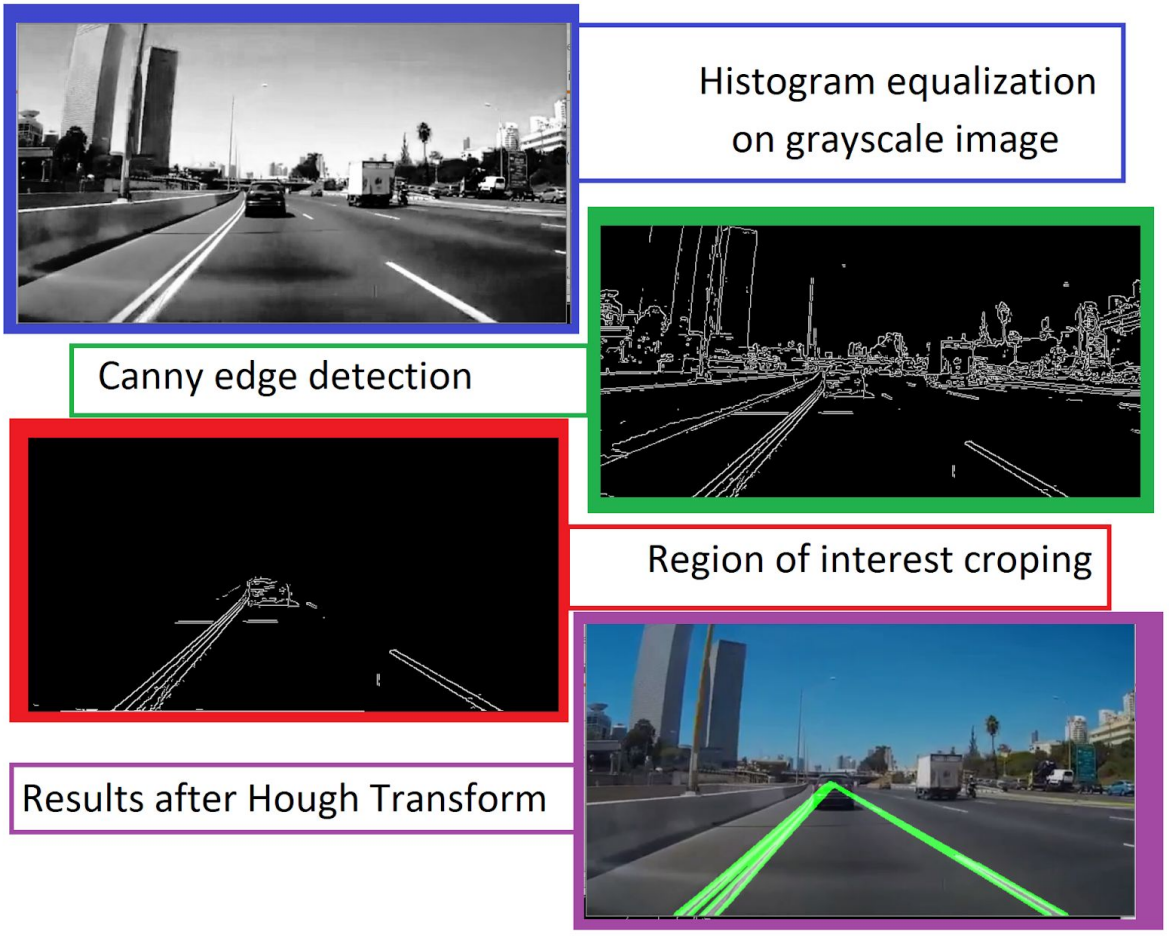
Lane-detection

Shaked Sibony

with minimal OpenCV usage



הקדמה:

מה הבעיה?

בעבר הלא רחוק מכוניות אוטונומיות בעולם הראיה הממוחשב היו לא רחוקות ממדע בדיוני.
אך בעקבות שינויים משמעותיים בעולם התוכנה אלגוריתמים מתוחכמים ניתן להגיע לתוצאות מרשימות בצורה יעילה ומהירה.
בעקבות העלייה המשמעותית בכלי הרכב הנמצאים על הכביש חלה גם עליה משמעותית בכמות תאונות הדרכים, ועל פי סטטיסטיקות רוב התאונות הם טעויות אנוש , ולכן יש צורך יותר מתמיד למנוע אותם.
הבעיה בה אני אתמקד היא בעיית העייפות בנהיגה וסטייה מהנתיב בכביש בפרט.

כיצד נוכל לפתור את הבעיה?

בעשור האחרון מכוניות אוטונומיות מתחילות לעלות על הכביש , בינה ממוחשבת מצמצמת את טעויות האנוש משמעותית ואף הציוד הנדרש לכך לא יקר באופן יחסי (mobileye דוגמה מעולה לכך).
בפרויקט זה אדגים את יכולת מצלמת הרכב הזולה הסטנדרטית לקבלת יכולת עיבוד תמונה מרשימות באמצעות אלגוריתמים בעולם עיבוד התמונה והשפה העילית python.

