא מוחזר ראש הרשימה המקושרת
def searchTST(root, word):
    if not root:
        return False
    if word[0] < root.data:
        return searchTST(root.left, word)
    elif word[0] > root.data:
        return searchTST(root.right, word)
    else:
        if len(word) > 1:
            return searchTST(root.eq, word[1:])
        else:
            if root.isEndOfString:
                return root
            else:
                return None
```

מחלקת LinkedListNode – איבר ברשימה מקושרת המייצג את האובייקט: נתיב להסרטה בה מופיע, מס' פריים ומיקום.

```
class LinkedListNode:
    def __init__(self, date, time, numFrame,box):
        self.date = date
        self.time = time
        self.numFrame = numFrame
        self.location=box
        self.next = None  # מצביע לצומת הבא ברשימה
```

פונקציה העושה אינדוקס לאובייקט חדש שזוהה בהסרטות

```
# הוספת איבר לרשימה מקושרת תוך שמירה על רשימה ממוינת

def add_sorted(headOfList, new_object):
    if headOfList.next is None:
        # return new_video
        headOfList.next=new_object
        return

    # מיון לפי תאריך ושעה
    if headOfList.next.date > new_object.date or (headOfList.next.date == new_object.date
                                                  and headOfList.next.time > new_object.time):
        new_object.next = headOfList.next
        headOfList.next=new_object
        return new_object

    curr = headOfList.next
    while curr.next is not None and (curr.next.date < new_object.date or (curr.next.date == new_object.date and
                                                  curr.next.time <= new_object.time)):
        curr = curr.next

    new_object.next = curr.next
    curr.next = new_object
    return headOfList
```

```
# חיפוש ברשימה המקושרת
def search_linked_list(videos, date, time):
    curr = videos.next
    while curr is not None:
        if curr.date == date and curr.time == time:
            return curr
        curr = curr.next
    return None
```

טעינת העץ אינדקס מהקובץ

```
# טעינת העץ והרשימה מהקובץ באמצעות Pickle
with open("data.pickle", "rb") as file:
    loaded_data = pickle.load(file)
```

שמירת העץ אינדקס בקובץ

```
# שמירת העץ והרשימה לקובץ באמצעות Pickle
with open("data.pickle", "wb") as file:
    pickle.dump( loaded_data, file)
```

מחלקת Previous_searches – אובייקט המתאר פרטי חיפוש שהתרחש, כל החיפושים נשמרים בקובץ.

```
class searchResults:
    def __init__(self, date,camera,numCamera,TimeRangeStart,TimeRangeEnd
                 ,Description,pathImage,pathVideo):
        self.date = date
        self.camera = camera
        self.numCamera = numCamera
        self.TimeRangeStart = TimeRangeStart
        self.TimeRangeEnd = TimeRangeEnd
        self.Description = Description
        self.pathImage = pathImage
        self.pathVideo = pathVideo
```

הוספת חיפוש שנערך לקובץ המתאר את החיפושים הקודמים שנערכו.

```
def add_search_result(date, camera, numCamera, TimeRangeStart, TimeRangeEnd, Description, pathImage, pathVideo):
    file_path = "objects.pkl"

    # טעינת האובייקטים מהקובץ אם הקובץ קיים
    if os.path.exists(file_path):
        with open(file_path, "rb") as file:
            objects = pickle.load(file)
    else:
        objects = []

    # יצירת אובייקט חדש והוספתו לרשימה
    new_object = searchResults(date, camera, numCamera, TimeRangeStart, TimeRangeEnd, Description, pathImage, pathVideo)
    objects.append(new_object)

    # שמירת הרשימה חזרה לקובץ
    with open(file_path, "wb") as file:
        pickle.dump(objects, file)
```

הפונקציה הבאה מקבלת את טווח הזמן בו רוצים לחפש וכן את הקטגוריה של האובייקט

הפונקציה בודקת אם כבר נערך בעבר חיפוש של אובייקט מהקטגוריה הזו. אם כן, היא תבצע תחילה חיפוש בעץ לפי אינדקס ואם לא היא תוסיף את הקטגוריה הנוכחית לעץ ותעבור מיד על עץ ההסרטות.

