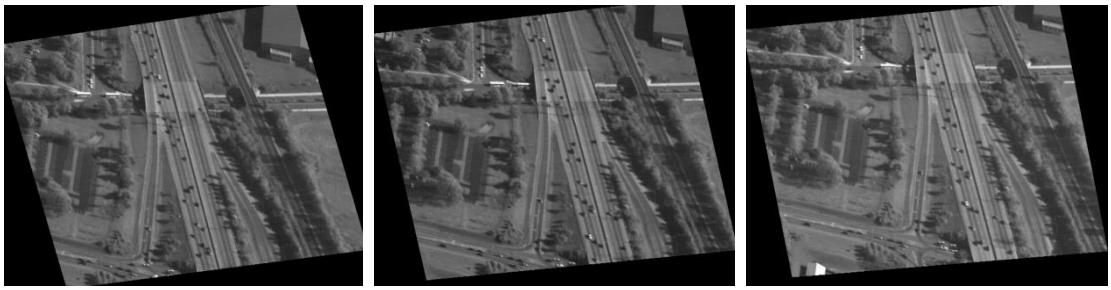
הצגת הבעיה

הבעיה המוצגת בפרויקט הינה עקיבה אחר אובייקטים ובפרט, עקיבה אחר אובייקטים מרובים.

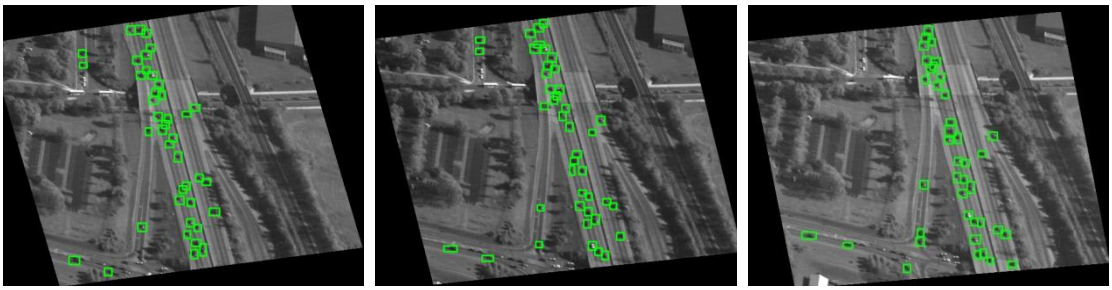
הבעיה של עקיבה אחר אובייקטים הינה משימה חשובה בתחום הראיה הממוחשבת. השגשוג של מחשבים בעלי כח חישוב חזק, הזמינות של מצלמות וידאו באיכות גבוהה והצורך העולה בניתוח וידאו אוטומטי יצרו עניין רב באלגוריתמים לעקיבה אחר אובייקטים.
פתרון לבעיה זו יתרום לישומים רבים כגון מעקב ואינטראקצית מחשב-אדם, ויספק מידע הכרחי למשימות כגון זיהוי וניתוח מאורעות והתנהגויות.

בפרויקט התעסקנו בתחום של עקיבה אחר אובייקטים רבים, כאשר המטרה היא לזהות את האובייקטים הנעים וליצור מסלולים של האובייקטים השונים על ידי זיהוי הקשר בין האובייקטים בפריימים השונים.
בהינתן סרט וידאו אותו מפרידים לפריימים, או פריימים רציפים בזמן ובמרווחים קבועים (כפי שניתן לראות באיור 1), יש לזהות את האובייקטים הנעים בכל תמונה (איור 2), ולאחר מכן לקשר בין האובייקטים כדי ליצור מסלול עבור כל אחד מהם. כלומר, יש לשייך כל אובייקט למסלול אשר מורכב מאובייקטים בפריימים האחרים (איור 3).

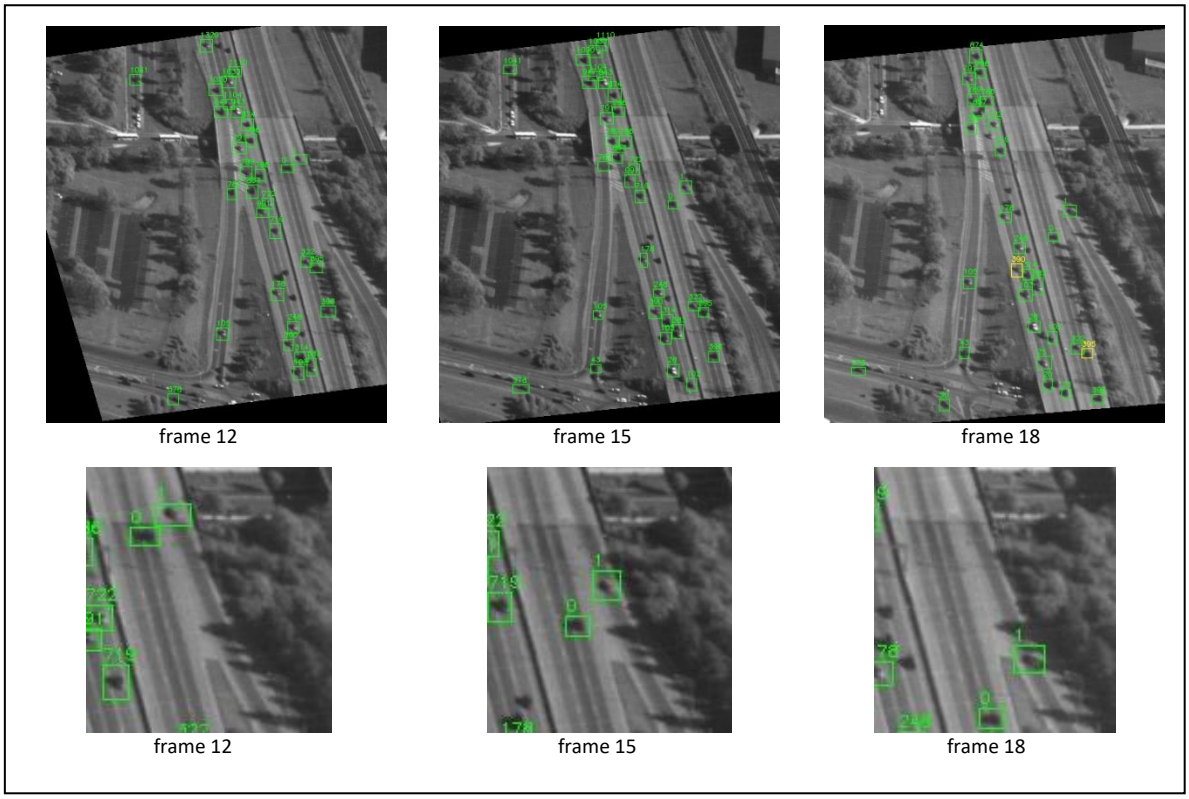
בנוסף לאתגרים הרגילים בעקיבה אחר אובייקטים, בתחום זה ישנם אתגרים נוספים הנובעים מריבוי האובייקטים. למשל שיוך האובייקטים כאשר מספרם אינו ידוע, ריבוי האפשרויות למיפוי בין אובייקטים בפריימים שונים, הקשר בין אובייקטים באותו פריים ועוד. במילים אחרות, גילוי מסויים יכול להתאים לאובייקטים רבים, או שגילויים רבים יכולים להתאים לאובייקט כלשהו.
הדבר מציב בעיה אשר מתעסקת במידע רב ולכן קשה ויקר לפתור אותה בצורה אופטימאלית.



איור 1: הקלט.



איור 2: זיהוי האובייקטים.



איור 3: שיוך האובייקטים למסלולים.

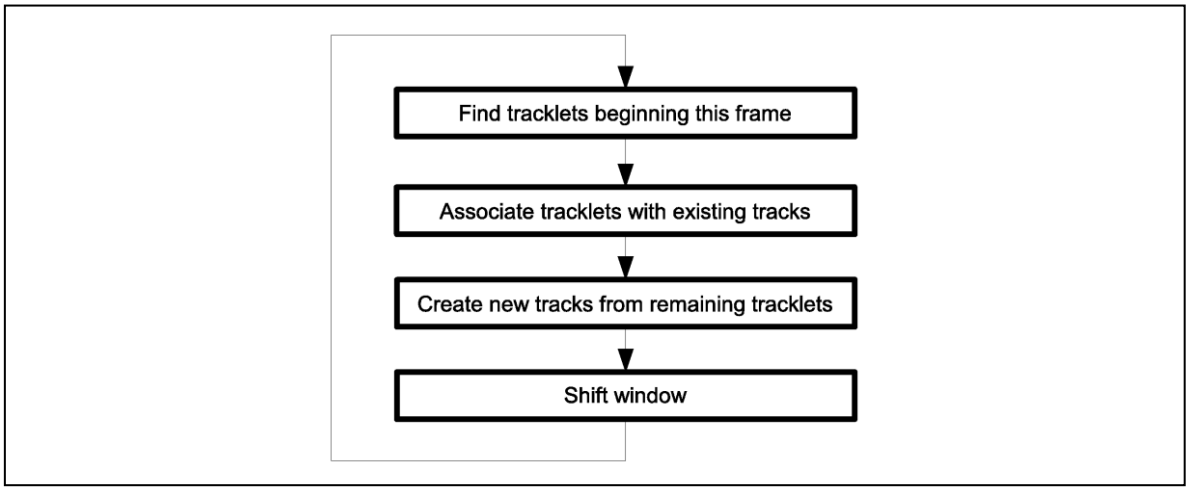
מטרת הפרוייקט

אם ננסה להבין מהו אופי הפרוייקט שביצענו, אז ניתן לאמר שהפרוייקט הוא פרוייקט בעל אופי מחקרי. כלומר, קיים אלגוריתם מסויים, ועלינו מוטלת המשימה לחקור את האלגוריתם, לנתח תוצאות, לשפר במידת הצורך, וכו'.
האלגוריתם המדובר הינו אלגוריתם לזיהוי אובייקטים מרובים, ובניית מסלולים.
למעשה, כשהתחלנו את הפרוייקט לא הייתה מטרה ספציפית כמו "יש להגיע לתוצאה X" או "יש ליצור תוכנית Y", אלא המטרה נבנתה במהלך העבודה על הפרוייקט.
הסיבה לכך היא שבתחילת הפרוייקט לא ידענו מהי נקודת ההתחלה שלנו. לכן, אופי העבודה על הפרוייקט היה כך שבכל שלב נפגשנו עם ד"ר ישי קמון (המנחה), והגדרנו את מטרת הביניים הבאה בהתאם למה שיש בידינו עד השלב הנוכחי.
ואם נסכם, מטרת הפרוייקט הייתה:
להבין את האלגוריתם שאיתו נעבוד - האלגוריתם לזיהוי אובייקטים מרובים ויצירת מסלולים.
להשיג קוד שמממש את אותו אלגוריתם, ולהעריך את ביצועיו.
לחקור את האלגוריתם, ולנסות לשפר את ביצועיו ככל שנוכל (ובהתאם למגבלות כמו כמות datasets שיש בידינו).