```
def check_in_index(start_date, start_time, end_date, end_time, class_id):
    start_date = datetime.strptime(start_date, "%d-%m-%Y")
    start_date = start_date.strftime("%d-%m-%y")
    end_date = datetime.strptime(end_date, "%d-%m-%Y")
    end_date = end_date.strftime("%d-%m-%y")
    my_array = []

    head_list = node.searchTST(loaded_data, classes_objects.classes[class_id])
    if head_list:  # אם הקטגוריה כבר מופיעה באינדקס
        current = head_list.next
        while current and current.date <= end_date:
            if (start_date < current.date < end_date) or (start_date == current.date and
                                                          current.time >= start_time) or (end_date == current.date and current.time <= end_time):
                found = display_specific_frame(
                    "01-05-2023_07-05-2023\\"+current.date+'\\'+current.time, current.numFrame, current.location)
                if found:  # אם נמצא האובייקט אפשר לעצור את הלולאה
                    break

                existing_item = next(
                    (item for item in my_array if item["date"] == current.date and item["time"] == current.time), None)
                if existing_item:
                    existing_item["numFrame"] = current.numFrame
                else:
                    new_item = {"date": current.date,
                                "time": current.time, "numFrame": current.numFrame}
                    my_array.append(new_item)
            current = current.next

            # מעבר על עץ ההסרטות פחות הזמנים שכבר הופיעו ברשימה
            Beyond_the_filming_tree.iterate_folders(
                start_date, start_time, end_date, end_time, my_array, class_id, head_list)
```

```
else:
    # הוספת קטגוריה חדשה לאינדקס
    node.insert(loaded_data, classes_objects.classes[class_id])
    head_list = node.searchTST(
        loaded_data, classes_objects.classes[class_id])
    Beyond_the_filming_tree.iterate_folders(
        start_date, start_time, end_date, end_time, None, class_id, head_list)

    # שמירת העץ והרשימה לקובץ באמצעות Pickle
with open("data.pickle", "wb") as file:
    pickle.dump(loaded_data, file)
    print("finish")
```

הפונקציה הבאה מקבלת נתיב להסרטה, מספר פריים ומיקום האובייקט בפריים. הפונקציה תפתח את ההסרטה ותקרא את הפריים שמספרו התקבל כקלט, תסמן את האובייקט ותקרא לפונקציה show() שתציג את הפריים למשתמש ויהיה עליו לאשר באמצעות הכפתורים שיופיעו אם אכן התכוון לחפש את האובייקט המסומן בתמונה או לא.

```
# הצגת פריים המכיל אוביקט השייך לקטגוריה הנמצא באינדקס
def display_specific_frame(video_path, frame_number, location):
    file_video = os.listdir(video_path)
    video_path1 = os.path.join(video_path, file_video[0])
    video_reader = imageio.get_reader(video_path1)

    # בדוק אם לסרטון יש לפחות פריים אחד
    if len(video_reader) == 0:
        print("The video is empty")
        return

    # בדוק אם מספר המסגרת נמצא בטווח התקף
    if frame_number < 0 or frame_number >= len(video_reader):
        print("Invalid frame number")
        return

    # קרא את המסגרת הרצויה
    frame = video_reader.get_data(frame_number)

    x1, y1, w, h = location
    cv2.rectangle(frame, (x1, y1), (x1+w, y1+h), (255, 0, 0), 3)
    pil_image = Image.fromarray(frame)
    found = showImage.show(pil_image)
    return found
```

```
def handle_yes():
    global found
    window2.destroy()
    plt.close()
    found = True

def handle_no():
    global found
    window2.destroy()
    plt.close()
    found = False
```

```python
def show(image):
    global window2
    # יצירת חלון של Tkinter
    window2 = tk.Tk()
    window2.title("Image Display")

    text_label = tk.Label(window2, text="do you mean it?", font=("Arial", 14))
    text_label.pack(pady=10)

    # הוספת הכפתורים
    buttons_frame = tk.Frame(window2)
    buttons_frame.pack(pady=10)

    yes_button = tk.Button(buttons_frame, text="Yes", command=handle_yes)
    yes_button.pack(side="left", padx=10)

    no_button = tk.Button(buttons_frame, text="No", command=handle_no)
    no_button.pack(side="left", padx=10)

    # יצירת חלון נוסף להצגת התמונה ב Matplotlib
    fig, ax = plt.subplots()
    ax.imshow(image)
    plt.show()
    return found
```

הפונקציה הבאה מקבלת את טווח הזמן בו רוצים לחפש את האובייקט, מערך המכיל את התאריך, השעה והמיקום בפריים של כל אחד מהאובייקטים המופיעים ברשימה של הקטגוריה בעץ אינדקס בטווח התאריכים הנ"ל, היא מקבלת גם את ה- id של הקטגוריה וכן את ראש הרשימה המקושרת שלה בעץ.

הפונקציה מבצעת מעבר על עץ ההסרטות, היא עוברת רק על הרמה התחתונה של העץ, על אלו השייכים לטווח הזמן הרצוי.

אם הפונקציה תקבל my_array שהוא none זה אומר שזו הפעם הראשונה שעורכים חיפוש של הקטגוריה הזו ולכן הפונקציה תעבור על כל ההסרטות החל מתאריך ההתחלה שהתקבל עד שיימצא האובייקט או עד תאריך סיום.

אחרת אם my_array אינו none, זה אומר שישנם אובייקטים בעץ אינדקס העונים על דרישות החיפוש ובוצע מעבר עליהם וכרגע יתבצע מעבר על עץ ההסרטות פחות הזמנים שהופיעו ברשימה.

עבור כל תאריך הנמצא ב my_array החיפוש יתבצע החל ממספר הפריים המופיע באותו אינדקס במערך.

```python
def iterate_folders(start_date, start_time, end_date, end_time, my_array, class_id, head_list):
    current_date = datetime.strptime(start_date, "%d-%m-%y")
    current_time = datetime.strptime(start_time, "%H-%M")
    end_date = datetime.strptime(end_date, "%d-%m-%y")
    end_time = datetime.strptime(end_time, "%H-%M")

    while True:
        numFrame = 0
        # בודקת אם הסרטון הופיע ברשימה
        if my_array:
            existing_item = next((item for item in my_array if item["date"] == current_date.strftime(
                "%d-%m-%y") and item["time"] == current_time.strftime("%H-%M")), None)
            if existing_item:
                numFrame = existing_item["numFrame"]
        folder_name = os.path.join(
            "01-05-2023_07-05-2023", current_date.strftime("%d-%m-%y"), current_time.strftime("%H-%M"))
        files = os.listdir(folder_name)
        for video_file in files:
            file_path = os.path.join(folder_name, video_file)
            Detecting_multiple_objects_in_a_video.video_detect(
                file_path, numFrame, class_id, head_list)
        current_time += timedelta(hours=1)
        if current_time.hour == 23:
            folder = os.path.join(
                "01-05-2023_07-05-2023", current_date.strftime("%d-%m-%y"), current_time.strftime("%H-%M"))
            video_files = os.listdir(folder)
            for video in video_files:
                file_path = os.path.join(folder, video)
                Detecting_multiple_objects_in_a_video.video_detect(
                    file_path, numFrame, class_id, head_list)
            current_date += timedelta(days=1)
            current_time = datetime.strptime("00:00", "%H:%M")