האלגוריתם

<u>קלט</u>: רשימת גילויים (אובייקטים) בכל frame.

<u>תקציר</u>:
האלגוריתם משתמש בטכניקת sliding window, כאשר בכל חלון נבנים מסלולונים שמתחילים במסגרת הראשונה בחלון, וגודל כל מסלולון הוא לכל היותר גודל החלון.
בסיום כל איטרציה מאחדים את המסלולונים החדשים עם מסלולים שנבנו באיטרציות קודמות – המסלולונים הנותרים הופכים למסלולים.
מזיזים את החלון וחוזרים על התהליך.



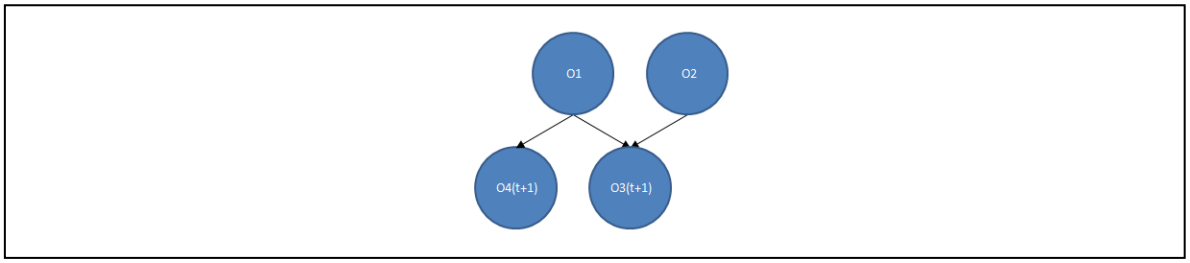
איור 4: טכניקת sliding window.

ניתן לחלק את פעולת האלגוריתם בכל איטרציה, ל-4 שלבים עיקריים:
1. יצירת עץ גילויים
2. חלוקת הגילויים בעץ לגילויים אמיתיים, ולגילויי סרק (valid/invalid)
3. בניית מסלולונים
4. איחוד מסלולונים עם מסלולים שנוצרו קודם

<u>תיאור האלגוריתם:</u>

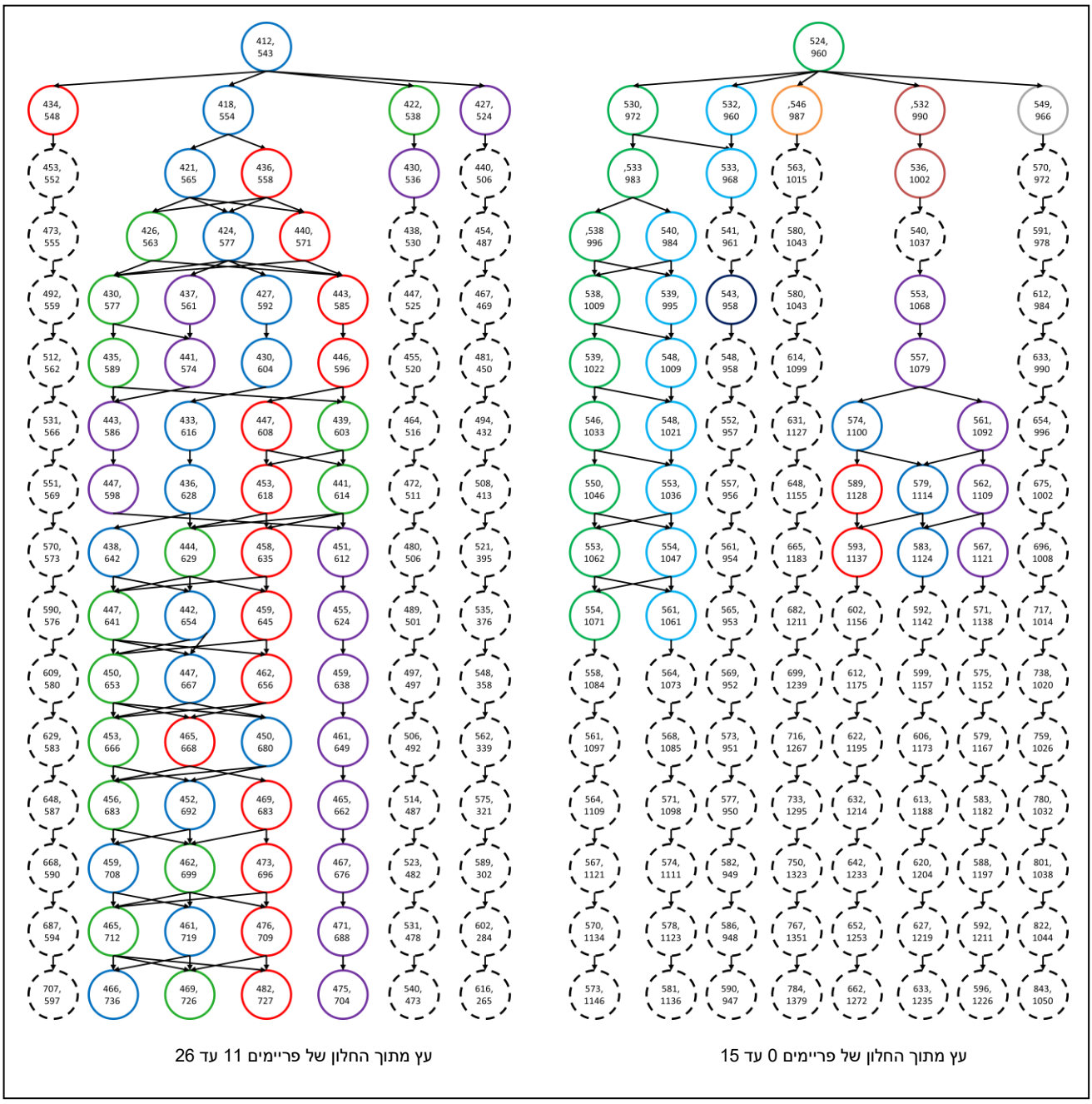
שלב א': יצירת עץ גילויים
בהינתן חלון של מסגרות בגודל T (T מסגרות בחלון), נרצה לבנות עץ גילויים עבור כל אובייקט שהתגלה במסגרת הראשונה בחלון.
נסמן אובייקט מסויים שהתגלה במסגרת t+1 ב-O(t+1), ואובייקט אחר שהתגלה במסגרת t ב-O(t), אז עץ הגילויים נבנה כך ש-O(t+1) הוא בן של O(t) אם שני האובייקטים "קרובים" אחד לשני. כלומר עבור אובייקט O(t) יהיו מספר כלשהו של בנים כאשר כולם התגלו במסגרת t+1 וכולם "קרובים" ל-O(t).
לכן, אם במסגרת הראשונה בחלון נתגלו K אובייקטים – באיטרציה הנוכחית יווצרו K עצי גילויים.
ההגדרה של "קרובים" היא תלויית מימוש, אך במקרה שהאלגוריתם פועל על תצלום אוויר, המשמעות היא למשל מרחק במטרים (או יותר נכון, מרחק בפיקסלים בהינתן יחס מטר לפיקסל).

דוגמא: באיור 5 רואים 2 עצי גילויים, עץ ראשון בעל שורש O1(t) עם שני בנים, O3, O4. ועץ נוסף עם שורש O2 ובן O4.
אין סתירה בכך ש-O3 נמצא ב-2 עצי גילויים, שכן עדיין לא החלטנו לאיזה מסלולון הוא שייך.



איור 5.

באיור 6 ניתן לראות דוגמא לעצי גילויים שהתקבל מריצת התוכנית.



איור 6: עצי גילויים שהתקבל מריצת התוכנית. המספרים בתוך הצמתים הינם קורדינטות של הגילויים. צמתים מקווקווים מסמנים גילויים וירטואלים. הגילויים צבועים בצבעים שונים על פי שיוכם למסלול אמיתי, לדוגמא, בעץ הימני, כל הצמתים הירוקים שייכים למסלול אמיתי אחד.

שלב ב': חלוקת הגילויים בעץ לגילויים אמיתיים, ולגילויי סרק (valid/invalid)
בשלב הקודם ייצרנו מספר עצי גילויים שיהוו את הבסיס לבניית מסלולונים. מספר המסלולונים שנובע ישירות מבניית העצים הוא עצום, וכמו כן יש להתחשב בעובדה שחלק מהגילויים הם גילויי סרק.
כשאנו מדברים על גילויי סרק הכוונה היא למשל לגילויים שקיבלנו כתוצאה מרעש, או גילויים שהם בנים של צומת בעץ <u>שלא אותו צומת יצר אותם</u>.
המטרה בשלב זה, היא למצוא את החלוקה הטובה ביותר של כלל הגילויים לאמיתיים וסרק.
האמצעי למציאת המטרה הוא אלגוריתם סטטיסטי בשם MAP inference (Maximum a posteriori estimation).
על מנת שנוכל למצוא את החלוקה בעזרת MAP, יש להתאים את הבעיה לאלגוריתם:
עבור גילוי i שהתגלה במסגרת t נסמן y_i^t בתור משתנה בינארי שמסמל האם הגילוי הוא אמיתי או סרק, ונסמן o_i^t את תכונות אותו גילוי (מיקום, מראה, וכו').
נסמן ב- y את כלל הגילויים בחלון, וב- o את כלל התכונות של הגילויים.
לכן ניתן להגדיר את בעיית החלוקה כ:

$$\arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{o})$$

נותר להגדיר את פונקציית ההסתברות, וכך נפתור את בעיית החלוקה:

$$p(\mathbf{y},\mathbf{o}) = p(y^0) \prod_{\substack{i,j,t>0 \\ y_i^t \text{near } y_j^{t-1}}} p(y_i^t|y_j^{t-1}) \prod_{i,t>0} p(\mathbf{o}_i^t|y_i^t)$$