        if current_date > end_date or (current_date == end_date and current_time > end_time):
            break
```

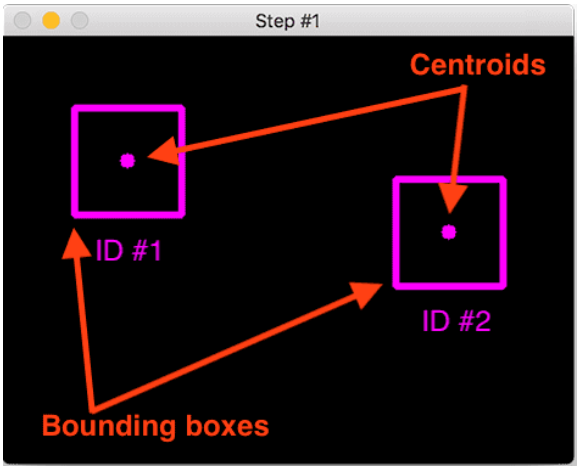
בשלב הבא הייתי צריכה למצוא בהסרטות את האובייקט שהמשתמש מחפש, תחילה חשבתי לבצע זיהוי אובייקטים בלבד על כל פריים עד שימצא האובייקט אבל דבר כזה לא יעיל כלל וכלל כי הסרטה מכילה מאות פריימים וכל אובייקט מופיע גם כן במספר מרובה של פריימים. הרי אובייקט יכול להופיע מספר שניות וידוע שישנם 30 פריימים בשנייה כך שאותו אובייקט היה מזוהה אינספור פעמים ויוצא שזה לא רלוונטי.

חשבתי על דרך אחרת יעילה ככל האפשר ועלה לי רעיון - לצורך כך יצרתי פונקציה המבצעת מעקב אחר אובייקטים.

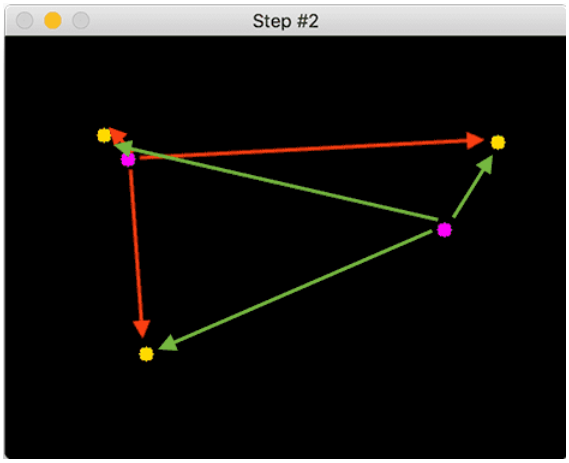
מעקב אובייקטים זה מסתמך על המרחק האוקלידי בין מוקדי אובייקט קיימים לבין מרכזי אובייקטים חדשים בין פריימים עוקבים בסרטון

עבור כל פריים חישבתי מרכזי אובייקטים על פי הקואורדינטות של תיבה תוחמת שהתקבלו מגלאי אובייקטים

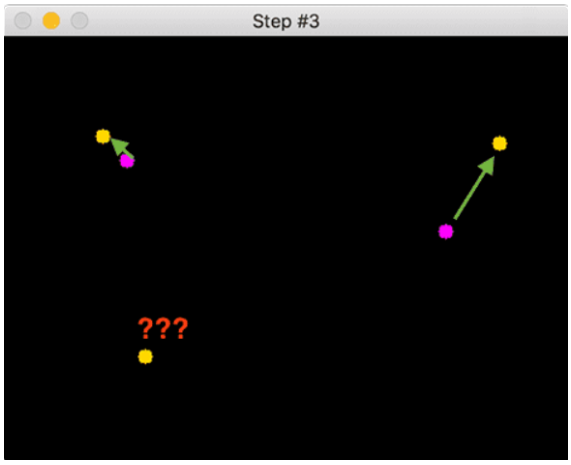
תחילה כשמדובר בסט הראשוני הראשון של תיבות תוחמות הקציתי להם מזהים ייחודיים



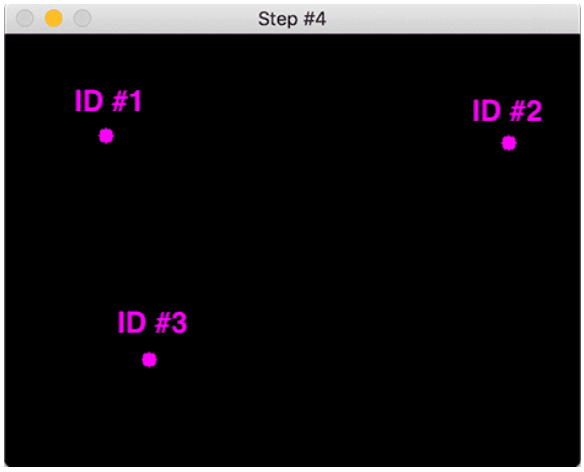
לאחר מכן חישבתי מרחקים אוקלידים בין תיבות תוחמות חדשות לאובייקטים קיימים.



ההנחה העיקרית של אלגוריתם המעקב שיצרתי היא שאובייקט נתון עשוי לעבור בין פריימים עוקבים, אך המרחק ב/ן המוקדים עבור פריימים F_t ו F_{t+1} יהיה קטן יותר מכל שאר המרחקים בין עצמים



ואז לאחר התאמה בין אובייקטים של הפריים הקודם לבין האובייקטים של הפריים הנוכחי אם יש אובייקט חדש שלא הותאם לאובייקט קיים צריך להוסיף אותו לרשימת האובייקטים במעקב שזה כולל הקצאת מזהה אובייקט חדש ואחסון המרכז של הקואורדינטות של התיבה התוחמת עבור אותו אובייקט.



בכל פעם שמזוהה אובייקט חדש הוא יתווסף לעץ אינדקס וכן יוצג למשתמש ויהיה עליו לאשר אם התכוון אליו. אם כן המערכת תעבור לבצע מעקב אחר האובייקט ואם לא אז המעקב אחר כלל האובייקטים ימשיך עד מציאת האובייקט הרצוי.

כמובן שהפונקציה דואגת לטפל גם במקרה שאובייקט אבד, נעלם או יצא משדה הראייה.

כאן התחשבתי בכך שגם אם אובייקט מופיע בפריים לא מחייב שתמיד גלאי האובייקטים מצליח לזהות אותו או שלפעמים יש משהו שאולי מסתיר את האובייקט לכמה שניות כמו למשל רכב חולף וכו'. אז קבעתי מספר של פריימים שגם אם אובייקט פתאום לא מופיע בהם ברציפות הוא עדיין נחשב למזוהה אם הוא נעלם כבר יותר מהמספר שנקבע הוא נמחק מרשימת האובייקטים הקיימים. כמו כן בכל פעם שאובייקט מזוהה סוכם הפריימים שלו שיכול לא להופיע מתאפס.