כאשר $p(y^0) = 1$ ומציין את העובדה שהאלגוריתם מזהה את הגילויים במסגרת הראשונה בחלון כאמיתיים בוודאות. זאת לא בעיה שכן, אם חלק מגילויים אלה הם גילויי סרק, אז גילינו זאת באיטרציה קודמת, וגילויים אלו ירדו בשלב ג' של בניית המסלולונים.
$p(o_i^t|y_i^t)$ מוגדרת כ"כמה הגילוי הנוכחי דומה לשורש העץ שהוא שייך לו", ובצורה מדוייקת יותר:

	$y_i^t = 0$	$y_i^t = 1$
$p(\mathbf{o}_i^t\|y_i^t)$	$1 - a(\mathbf{o}_i^t, \mathbf{o}^0)$	$a(\mathbf{o}_i^t, \mathbf{o}^0)$

הפונקציה $a(o_i^t, o^0)$ היא פונקציית דמיון כלשהי, וניתן להשתמש בכל פונקציה שתחזיר ערך בין 0 ל-1.
באופן דומה מגדירים את $p(y_i^t|y_j^{t-1})$ כמדד הדמיון בין גילוי לבין האב שלו בעץ:

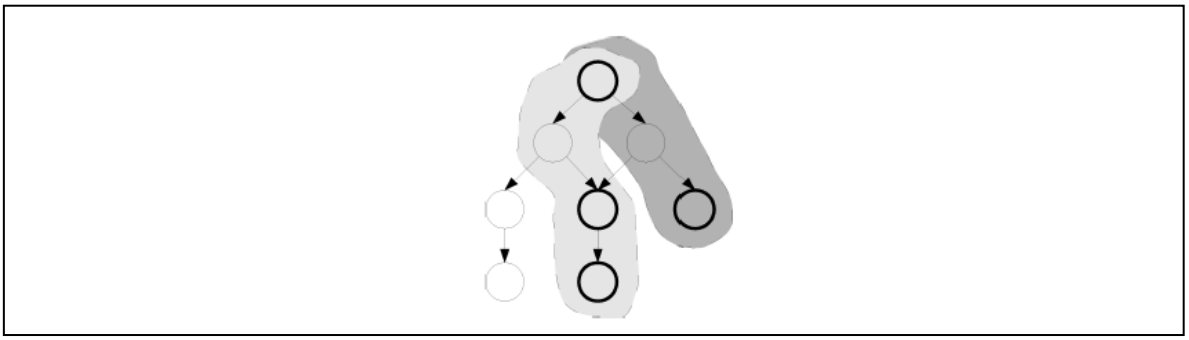
	$y_i^t = 0$	$y_i^t = 1$
$y_j^{t-1} = 0$	0.5	0.5
$y_j^{t-1} = 1$	$1 - a(\mathbf{o}_i^t, \mathbf{o}_j^{t-1})m(\mathbf{o}_i^t)$	$a(\mathbf{o}_i^t, \mathbf{o}_j^{t-1})m(\mathbf{o}_i^t)$

נשים לב שכעת ישנו חלק נוסף במשוואה: $m(o_i^t)$
פונקציה זו משמעותה היא כמה התנועה של הגילוי הנוכחי היא "הגיונית" (motion likelihood), מכיוון שלכל גילוי יכולים להיות מספר אבות באותו עץ גילויים – בוחנים את המסלול דרך האבות שדרכם התנועה היא ההגיונית ביותר (הפונקציה מקבלת ערך מקסימלי).
לכן נגדיר motion parent בתור האב שמוביל למסלול ההגיוני ביותר.
על מנת למדוד האם התנועה הגיונית, האלגוריתם משתמש במודל linear Gaussian, ולא נרחיב עליו (-:

כעת, ניתן לפתור את בעיית החלוקה – ובסיום שלב זה כל צומת בכל עץ גילויים מסומן כאמיתי או סרק.

שלב ג': בניית מסלולונים
כעת בידינו מספר עצי גילויים כאשר כל צומת מסווג כאמיתי או סרק.
ראשית יש למצוא את כל המסלולונים האפשריים, על ידי טיפוס בכל צומת בעץ עד לשורש **דרך ה- motion parent** שהגדרנו קודם לכן, כאשר מורידים מסלולונים שהם prefix של מסלול אחר.

נשים לב שמרחב החיפוש שלנו הצטמצם משמעותית, שכן יש לבדוק רק את הצמתים ה"אמיתיים".



איור 7.

סינון נוסף של המסלולונים מתבצע על ידי מספר קריטריונים כמו:

- תאוצה הגיונית (מרחק בין אובייקטים סמוכים במסלול)
- אורך מסלול – לפחות חצי מגודל החלון
- תנועה "חלקה" (וריאציה על ממוצע של $m(o_i^t)$) – נשים לב שאם קיים מסלולון של אובייקט שמבצע למשל פנייה ב-90 מעלות, אם התנועה עד לפנייה ואחריה תהיה חלקה, אז ה"תקלה" שפוגעת בתנועה החלקה מתמוססת לעמות הערך הממוצע.

השלב השני הוא מיזוג מסלולונים דומים, שנוצרים כתוצאה מאפקטים של split-merge של אובייקטים.
בשלב זה עוברים על כל המסלולונים, אם שני מסלולונים נחשבים ל"זהים" – נמחק את המסלולון הקצר יותר.
מסלולונים "זהים" הם 2 מסלולונים, אשר ערך הפונקציה $sim(t1,t2)$ גדול מרף מסויים.
הפונקציה $sim(t1,t2)$ מוגדרת על ידי:

$$
\begin{aligned}
sim(\tau_1,\tau_2) &= \frac{1}{T}\sum_{1}^{T} sim(\tau_1,\tau_2,t) \\
sim(\tau_1,\tau_2,t) &= a(\tau_1^t,\tau_2^t) * p(\mathbf{x}_1^t,\mathbf{x}_2^t) * v(\mathbf{v}_1^t,\mathbf{v}_2^t) \\
p(\mathbf{x}_1,\mathbf{x}_2) &= \exp(-\|\mathbf{x}_1-\mathbf{x}_2\|/c) \\
v(\mathbf{v}_1,\mathbf{v}_2) &= \exp(-\|\mathbf{v}_1-\mathbf{v}_2\|/c)\ ,
\end{aligned}
$$

בפשטות, נסביר שהפונקציה מתארת דמיון בין שני מסלולים מבחינת מראה (a()), מרחק (p()), ומהירות (v()).

שלב ג'1: טיפול באובייקטים חסומים (occlusion handling)

עד כה, התעלמנו ממקרים בהם חסרים גילויים של מסלול מסויים, למשל כתוצאה מחסימת אובייקט.
במקרה והאובייקט <u>החוסם</u> מתגלה – האלגוריתם יפעל כשורה שכן עץ הגילויים אינו משתנה למעט העובדה שהגילויים החסומים מסווגים כגילויי סרק. כמובן שבמידה והאובייקט חסום במשך יותר מחצי גודל חלון – האלגוריתם לא ימצא את המסלולון.
לכן, הבעיה המשמעותית יותר היא כאשר האובייקט החוסם לא התגלה.
במקרה זה, עומקו של עץ הגילויים יהיה קטן מ-|T|(גודל חלון), ואם המסלולון קצר מדי – האלגוריתם ימחק אותו.
הפיתרון הוא, כאשר לאובייקט מסוים <u>אין גילויים קרובים במסגרת הבאה</u>, נחשב היכן סביר שיהיה הגילוי הבא ונוסיף את הגילוי המשוערך לעץ הגילויים. גילויים כאלה נקראים "גילויים וירטואלים".
הגילוי הוירטואלי מקבל את אותו ערך של פונקצית הנראות(סומנה קודם בתור a(), בעת חישוב ההסתברויות בשלב החלוקה).

שלב ד': איחוד מסלולונים עם מסלולים שנוצרו קודם

הרעיון הוא לתת מענה לאיחוד מסלולונים עם מסלולים, תוך התחשבות במקרים של many-many mapping.
למשל, כאשר שני אובייקטים מתמזגים(2 מכוניות קרובות מאד אחת אל השנייה) ונשארים כך לאורך כל החלון – נוצר מסלולון יחיד, ויש לאחד אותו עם 2 מסלולים שמצאנו באיטרציה קודמות.
דוגמא נוספת, כשאובייקט מתפצל לשני אובייקטים ונשאר כך לאורך כל החלון – נוצרים 2 מסלולונים (בהנחה ולא מתמזגים ליחיד) ויש לאחד כל אחד מהם עם המסלול הקודם (וליצור 2 מסלולים חדשים).
השיטה שבה האלגוריתם משתמש, דומה מאד למה שעשינו בשלב ג':
בודקים עבור כל מסלולון וכל מסלול שמצאנו קודם, האם ערך פונקציית דמיון כלשהי גבוה מרף מסוים – אם כן, אז ניצור מסלול חדש.
פונקצית הדמיון בין מסלולים מוגדרת על ידי:

$$sim2(\tau_1, \tau_2) = \frac{1}{T}\sum_{1}^{T} sim2(\tau_1, \tau_2, t)$$

$$sim2(\tau_1, \tau_2, t) = a(\tau_1^t, \tau_2^t) * ps(\mathrm{x}_1^t, \mathrm{x}_2^t)$$

$$ps(\mathrm{x}_1, \mathrm{x}_2) = \exp(-\|\mathrm{x}_1 - \mathrm{x}_2\|/c)\ .$$

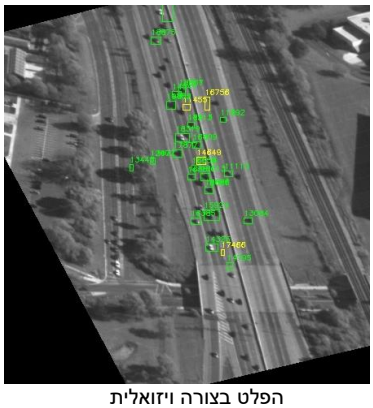
בדומה לשלב ג' הפונקציה ()ps נותנת מדד למרחק בין 2 אובייקטים, אולם כעת בהכרח חייב להיות לפחות אוביקט אחד שקיים במסלולון ולא קיים במסלול (כי החלון זז), ולהיפך – לכן במקרים כאלה משערים היכן סביר שהאובייקט החסר אמור להיות (בהתאם לתנועה).