```
g = 0
center_points_prv_frame = []
tracking_objects = {}
tracking_cordinates = {}
track_id = 0
maxDisappeared = 13

# פרוק הסרטה לפריימים
def video_detect(filename, firstFrame, category, head_list):
    global center_points_prv_frame
    global tracking_objects
    global track_id
    global maxDisappeared

# (נכון לרגע זה אין אחת כזו טובה מספיק)  לשלב ספריית זיהוי פנים טובה
# (נכון לרגע זה אין אחת כזו טובה מספיק)  לשלב ספריית השוואת אובייקטים טובה

    # קריאה לקובץ הסרטה
    cap = cv2.VideoCapture(filename)

    # בדיקה אם הקובץ נפתח כראוי
    if not cap.isOpened():
        print("Error opening video file")
        exit()

    numFrame = 0
    num = 0

    # פתיחת לולאה לקריאת הפריימים ושמירתם
    while cap.isOpened() and numFrame < firstFrame:
        ret, frame = cap.read()
        numFrame += 1
```

```
    # פתיחת לולאה לקריאת הפריימים ושמירתם
    while cap.isOpened():
        ret, frame = cap.read()
        thisFrame = int(cap.get(cv2.CAP_PROP_POS_FRAMES))

        # בדיקה אם הפריים נקראו כראוי
        if not ret:
            break

        if num % 6 == 0:

            #  נקודות המסגרת הנוכחית
            center_points_cur_frame = []

            # המרת הפריים לגווני אפור
            cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

            (class_ids, scores, bboxes) = model.detect(frame)

            # התיבה האמיתית עבור כל אוביקט
            final_boxes = One_bounding_box.one_box(
                bboxes, scores, frame.shape[0], frame.shape[1])

            for i in final_boxes:
                if (class_ids[i] == category):
                    x1, y1, w, h = bboxes[i]
                    cx = int((x1+x1 + w) / 2)
                    cy = int((y1 + y1 + h) / 2)
                    center_points_cur_frame.append(
                        {"center_point": (cx, cy), "box": (x1, y1, w, h), "disappeared": 0})
```

```
        if num <= 12:
            for pt in center_points_cur_frame:
                for pt2 in center_points_prv_frame:
                    distance = math.hypot(
                        pt2["center_point"][0] - pt["center_point"][0], pt2["center_point"][1] - pt["center_point"][1])
                    if distance < 600:
                        tracking_objects[track_id] = pt
                        track_id += 1
                        box = pt["box"][0], pt["box"][1], pt["box"][0] + \
                            pt["box"][2], pt["box"][1]+pt["box"][3]

                        if firstFrame == 0:
                            # ברגע שמוצא אוביקט מעדכן ברשימה בעץ נתיב ההסרטה ומספר פריים
                            newObject = LinkedListNode(
                                filename[22:30], filename[31:36], thisFrame, bboxes[i])
                            TST_tree_LinkedListNode.add_sorted(
                                head_list, newObject)
                            found = Search_in_an_index.display_specific_frame(
                                "\\".join(filename.split("\\")[:-1]), thisFrame, bboxes[i])
                            if found:
                                # נפעיל מעקב אחר האובייקט לגילוי תנועה שלו
                                Strack.objectTrack(filename,thisFrame,bboxes[i])
```

```
        else:
            tracking_objects_copy = tracking_objects.copy()
            center_points_cur_frame_copy = center_points_cur_frame.copy()
            for object_id, pt2 in tracking_objects_copy.items():
                object_exists = False
                for pt in center_points_cur_frame_copy:
                    distance = math.hypot(
                        pt2["center_point"][0] - pt["center_point"][0], pt2["center_point"][1] - pt["center_point"][1])

                    # עדכון מיקום מזהים
                    if distance < 600:
                        tracking_objects[object_id] = pt
                        object_exists = True
                        if pt in center_points_cur_frame:
                            center_points_cur_frame.remove(pt)
                        continue

                # הסר תעודות זהות שאבדו
                if not object_exists:
                    pt2["disappeared"] += 1
                    if pt2["disappeared"] > maxDisappeared:
                        tracking_objects.pop(object_id)

            # הוסף מזהים חדשים שנמצאו
            for pt in center_points_cur_frame:
                tracking_objects[track_id] = pt
                track_id += 1
```

```
                # ברגע שמזוהה אוביקט מעדכן ברשימה בעץ נתיב ההסרטה ומספר פריים
                newObject = LinkedListNode(
                    filename[22:30], filename[31:36], thisFrame, bboxes[i])
                TST_tree_LinkedListNode.add_sorted(head_list, newObject)

                # להציג לאישור המשתמש
                found = Search_in_an_index.display_specific_frame(
                    "\\".join(filename.split("\\")[:-1]), thisFrame, bboxes[i])
                if found:
                    # נפעיל מעקב אחר האובייקט לגילוי תנועה שלו
                    Strack.objectTrack(filename,thisFrame,bboxes[i])

        # צור העתק של הנקודות
        center_points_prv_frame = center_points_cur_frame.copy()
    num += 1

cap.release()
print("Done!")
```

הפונקציה הבאה מקבלת נתיב לקובץ הסרטה, מספר פריים ומיקום על הפריים ומבצעת מעקב אחר האובייקט המופיע במיקום הזה על הפריים בהסרטה, בודקת ממתי האובייקט התחיל לזוז ומתי הוא כבר לא נראה ואז שולחת לפונקציה שתחתוך את הקטע הזה מההסרטה.

```
def objectTrack(filename, thisFrame, bbox):
    cap = cv2.VideoCapture(filename)
    frame_counter = 0
    object_start_frame = -1
    object_end_frame = -1

    for _ in range(thisFrame):
        ret, frame = cap.read()
        if not ret:
            print("Failed to read video")
            break
        frame_counter += 1

    start = False
    x, y, w, h = bbox
    prev_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    prev_center = np.array([x + w / 2, y + h / 2], dtype=np.float32)

    lk_params = dict(winSize=(15, 15),
                     maxLevel=2,
                     criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
    while True:
        ret, frame = cap.read()

        if not ret:
            break
        frame_counter += 1

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        next_pts, status, _ = cv2.calcOpticalFlowPyrLK(prev_gray, gray, np.array([prev_center]), None, **lk_params)

        if start and object_start_frame == -1 and is_object_moving(next_pts, prev_center):
            object_start_frame = frame_counter
```

```
        height, width, channels = frame.shape
        if start and is_object_out_of_bounds(next_pts, width, height):
            object_end_frame = frame_counter
            break

        prev_center = next_pts[0].flatten()

        center = tuple(map(int, prev_center))

        if cv2.waitKey(1) == 27:
            break
        start = True

        prev_gray = gray.copy()

    frame_rate = 30
    object_start_time = round(object_start_frame / frame_rate)
    object_end_time = round(object_end_frame / frame_rate)

    cut_video(filename, "CutVideo.mp4", object_start_time, object_end_time)

    search1 = search_results()
    search1.display_screen("02-03-02", "10:00", "02", "01-02", "10:10", "03", "CutVideo.mp4")

    cap.release()
    cv2.destroyAllWindows()
```

אלה פונקציות שהפונקציה objectTrack משתמשת.