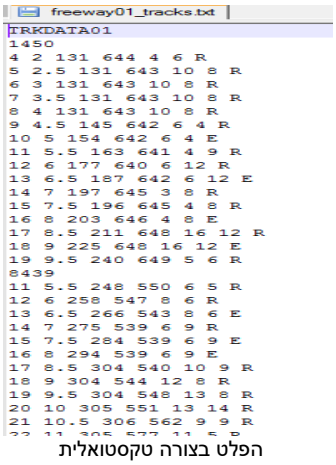
מה עשינו בפרויקט

1. הרצה ראשונה

כאשר התחלנו את הפרויקט, המטרה הראשונית הייתה להבין את האלגוריתם במאמר, להשיג קוד שהכותבים מימשו, ולקמפל ולהריץ את הקוד.
במאמר נאמר כי הקוד קיים באתר של מחברי המאמר. לצערנו, באתר היו רק קבצי הרצה אך לא היה קוד. בנקודה זו, תוך התייעצות עם המנחה, יצרנו קשר עם אחד ממחברי המאמר בבקשה לקבל את הקוד, והוא מצידו ענה שהקוד לא באתר כי הוא וחבריו עדיין לא פרסמו גרסה להורדה. לאחר אינספור תזכורות ונידנודים מחבר המאמר שלח לנו גרסה לשימוש פנימי של הקוד. הוא כמובן הזהיר אותנו שהקוד אינו מתועד, מבולגן ודורש מאמץ כדי לקמפל ולהריץ.
במקביל לנסיונות להשגת הקוד, התחלנו לשחק עם הקבצים הבינארים כדי ללמוד את אופן הפעלת התוכנה.
לאחר קבלת הקוד עבדנו על קימפול הקוד תוך לימוד עצמי של CMake. כדי לקמפל את הקוד נדרשו תוכנות, ספריות וקבצים נוספים. מכיוון שלא היה מסמך ברור כיצד לעשות זאת נאלצנו לעבוד זמן רב יחסית על קמפול הקוד.
לבסוף, בנינו את התוכנה תוך כדי וידוא שהפלט זהה לפלט של הקבצים הבינארים שהורדנו מהאתר.

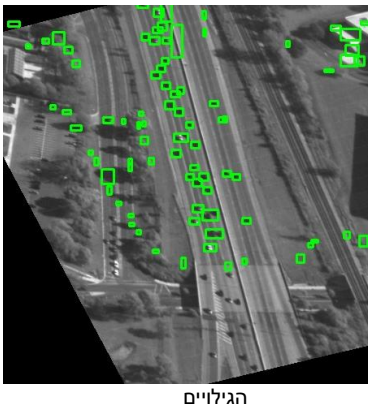
התוכנה מקבלת סרט וידאו אותו מפרידים לפריימים, או פריימים רציפים בזמן ובמרווחים קבועים.
בשלב הראשון, התוכנה מבצעת גילוי של העצמים הנעים בתמונה ופולטת לתיקיית ההרצה את תמונות הקלט עם סימונים של הגילויים (כפי שניתן לראות באיור 8)
לאחר מכן, מופעל האלגוריתם למציאת המסלולים. התוכנית פולטת את התוצאות הן בצורה ויזואלית והן בצורה טקסטואלית. נוצר קובץ טקסט של המסלולים ובנוסף תמונות הקלט עם סימונים של המכוניות בצירוף ה- ID (כפי שניתן לראות באיור 8).


הפלט בצורה ויזואלית


freeway01_tracks.txt

```
TRKDATA01
1450
4 2 131 644 4 6 R
5 2.5 131 643 10 8 R
6 3 131 643 10 8 R
7 3.5 131 643 10 8 R
8 4 131 643 10 8 R
9 4.5 145 642 6 4 R
10 5 154 642 6 4 E
11 5.5 163 641 4 9 R
12 6 177 640 6 12 R
13 6.5 187 642 6 12 E
14 7 197 645 3 8 R
15 7.5 196 645 4 8 R
16 8 203 646 4 8 E
17 8.5 211 648 16 12 R
18 9 225 648 16 12 E
19 9.5 240 649 5 6 R
8439
11 5.5 248 550 6 5 R
12 6 258 547 8 6 R
13 6.5 266 543 8 6 E
14 7 275 539 6 9 R
15 7.5 284 539 6 9 E
16 8 294 539 6 9 E
17 8.5 304 540 10 9 R
18 9 304 544 12 8 R
19 9.5 304 548 13 8 R
20 10 305 551 13 14 R
21 10.5 306 562 9 9 R
```

הפלט בצורה טקסטואלית


הגילויים

איור 8: הפלט של התוכנה.

2. בניית מדדים

לאחר השלב הראשון, בו הגענו למצב שיש בידנו תוכנית רצה רצינו לבדוק מה טיב התוכנית.
לשם כך, היה עלינו להגדיר מדדים לביצועי האלגוריתם/תוכנית.
בהתבסס על המאמר, ומאמר נוסף שמצויין בו, החלטנו על ארבעה מדדים:

- FAR – False Alarm Rate

$$\mathrm{FAR} = \frac{\#\text{false detections}}{\#\text{detections}}$$

מדד זה מייצג את אחוזי גילויי הסרק (שנוצרים למשל כתוצאה מרעש).

- ODR – Object Detection Rate

$$\mathrm{ODR} = \frac{\#\text{corect detections}}{\#\text{objects}}$$

מדד זה מייצג את אחוז הגילויים מתוך כלל האובייקטים.

- NTF – Normalized Track Fragmentation

$$\mathrm{NTF} = \frac{\sum_i |G_i| \cdot |A(G_i)|}{\sum_{i,A(G_i)\neq 0} |G_i|}$$

G_i – מסלול "אמיתי" (מסלול שהאלגוריתם אמור למצוא אותו).
T_i - מסלול שהתגלה על ידי האלגוריתם.
$A(G_i)$ – כל המסלולים $T_{i1}, T_{i2}, .., T_{in}$ שקיים בהם אובייקט אשר שייך למסלול G_i.
$|A(G_i)|$ – כמה מסלולים שונים מרכיבים את G_i

כאשר הערך של NTF הינו n המשמעות היא שאובייקט זוהה עם n מסלולים שונים. NTF אופטימאלי הינו 1. כמו כן, המדד נותן משקל גדול יותר למסלולים ארוכים כיוון שהם קשים וחשובים יותר.

- IDC - ID consistency

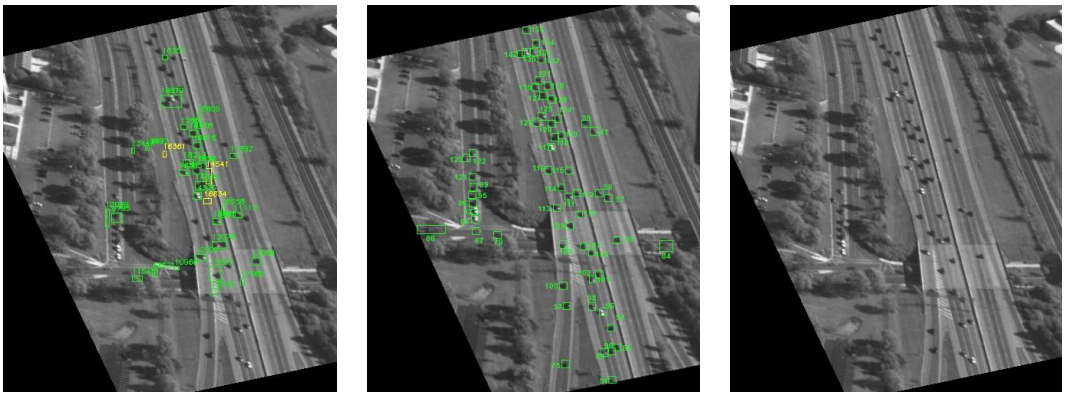
$$\mathrm{IDC} = \frac{1}{\sum_{G_i} |G_i|} \cdot \sum_{G_i} |G_i| \cdot \mathrm{IDC}_i \quad , \quad \mathrm{IDC}_i = \frac{\max_l \left|\{G_{ij} \mid \mathrm{ID}(G_{ij}) = l\}\right|}{|G_i|}$$

G_{ij} – סט הגילויים שהם חלק מהמסלול האמיתי G_i.
IDC_i – עבור מסלול G_i, מסמל את העקביות של הגילוי שלו. זהו למעשה, האחוז מתוך G_i שהתגלה על ידי אותו מסלול שחיתוכו עם G_i מקסימאלי.

מדד זה הינו ממוצע משוקלל לפי אורך המסלול של IDC_i. ערך אופטימאלי הינו 1.

כדי להעריך את תוצאות האלגוריתם, ביצענו הערכה <u>ידנית</u> של הפרמטרים:

- הכנו ידנית ground truth, כלומר מאגר של תמונות זהות לקלט, כאשר ההבדל הוא שסימנו בכל תמונה (frame) את האובייקטים, ומספרנו אותם לפי שיוך למסלול אמיתי (סיפור אמיתי).



תמונת פלט | תמונת ground truth שיצרנו | תמונת קלט

איור 9.

- ספרנו בכל frame את מספר האובייקטים, מספר הגילויים, ומספר הגילויים הנכונים.
- מיפינו כל מסלול אמיתי, למסלולים שהתגלו על ידי התוכנה.
- על סמך מדידות אלה, חישבנו את תוצאות הריצה הראשונה.

track_len	track	A(Gi)	A(Gi)*len
37	39	2	74
37	41	1	37
33	57	2	66
33	58	3	99
31	65	1	31
18	72	3	54
18	74	3	54
19	76	1	19
19	78	3	57
13	79	2	26
14	80	2	28
27	82	1	27
27	83	1	27
11	84	1	11
14	85	1	14
19	89	2	38
19	91	2	38
18	94	4	72
26	95	2	52
18	96	1	18
18	97	1	18
18	98	2	36
26	99	2	52
19	100	1	19

חישוב NTF

track_len	track	#detections	max	IDCi	Gi*IDCi
37	39	10	8	0.216216	8
37	41	11	11	0.297297	11
33	57	2	1	0.030303	1
33	58	23	21	0.636364	21
31	65	16	16	0.516129	16
18	72	3	1	0.055556	1
18	74	3	1	0.055556	1
19	76	1	1	0.052632	1
19	78	3	1	0.052632	1
13	79	3	2	0.153846	2
14	80	10	9	0.642857	9
27	82	1	1	0.037037	1
27	83	2	2	0.074074	2
11	84	3	3	0.272727	3
14	85	3	3	0.214286	3
19	89	2	1	0.052632	1
19	91	2	1	0.052632	1
18	94	17	14	0.777778	14
26	95	6	4	0.153846	4
18	96	10	10	0.555556	10
18	97	14	14	0.777778	14
18	98	2	1	0.055556	1
26	99	10	6	0.230769	6
19	100	15	15	0.789474	15

חישוב IDC

frame	total	good
0	44	0
1	41	0
2	41	0
3	45	0
4	42	0
5	49	0
6	50	0
7	54	0
8	51	0
9	52	0
10	55	0
11	52	0
12	53	1
13	57	2
14	54	3
15	52	4
16	56	5
17	54	9
18	54	16
19	52	17
20	53	20
21	56	20
22	53	18
23	51	20

חישוב ODR

	detections	FALSE	TRUE	false/det.
frame00	0	0	0	
frame01	0	0	0	
frame02	0	0	0	
frame03	0	0	0	
frame04	1	1	0	1
frame05	1	1	0	1
frame06	1	1	0	1
frame07	1	1	0	1
frame08	1	1	0	1
frame09	2	2	0	1
frame10	2	2	0	1
frame11	5	5	0	1
frame12	7	6	1	0.857143
frame13	10	8	2	0.8
frame14	12	9	3	0.75
frame15	14	10	4	0.714286
frame16	15	11	4	0.733333
frame17	18	9	9	0.5
frame18	24	8	16	0.333333
frame19	26	9	17	0.346154
frame20	27	8	19	0.296296
frame21	30	12	18	0.4
frame22	31	13	18	0.419355
frame23	34	14	20	0.411765

חישוב FAR

איור 10: הטבלאות של המדידות שעשינו ידנית כדי לחשב את המדדים.