```
def is_object_moving(next_pts, prev_center):
    distance = np.linalg.norm(next_pts[0] - prev_center, ord=2)
    print(distance)
    if distance > 4:
        return True
    else:
        return False

def is_object_out_of_bounds(pts, frame_width, frame_height, padding=10):
    next_pts = pts[0] + padding
    return (
        next_pts[0] < 0 or
        next_pts[0] >= frame_width or
        next_pts[1] < 0 or
        next_pts[1] >= frame_height
    )
```

הפונקציה הבאה חותכת קטע מהסרטה אשר בו בעצם נראה האובייקט בתנועה

```
from moviepy.video.io.ffmpeg_tools import ffmpeg_extract_subclip

def cut_video(input_file, output_file, start_time, end_time):
    ffmpeg_extract_subclip(input_file, start_time, end_time, targetname=output_file)
```

23 מדריך למשתמש

בכניסה למערכת ישנם שני כפתורים לבחירה, הראשון " חיפוש חדש" והשני "צפיה בחיפושים קודמים" .

במידה והמשתמש בחר בחיפוש חדש הוא יעבור למסך אחר ובו יהיה עליו להכניס את כל הפרטים הנחוצים למערכת כדי לבצע את החיפוש.

הפרטים כוללים:

בחירה בסוג המצלמה ובמספרה מבין המצלמות הקיימות אצלו.

טווח הזמן בו יחפשו את האובייקט (תאריך ושעה)

העלאת תמונה של האובייקט לחיפוש.

לאחר הכנסת כל הפרטים עליו ללחוץ על כפתור "התחל חיפוש"

כעת המערכת תתחיל לבצע את החיפוש, במהלך עבודתה היא תציג למשתמש תמונות של אובייקטים העונים על הדרישות ויכולים להיות אולי האובייקט המבוקש. על המשתמש יהיה לאשר אם התכוון אל האובייקט המוצג באמצעות אחד מהכפתורים – כן או לא שיוצגו על המסך.

לבסוף המערכת תביא למשתמש קטע חתוך מההסרטות בו נמצא האובייקט המבוקש בתזוזה , תהיה גם אפשרות למשתמש לבחור באופציה של שמירת הסרטון החתוך.

אם הסרטון המוחזר לא סיפק את המשתמש המערכת תמשיך במעקב אחר האובייקט.

במידה ובמסך הראשי המשתמש בחר בצפייה בחיפושים קודמים הוא יעבור למסך ששם יופיעו כל פרטי החיפושים שהתרחשו הכוללים את: התאריך בו בוצע החיפוש, האובייקט, המצלמה, טווח הזמן שהתבקש, וסרטון התוצאה הסופית.

24 בדיקות והערכה

על מנת לבדוק את תקינות המערכת ולוודא שאכן המודל שבחרתי להשתמש פועל כמצופה, ניסיתי אותו על עשרות תמונות / הסרטות של מקומות שצולמו במרחקים שונים מהמצלמה ומזוויות שונות.

המודל סיפק את התוצאות הרצויות עבר כל תמונה וסיווג כראוי את האובייקטים המופיעים בה למעט חריגות.

כמו כן לאחר הרצת האלגוריתם כולו נבחנו כל האילוצים שדרושים כדי להביא למעבר יעיל ומהיר ככל האפשר על עץ הסרטות לחיפוש אובייקט ותנועתו.

כאשר הופיעו טעויות ובאגים בביצוע של האלגוריתם, נבדק הקוד שוב עד שתוקנו הבעיות.

לאחר בדיקות רבות אחר כל מקרי הקצה שעלו בדעתי, והרצת האלגוריתם מספר פעמים על נתונים שונים, האלגוריתם הגיע לקירוב האפשרי ביותר בכלים העומדים לרשותי.

25 מסקנות

בעת שניגשתי לתכנן את פרויקט הגמר,ראיתי כי המשימה מורכבת ודורשת השקעה רבה הן בלימוד החומר קודם תחילת בנייתו, הן בתכנון בנייתו, תכנון האלגוריתם והן בבניה עצמה. כתיבת הקוד הייתה מורכבת ומאתגרת אך בעיקר מלמדת.למדתי להשתמש בספריות Python רבות ומתקדמות, לכתוב קוד בסטנדרטים גבוהים, רכשתי יכולת קידוד גבוהה ב – Python.

ראשית השקעתי שעות רבות בחקר הנושא לעומקו, ובלמידה עצמית של נושאים שונים כמו גם בבינה
מלאכותית ורשתות נוירונים שהשתמשתי בכדי שלא יהוו בשבילי כ 'קופסה שחורה'.