התוצאות שקיבלנו בשלב זה:

- FAR = 0.61
- ODR = 0.26
- IDC = 0.18
- NTF = 4.46

3. שיפור ראשון

לאחר קבלת התוצאות הראשונות, ולאחר הפגישה עם ד"ר ישי קמון (המנחה) עלו מספר מסקנות:

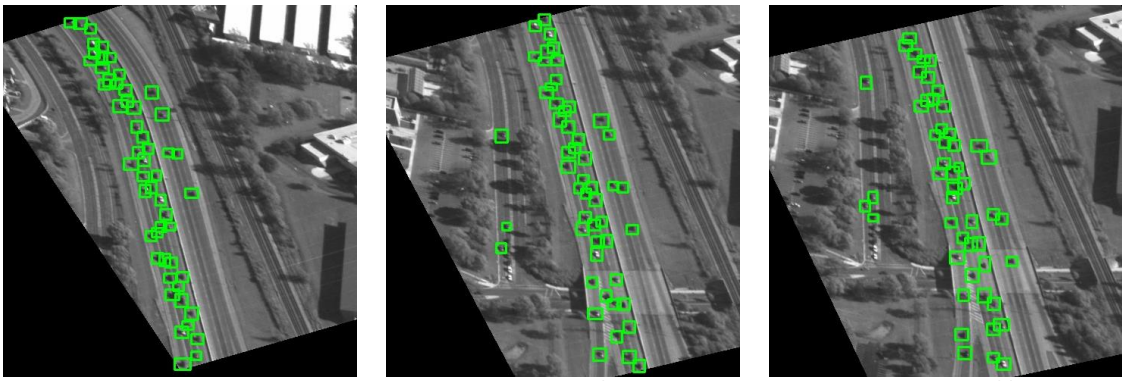
- לא סביר ולא כדאי לבצע מדידות ידניות... (-:
- תוצאות האלגוריתם כפי שהן לא טובות, ולא דומות למה שפורסם במאמר.
- נראה כי ביצועי האלגוריתם הורעו, בעיקר כתוצאה מגילוי העצמים בכל מסגרת (לפי FAR, ODR).

כתוצאה מכך, הוחלט בשיתוף עם ישי לשנות את התכנית כך שהקלט יכלול גם קובץ גילויי אמת (ground truth), שיחליף את הצורך בהפעלת אלגוריתם עזר לגילוי אובייקטים.
יש לציין, כי השימוש בקובץ היה לצורך גילוי האובייקטים בלבד – ולא לשם שיוכם למסלול כלשהו. כלומר, המטרה הייתה לדמות מצב של עולם מושלם שבו הגילויים נעשים בצורה מדוייקת, ונותר לבחון את פעולת האלגוריתם עצמו – שיוך האובייקטים למסלולים. לכן, אין יותר צורך במדדים FAR וODR.
כמו כן, עלה הצורך ביצירת כלי אוטומטי לחישוב המדדים, ובנוסף היה עלינו להשיג קובץ GT (ground truth) טקסטואלי.

לשם השגת קובץ GT פנינו שוב לאחד ממחברי המאמר. הוא שלח לנו קובץ GT יחיד, ולא היו לו קבצים למאגרי המידע הנוספים.

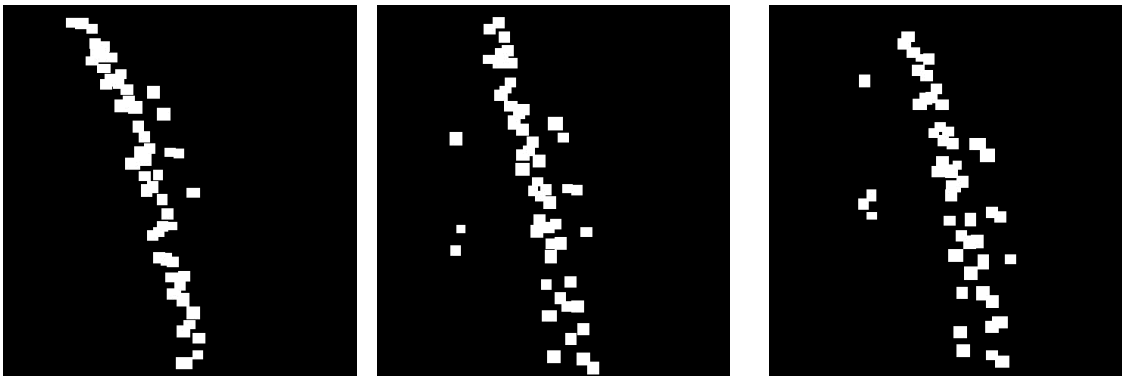
בשלב זה, חקרנו את הקוד שסופק לנו. ניסינו לזהות ולהפריד את חלקי האלגוריתם השונים, כאשר המטרה העיקרית הייתה לזהות את החלק שבו מופעל אלגוריתם העזר לגילוי עצמים, ולספק אסטרטגיה שבעזרתה נוכל להחליף במידת הצורך את גילוי העצמים באוביקטים מקובץ ה- GT.
הוספנו לתוכנה אופציה שמאפשרת לקבל כקלט קובץ GT. על מנת להחליף את גילוי האובייקטים בקלט מקובץ GT, הכנו פונקציה שקוראת קבצי GT וממירה אותם לפורמט של

אובייקטים שהתגלו על ידי האלגוריתם. כאשר האלגוריתם מגיע לשלב גילוי האובייקטים, הוא משתמש באובייקטים שסופקו על ידי הפונקציה במקום להפעיל אלגוריתם עזר לגילוי עצמים.



איור 11: הגילויים לאחר שינוי הקלט. ניתן לראות שהגילויים מדוייקים כעת בניגוד לגילויים לפני השינוי.

בנקודה זו נתקלנו בבעיה מכיוון שפונקצית הדמיון שבה משתמש האלגוריתם (עליה דובר בתיאור האלגוריתם) משתמשת ב-masking שמתבסס על masking שנוצר בעזרת אלגוריתם העזר לגילוי עצמים (וממומש על ידי תוכנה נפרדת). לכן, במקרה שגילויי האובייקטים באים מקובץ GT, היה עלינו ליצור masking שמותאם לאותם אובייקטים.
בעזרת ספריית OpenCV עליה למדנו תוך כדי העבודה (נעשה בה שימוש רב בתוכנה) כתבנו תוכנית עזר שעוברת על קובץ ה- GT ויוצרת masking שמתאים לקובץ.



איור 12: הפלט של תוכנית העזר שיוצרת מסכות.

כתבנו תוכנית שמחשבת את המדדים. התוכנית מקבלת כקלט את קובץ ה- GT וקובץ הפלט של התוכנית ומוציאה כפלט את המדדים NTF ו- IDC.

4. התוצאות לאחר שיפור זה:
- IDC = 0.52
- NTF = 1.49

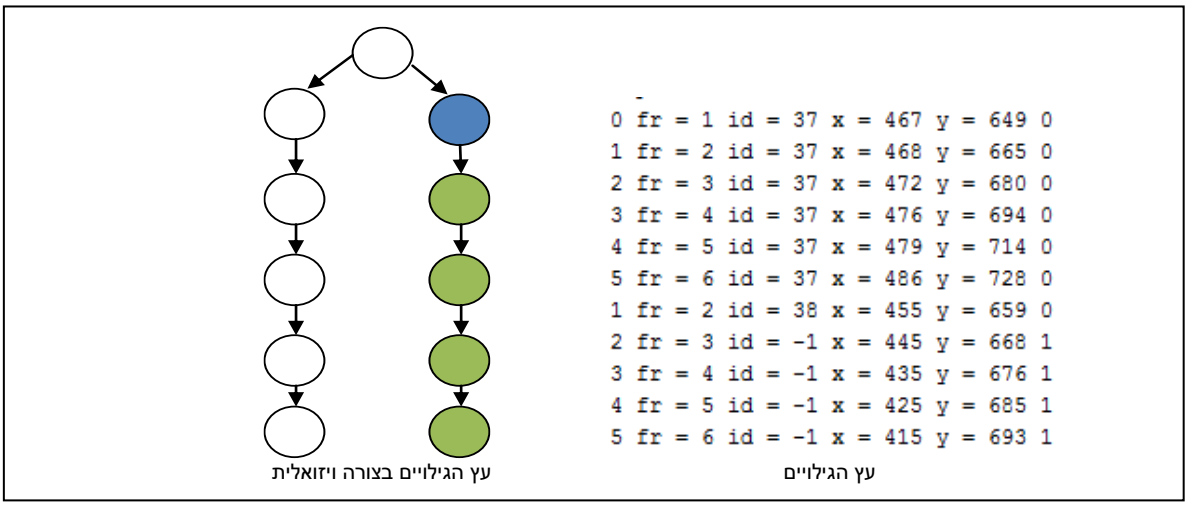
5. מחקר עצמי
בשלב זה ניסינו להבין מה גורם למדדים להיות לא מספיק טובים.
כדי לעשות זאת חיפשנו דוגמאות מעניינות על ידי בחירת מסלולים ספציפים מתוך קובץ ה-GT:
- מסלול יחיד (ארוך, קצר, מההתחלה, מהאמצע וכו')
- שני מסלולים שרחוקים אחד מהשני.
- שני מסלולים קרובים אחד לשני.

- שלושה מסלולים קרובים אחד לשני.
- מקבץ מסלולים ספציפים שהגילוי שלהם לוקה בחסר בהרצה על קובץ ה- GT המלא.

ההרצות על מסלול יחיד ועל שני מסלולים רחוקים היו מושלמות (IDC = NTF = 1).
בכמה מן הדוגמאות של מספר מסלולים קרובים גילינו מספר תופעות:

- מקרים שבהם אחד המסלולים לא התגלה כלל והשני/האחרים התגלו במלואם.
- מקרים שבהם תחילת המסלולים לא התגלו והשאר התגלה.
- מסלולים קצרים מחצי אורך חלון לא התגלו כלל.

במקרים שבהם תחילת המסלול לא התגלתה, חקרנו וחיפשנו את הגורם להסרת התחילית.
נתבונן בעץ הגילויים של אחת מן הדוגמאות שבהן תחילת המסלול הוסרה.
באיור 13 ניתן לראות הדפסה של עץ גילויים שהתבצע בהרצה עם גודל חלון 6, לאחר שבונים את המסלולון הראשון.



איור 13.

באיור 13:

- הצמתים הלבנים הם הגילויים ה"אמיתיים" של המסלול שמתחיל בשורש העץ.
- הצומת הכחול הוא גילוי אמיתי, אך של אובייקט ששייך למסלול אחר. אולם אובייקט זה היה "קרוב" אל שורש העץ שהתגלה קודם לכן, ולכן מופיע בעץ הגילויים. כמו כן, כנראה ששאר האובייקטים הכחולים אינם מופיעים בעץ מכיוון שסווגו כגילויי סרק (שלב ב') ולכן יש צורך בהוספת גילויים וירטואליים (שלב ג'1).
- הצמתים הירוקים הם אותם גילויים וירטואליים שהזכרנו.

הבעיה שנוצרה במקרה ספציפי זה, היא שהאלגוריתם בחר במסלולון ה"וירטואלי" בתור המסלולון ההגיוני ביותר שיצא מהגילוי הראשון, ולאחר מכן (כפי שתואר בהסבר על האלגוריתם) הוריד את מסלולון זה שכן מספר הגילויים הוירטואלים גדול מגודל חצי חלון.
הניסיון שלנו לפתרון מקרה זה, היה לתת עדיפות בבחירת מסלולון, למסלולון "אמיתי" – ואכן, הצלחנו לפתור את הבעיה עבור מקרה זה ועבור מקרים דומים.
אולם, בהרצה על ה-GT המלא, השינוי נתן תוצאות פחות טובות מאשר לפני השינוי.

במקרים בהם אחד המסלולים לא התגלה, גילינו כי בשלב כלשהו המסלול מתגלה אך בשלב מאוחר יותר האלגוריתם מוחק אותו. הסיבה לכך נעוצה בדרך הפעולה של האלגוריתם (שלב סינון מסלולונים דומים). נזכור שקיים רף מסויים שקובע האם מסלולונים דומים.
מצאנו שאכן במקרים כאלה, מסלול אחד לא התגלה שכן המסלולונים שמרכיבים אותו עברו את סף הדימיון ביחס למסלולונים של המסלול האחר.
ניסינו לשחק עם הסף המדובר עד שקיבלנו תוצאות טובות עבור דוגמאות אלו ואכן, הצלחנו

לשפר את התוצאות במקרים ספציפים. אולם, בהרצה על ה- GT המלא שוב השינוי נתן תוצאות פחות טובות.
הסיבה להרעה בתוצאות היא שעבור סף דמיון גבוה מדי, מתקבלים מסלולים רבים בעלי תחילית שונה וסיומת משותפת (מורכבים ממסלולונים הגיוניים שכן האובייקטים קרובים אחד לשני, ותנועתם דומה).

עבור מסלולים קצרים מחצי אורך חלון, לא ניתן היה לטפל בכך שלא התגלו כיוון שכפי שדובר בתיאור האלגוריתם, הם אינם אמורים להתגלות.

בשלב זה לא הצלחנו להשיג שיפור, לכן חשבנו מדוע שיפור שעבד עבור מקרה ספציפי הרס את התוצאות עבור המקרה הכללי. התמקדנו במקרה שבו הגדלנו את רף הדימיון בין מסלולונים, וכזכור במקרה זה נוספו מסלולים בעלי חלק משותף ותחילית שונה. בשלב זה שיערנו שאם למשל מסננים פחות מסלולים על ידי מדד דמיון, אז אולי יש להחמיר עם מדד התנועה החלקה.
לכן, החלטנו לבדוק פרמטרים נוספים פרט למדד הדמיון. בחרנו מספר פרמטרים שראינו שמימוש האלגוריתם משתמש בהם על מנת לסנן מסלולונים, ובדקנו מספר שילובים ואת השפעתם.
גילינו כי לא רק ששיפרנו את תוצאות המקרים הספציפים שתיארנו, אלא הצלחנו גם לשפר את תוצאות האלגוריתם על ריצת המקרה הכללי.

6. שיפור שני - ניסוי ידני של מציאת פרמטרים

לאחר שגילינו כי ניתן לשפר את האלגוריתם באמצעות שינוי בפרמטרים הפנימיים, החלטנו על מספר פרמטרים שהאלגוריתם משתמש בהם – שאותם נשנה כך שנקבל תוצאות טובות יותר:

א. Duplicate in window – פרמטר שקובע את הרף שעל פיו האלגוריתם **מסנן מסלולונים דומים** (ראה שלב ג' בתיאור האלגוריתם)
ב. Associate – פרמטר שקובע את הרף שעל פיו האלגוריתם מחליט האם **למזג מסלולון למסלול קיים** (שלב ד' בתיאור האלגוריתם)
ג. Motion smoothness – פרמטר שקובע את הרף שעל פיו האלגוריתם **מסנן מסלולונים בעלי תנועה "לא חלקה"** (שלב ג' באלגוריתם)
ד. Background threshold – פרמטר זה ואחרים משמשים את האלגוריתם כאשר יש שימוש באלגוריתם עזר **לגילוי עצמים**, ולכן פרמטר זה ודומיו אינם רלוונטים לתוצאת האלגוריתם ובחרנו שרירותית באחד מהם כדי שתהיה לנו בקרה לניסוי (התוצאות לא אמורות להשתנות כאשר נשנה את הפרמטר).

לאחר שהוחלט על הפרמטרים לבדיקה, ביצענו בדיקה ידנית מקיפה של הפרמטרים, כאשר רשמנו את התוצאות של כל שילוב של פרמטרים עד שלבסוף מצאנו את שילוב הפרמטרים שהיה הטוב ביותר עבורנו **עבור אותו dataset.**

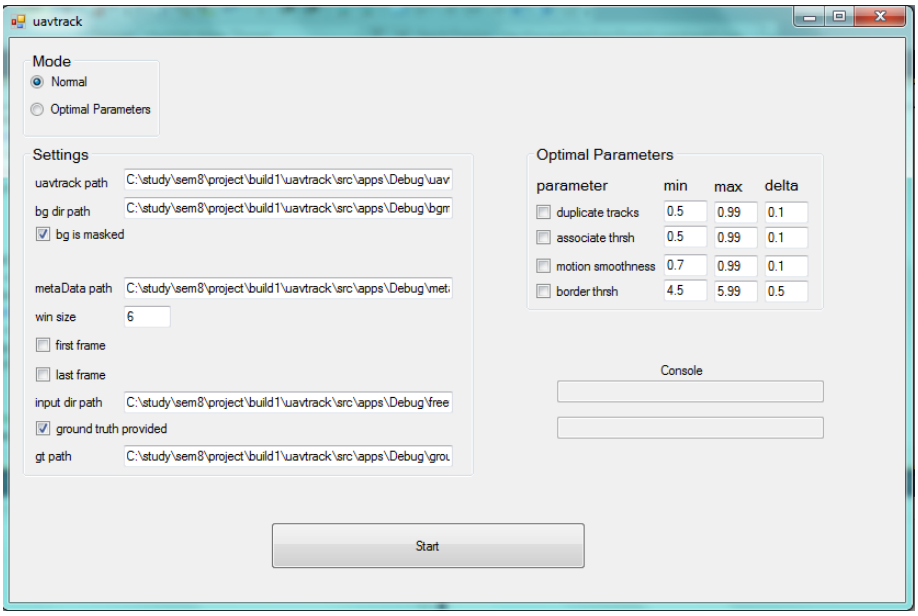
dupIInWindowThresh	associateThresh	motionProbThresh	minObservThresh	motionSmoot	minDeltaT	borderThres	NTF	IDC
0.55	0.55	0.1	0.75	0.82	0.495	5	1.48	0.52
0.9	0.55	0.1	0.75	0.82	0.495	5	1.58	0.59
0.95	0.55	0.1	0.75	0.82	0.495	5	1.61	0.58
0.8	0.55	0.1	0.75	0.82	0.495	5	1.48	0.61
0.7	0.55	0.1	0.75	0.82	0.495	5	1.45	0.61
0.6	0.55	0.1	0.75	0.82	0.495	5	1.48	0.56
0.7	0.9	0.1	0.75	0.82	0.495	5	1.47	0.67
0.7	0.8	0.1	0.75	0.82	0.495	5	1.44	0.68
0.7	0.7	0.1	0.75	0.82	0.495	5	1.4	0.68
0.7	0.6	0.1	0.75	0.82	0.495	5	1.41	0.63
0.7	0.7	0	0.75	0.82	0.495	5	1.09	0.63
0.55	0.55	0	0.75	0.82	0.495	5	1.12	0.54
0.7	0.7	0	0.9	0.82	0.495	5	1.09	0.63
0.7	0.7	0	0.3	0.82	0.495	5	1.09	0.63
0.7	0.7	0	0.75	0.6	0.495	5	1.1	0.64
0.7	0.7	0	0.75	0.7	0.495	5	1.09	0.64
0.7	0.7	0	0.75	0.8	0.495	5	1.08	0.63
0.75	0.7	0	0.75	0.8	0.495	5	1.11	0.63
0.7	0.7	0	0.75	0.8	0.995	5	1.14	0.64
0.7	0.7	0	0.75	0.8	0.495	0	1.08	0.63
0.7	0.7	0	0.75	0.8	0.495	10	1.09	0.64

איור 14: טבלה של המדידות הידניות. בכל מדידה מפורטים ערכי הפרמטרים וה- NTF ו- IDC שהתקבלו.