כל זאת בד בבד עם תכנון האלגוריתם, וחלוקה הגיונית לפונקציות.
לאחר מכן, ניגשתי לכתיבת הקוד. היה עלי לכתוב אותו תוך חשיבה מעמיקה ולימוד מקיף של התחום.

במהלך הפרויקט רכשתי ידע רב בנושאים הבאים:

- ניסיון בתחום הרשתות נוירונים, Machine Learning
- ניסיון רב בקידוד ב - Python
- שימוש בטכנולוגית tkinter וספריות נוספות ב - Python
- פיתוח יכולת למידה עצמית גבוהה וחשיבה לוגית.

לאחר שעות רבות מספור של עמל, טורח והשקעה. כדי להגיע לרמה תכנותית מקצועית ומדויקת אני חשה סיפוק רב, ויודעת כי הפקתי תועלת רבה מפיתוח הפרויקט.

ההשוואה בין התכנון הראשוני לתוצר הסופי, מגלה כי התוכנה עומדת בדרישות וזהה לתכנון. הגם שבמהלך הפרויקט עלו שאלות רבות לגבי ביצוע טכני ותכנותי בפרויקט, דברים מסוימים נראו כבלתי אפשריים, בסופו של דבר התגברתי עליהם בדרכים יצירתיות ומגוונות. במבט לאחור, הפרויקט היווה עבורי התנסות בהתמודדות עם פרויקט בסדר גודל , כתיבת אלגוריתם מורכב שחידד והחכים.

ניתן לומר כי הפרויקט תרם לי רבות כסטודנטית וכמהנדסת.

26 פיתוחים עתידיים

בגרסה הבאה נדאג למצוא ספריית זיהוי פנים טובה וספרית השוואת אובייקטים טובה מספיק.

במידה ותתרחב רשימת הקטגוריות שמכיר המודל YOLO נדאג לעדכן זאת.

אם הזמן היה עומד ברשותי הייתי עומלת וטורחת על דברים נוספים כגון:

הגנה נוספת על קבצי תוצאות החיפוש הנשמרים בדיסק המקומי מפני מחיקה והגדרת הקובץ כקובץ מערכת מוגן.

27 בבליוגרפיה

לצורך יצירת פרויקט זה, ביצעתי חקר לעומק של הנושא, והתבצעה הקפה של כמה שיותר חומרים ומידע על מנת שהפרויקט יהיה כמה שיותר מקצועי.
להלן רשימת אתרים וספרות בהם נעזרתי:

מודל זיהוי אובייקטים

https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-/object-detection

/https://wellsr.com/python/object-detection-from-images-with-yolo

https://dontrepeatyourself.org/post/yolov4-custom-object-detection-with-opencv-/and-python

/https://blog.roboflow.com/coco-dataset

https://towardsdatascience.com/how-to-work-with-object-detection-datasets-in-coco-format-9bf4fb5848a4

משם הסברתי https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31

זיהוי תמונה

/https://learnopencv.com/image-recognition-and-object-detection-part1

זיהוי אובייקטים

https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-/detection

/https://machinelearningmastery.com/object-recognition-with-deep-learning

מעקב אחר אובייקטים

/https://blog.roboflow.com/what-is-object-tracking-computer-vision

/https://viso.ai/deep-learning/object-tracking

/https://www.geeksforgeeks.org/track-objects-with-camshift-using-opencv

/https://pysource.com/2021/11/02/kalman-filter-predict-the-trajectory-of-an-object

תנועת אובייקטים

https://www.hamichlol.org.il/%D7%92%D7%99%D7%9C%D7%95%D7%99_%D7%AA%D7%A0%D7%95%D7%A2%D7%94_%D7%91%D7%95%D7%95%D7%99%D7%93%D7%90%D7%95

/https://pyimagesearch.com/2015/09/21/opencv-track-object-movement

השוואת תמונות

https://stackoverflow.com/questions/42499927/how-to-compare-if-two-images-representing-the-same-object-if-the-pictures-of-the

https://www.baeldung.com/cs/image-comparison-algorithm

פיתון

/https://packaging.python.org/en/latest/tutorials/installing-packages