7. תוצאות לאחר שיפור זה:
   - IDC = 0.64
   - NTF = 1.09

8. שיפור שלישי - בניית המערכת למציאה אוטומטית של פרמטרים אופטימליים

השלב הבא שהחלטנו עליו, הוא שבהינתן dataset מסויים, ניתן למחשב למצוא עבורנו את הפרמטרים שייתנו את התוצאות הטובות ביותר עבור אותו dataset.
לשם כך הגדרנו מרחב חיפוש, שמוגדר כך שלכל פרמטר יש ערך מינימלי, ערך מקסימלי, ודלתא. המטרה שלנו הייתה לבנות תוכנית שתעבור על מרחב החיפוש שהוגדר, ובכל איטרציה תריץ את האלגוריתם עם המצב הנוכחי ותמצא את השילוב שנותן מקסימום תוצאה.
מכיוון שאנו מחפשים לשפר הן את מדד ה-NTF, והן את מדד ה-IDC, עלינו למצוא נוסחה שתאגד את שני המדדים למספר יחיד. כמו כן, אנו יודעים שככל שמשפרים את ביצועי האלגוריתם, מדד ה-NTF שואף ל-1 מצד ימין ($\mathrm{NTF} \geq 1$), ומדד ה-IDC שואף ל-1 מצד שמאל ($\mathrm{IDC} \leq 1$). לכן הנוסחה שהצענו היא פשוט: $\frac{\mathrm{IDC}}{\mathrm{NTF}}$.
על מנת לממש זאת, הוספנו לתכנה אפשרות לקבלת קובץ קונפיגורציה והפעלת mode חדש של מציאת פרמטרים אופטימיליים. בmode העבודה החדש, אתחלנו את הפרמטרים בהתאם לקובץ הקונפיגורציה, ובכל איטרציית חיפוש במרחב החיפוש שנוצר הרצנו תחת אותו process את האלגוריתם.
בנקודה זו נתקלנו בבעיה, שכן התברר לנו שבמימוש האלגוריתם שקיבלנו קיימת דליפת זיכרון ולכן התכנה קרסה לאחר מספר קטן של איטרציות.
מכיוון שמדובר בכמות קוד עצומה ולא מתועדת, ומכיוון שהקוד נכתב בידי מישהו אחר – הגענו למסקנה שאין טעם "לדבג" את הקוד ולמצוא את דליפת הזיכרון, אלא עדיף למצוא דרך חלופית.
הפיתרון שלנו לבעיה זו, היה לגרום לכל הרצה של האלגוריתם (כל איטרציה במרחב החיפוש) לרוץ בתור process נפרד, וכך בסיום כל ריצה מערכת ההפעלה תפתור אותנו מבעיית דליפת הזיכרון.
כמו כן, החלטנו שננצל את העבודה על הפרדת ההרצה לתהליכים נפרדים על מנת ליצור סביבת הרצה נוחה יותר של האלגוריתם (ב- mode הרגיל, וב- mode של מציאת פרמטרים אופטימאלים) – ולכן החלטנו ליצור GUI שיהיה אחראי על הרצת האלגוריתם.



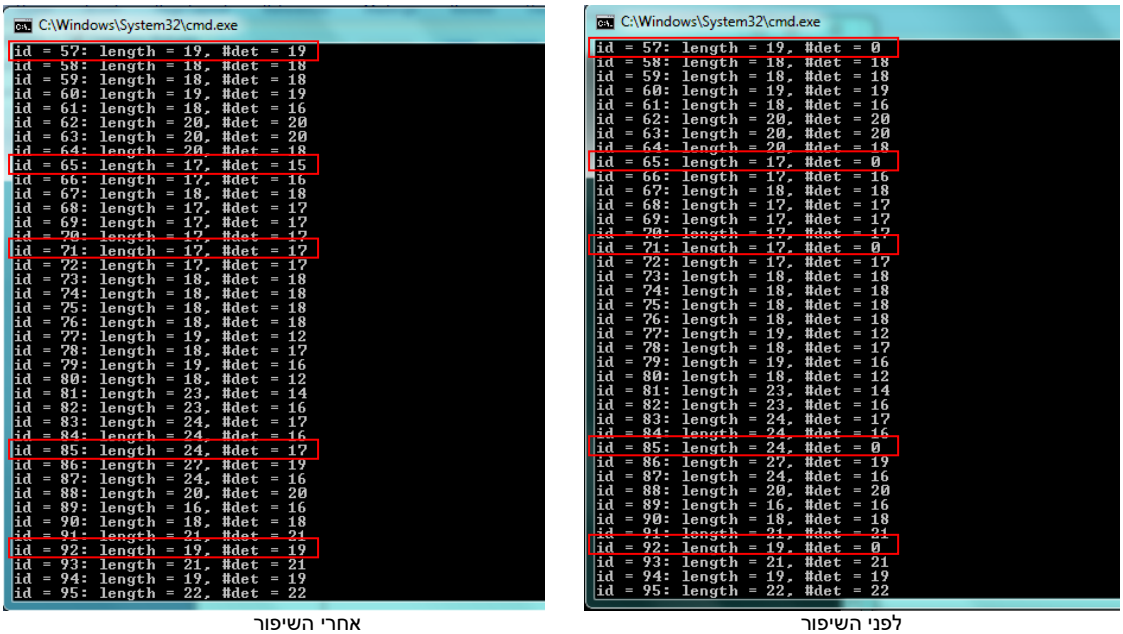
איור 15: ה- GUI איתו ניתן להריץ את התוכנית. ניתן לראות שאפשר לבחור בין הרצה ב- mode רגיל או ב- mode של מציאת פרמטרים אופטיאמלים. כמו כן, ניתן לבחור מהו גודל החלון, האם הקלט הוא קובץ GT או תמונות ועוד.
במקרה של הרצה ב- mode של מציאת פרמטרים אופטיאמלים יש לסמן אילו פרמטרים רוצים לבדוק, מה הערך המינימאלי והמקסימאלי של כל אחד ומהו הדלתא.

9. תוצאות לאחר שיפור זה:

- IDC = 0.64
- NTF = 1.08

10. שיפור רביעי

לאחר השיפור שקיבלנו על ידי מציאת פרמטרים, רצינו לבדוק איזה עוד שיפור ניתן להכניס לאלגוריתם. חיפשנו סטטיסטיקות על התוצאות שיתנו לנו כיוון שבו נוכל לשפר את האלגוריתם.
שמנו לב שעדיין ישנם מסלולים רבים שאינם מתקבלים בגלל שהם קרובים למסלול אחר ולכן מתבטלים (כפי שניתן לראות באיור 16)
החלטנו לבדוק מה קורה כשמריצים את האלגוריתם על המסלולים שלא התגלו בלבד. הסתבר שישנם מסלולים שלא מתגלים בהרצה הראשונה מתגלים בהרצה השניה. כלומר, ההשערה שעדיין יש מסלולים שמתבטלים בגלל היותם קרובים למסלול אחר נכונה.
לכן, החלטנו לבצע שיפור נוסף- במקום להריץ את האלגוריתם פעם אחת, נבצע שתי איטרציות. איטרציה ראשונה מפעילה את האלגוריתם על קובץ ה- GT המלא. לאחר מכן, מתוך כל הגילויים, מורידים את הגילויים שכבר שויכו למסלול כלשהו ומריצים את האלגוריתם שוב על גילויים שלא שויכו.



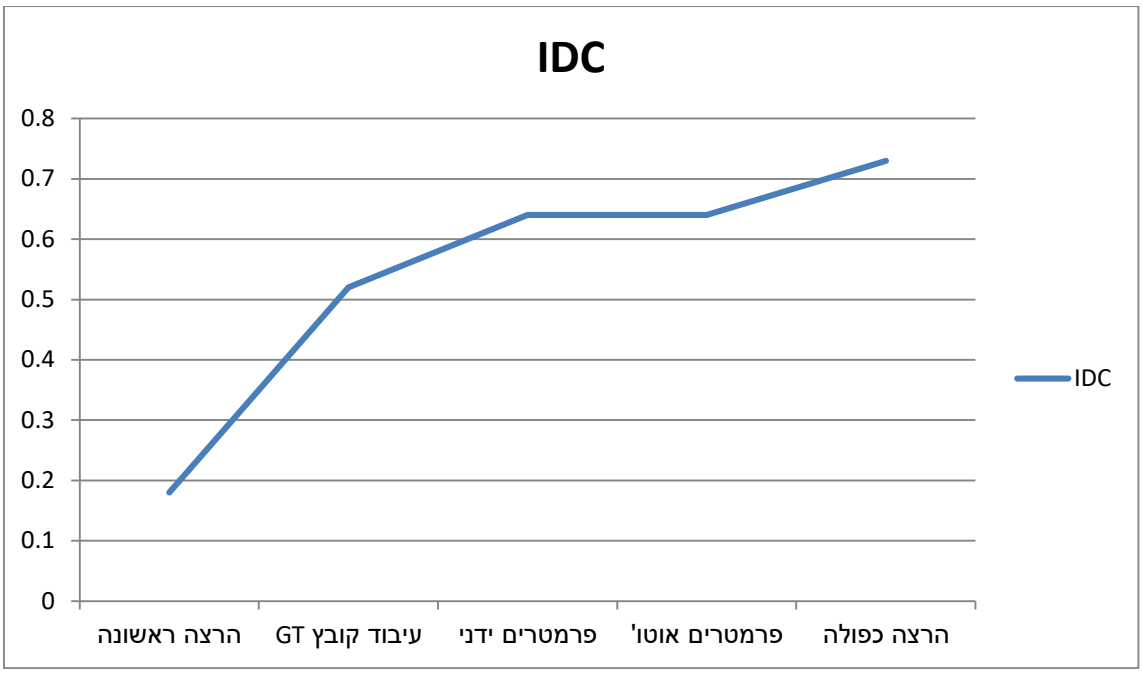
איור 16: הדפסות מתוך ניתוח הפלט. ניתן לראות שישנם מסלולים שלפני השיפור לא מתגלים כלל ואחריו מתגלים במלואם או כמעט במלואם.

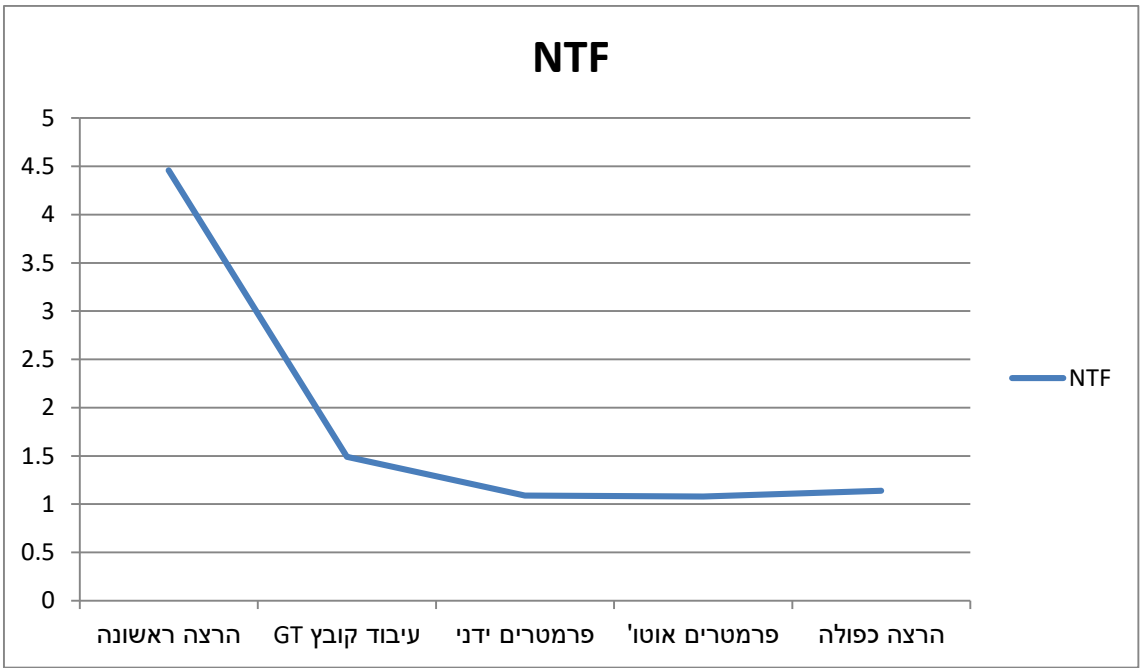
כפי שקיווינו, לאחר השיפור ישנם מסלולים נוספים שקודם לא התגלו כלל וכעת מתגלים באופן מלא (או כמעט מלא). עם זאת, מבחינת המדדים, ה- IDC השתפר אך ה- NTF הורע מעט.
דבר זה הגיוני שכן ברגע שמוסיפים עוד מסלולים יש סבירות רבה לפגיעה ב- NTF. השיפור ב- IDC היה יותר משמעותי מהפגיעה ב- NTF ולכן החלטנו ששיפור זה כדאי.

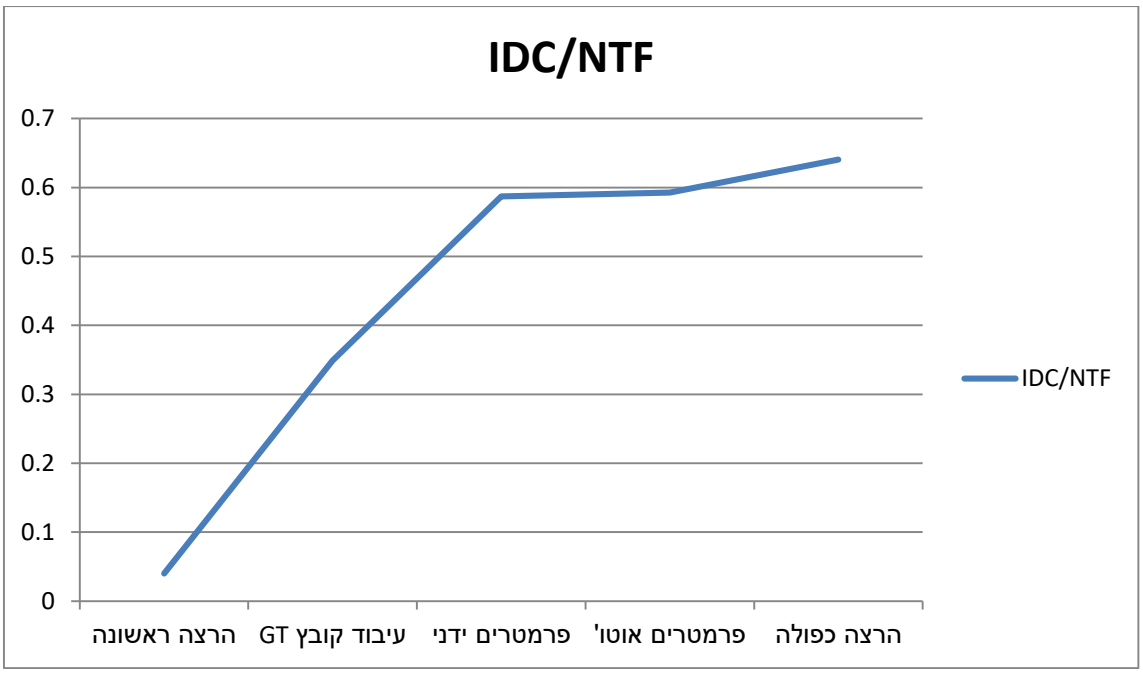
11. תוצאות לאחר שיפור זה:
- IDC = 0.73
- NTF = 1.14

סיכום

מטרת הפרויקט הייתה להריץ את האלגוריתם שמוצע במאמר, כאשר ציפינו לתוצאות שניתנו שם.
בפועל קיבלנו קוד עבור האלגוריתם אך התוצאות היו רחוקות מהמובטח.
מנקודה זו התחלנו לחקור לעומק את האלגוריתם ואת הסיבות לפער בביצועים. ככל שעבר הזמן, וככל שהעמקנו יותר באלגוריתם ובקוד עצמו, הצלחנו אט אט לשפר את הביצועים – וזאת למרות שהיינו "כבולים" במידה מסוימת שכן הצלחנו להשיג רק dataset אחד.
ניתן לראות את ההתקדמות בביצועי האלגוריתם בגרף המתאר נקודות דגימה של הביצועים לאורך שלבי הפרויקט:


IDC
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
הרצה ראשונה
עיבוד קובץ GT
פרמטרים ידני
פרמטרים אוטו'
הרצה כפולה
IDC


NTF
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
הרצה ראשונה
עיבוד קובץ GT
פרמטרים ידני
פרמטרים אוטו'
הרצה כפולה
NTF



בנוסף, תוך כדי המחקר שביצענו ותוך התייעצות עם ישי (המנחה), עלה בידינו ליצור כלי שעזר לנו לשפר את ביצועי האלגוריתם. כתבנו תכנית שבהינתן dataset מסוים, "מעדנת" את האלגוריתם על ידי מציאת פרמטרים מתאימים כך שביצועיו משתפרים.
אנו חושבים שכלי זה הוא ייחודי, שימושי ומועיל. השימוש שאנו רואים לכלי כזה:
כאשר רוצים להשתמש באלגוריתם, ניתן להכין (גם ידנית) דוגמא של מסלולים אמיתיים, למשל על ידי תצפיות על איזור מסויים. על ידי דוגמא זו, וביחד עם הכלי שהכנו ניתן לעדן את האלגוריתם כך שיתאים את עצמו למציאת מסלולים בצורה שעובדת טוב יותר עבור אותו תוואי השטח שנצפה.
לדוגמא, כאשר רוצים להפעיל את האלגוריתם על תוואי שטח מסוג:

- כביש מהיר: אובייקטים קרובים אחד לשני בכל frame, ותנועתם קרובה מאד לקו ישר. לכן יש להקל עם מדד הדמיון, ולעומת זאת להחמיר עם מדד התנועה.
- עירוני, רחובות חד נתיביים: אובייקטים בכל frame רחוקים אחד מן השני (כביש חד נתיבי), קיימים צמתים שפוגעים במעט ב"איכות התנועה". לכן, יש להחמיר במדד הדמיון, ולהקל במדד התנועה.

כלומר, בהינתן תוואי שטח, נוכל להשתמש בכלי הנ"ל על מנת "לכייל" את האלגוריתם.
נציין, שלדעתנו פרמטרים שיתאימו לתוואי שטח של כביש מהיר יפגעו בביצועי האלגוריתם על תוואי שטח עירוני חד נתיבי. כמו כן, לדעתנו לא סביר שקיימים פרמטרים שעבורם האלגוריתם מתאים לכל תוואי שטח.

0 fr = 0 id = 3 x = 524 y = 960 0
my childern:
fr = 1 id = 3 x = 530 y = 972 0
fr = 1 id = 4 x = 549 y = 966 0
fr = 1 id = 5 x = 532 y = 960 0
fr = 1 id = 19 x = 546 y = 987 0
fr = 1 id = 20 x = 532 y = 990 0

1 fr = 1 id = 3 x = 530 y = 972 0
my childern:
fr = 2 id = 3 x = 533 y = 983 0
fr = 2 id = 5 x = 533 y = 968 0

2 fr = 2 id = 3 x = 533 y = 983 0
my childern:
fr = 3 id = 3 x = 538 y = 996 0
fr = 3 id = 5 x = 540 y = 984 0

3 fr = 3 id = 3 x = 538 y = 996 0
my childern:
fr = 4 id = 3 x = 538 y = 1009 0
fr = 4 id = 5 x = 539 y = 995 0

4 fr = 4 id = 3 x = 538 y = 1009 0
my childern:
fr = 5 id = 3 x = 539 y = 1022 0
fr = 5 id = 5 x = 548 y = 1009 0

5 fr = 5 id = 3 x = 539 y = 1022 0
my childern:
fr = 6 id = 3 x = 546 y = 1033 0
fr = 6 id = 5 x = 548 y = 1021 0

6 fr = 6 id = 3 x = 546 y = 1033 0
my childern:
fr = 7 id = 3 x = 550 y = 1046 0
fr = 7 id = 5 x = 553 y = 1036 0

7 fr = 7 id = 3 x = 550 y = 1046 0
my childern:
fr = 8 id = 3 x = 553 y = 1062 0
fr = 8 id = 5 x = 554 y = 1047 0

8 fr = 8 id = 3 x = 553 y = 1062 0
my childern:
fr = 9 id = 3 x = 554 y = 1071 0
fr = 9 id = 5 x = 561 y = 1061 0

9 fr = 9 id = 3 x = 554 y = 1071 0
my childern:
fr = 10 id = -1 x = 558 y = 1084 1