כיצד ניגשתי לבעיה?

ראשית בחרתי לעבוד עם פייטון כשפת התכנות שבה אעבוד על הפרויקט בגלל הניסיון שיש לי בה וכן היא השפה הפופולרית ביותר לפרויקטי עיבוד תמונה פשוטים.
ניתן לחלק את הפרויקט ל8 חלקים עיקריים של האלגוריתם:

1.קבלת פריים (תמונה) מקורית מהוידאו.
2.הפיכת התמונה לgrayscale.
3.הפעלת Histogram Equalization כדי לחדד את צבע הנתיבים (צבעים בהירים) מצבע הכביש (צבעים כהים).
4.לעשות edge detection באמצעות אלגוריתם Canny באופן דינמי צבעי התמונה.
5.לחתוך את התמונה כדי להיפטר מאזורים לא רלוונטים בתמונה (מנורות רחוב,בניינים,שמיים).
6.הפעלת אלגוריתם Hough Transform למציאת קווים ישרים (הנתיבים שלנו).
7.סינון קווים ישרים לא רצויים וציור הקווים הרלוונטים בלבד בתמונה המקורית.
8.הצגת התמונה המקורית תוך מעבר לפריים הבא חלילה לשלב '1'.

פירוט האלגוריתם:

שלב 1 -לקיחת פריים

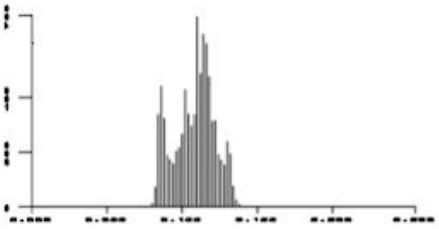
קבלת הפריים היא פשוטה יחסית באמצעות שימוש בספריית הקוד הפתוח OpenCV

שלב 2 - גווני אפור

גם הפיכת התמונה לגווני אפור פשוטה מאוד באמצעות OpenCV

שלב 3 - הפעלת Histogram Equalization

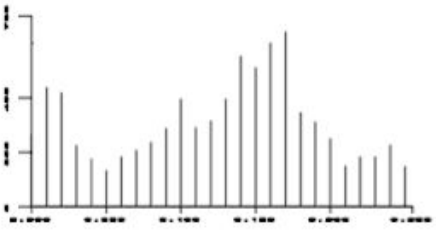
לאחר ניסוי וטעיה ראיתי כי ברוב המקרים הפעלת השוואת גוונים זו מחדדת את התוצאות משפרת משמעותית את תוצאות הנסיעה הלילית בעקבות חידוד הנתיבים



3.1 הטכניקה הפשוטה ביותר היא מתיחת היסטוגרמה (histogram stretching) זוהי מתיחה לינארית של תחום רמות האפור, המבוססת על המיפוי

$$\alpha_{new} = \frac{\alpha - L^-}{L^+ - L^-}(L_{new} - 1)$$

כאשר $\alpha \in [L^-, L^+]$ היא רמת האפור באות הכניסה, ו- $\alpha_{new} \in [0, L_{new} - 1]$ היא רמת האפור באות התוצאה.

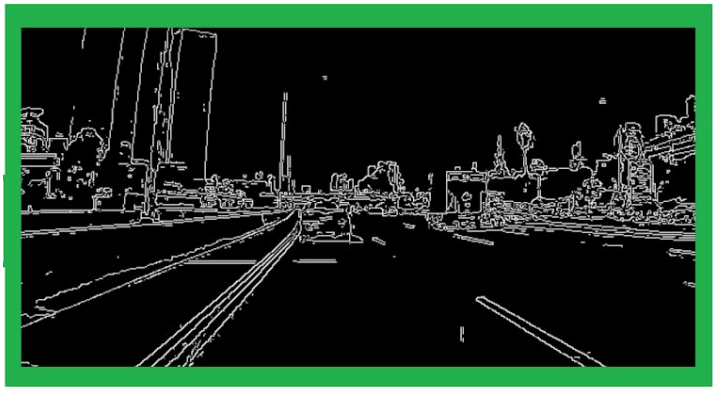


שלב 4 - edge detection with canny algorithm

בתחילת העבודה עשיתי מחקר נרחב על האלגוריתם המועדף עליי על מנת לזהות גבולות, בסופו של דבר לפי הבדיקות שעשיתי מצאתי כי Canny הוא בעל האיכות האופטימלית מבחינתי לביצוע מטלה זו

תיאור כללי של שלבי האלגוריתם:

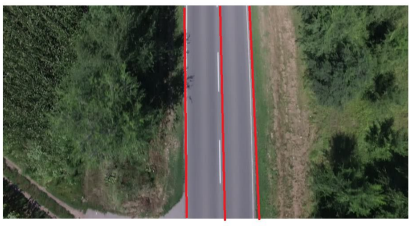
1. חישוב פונקציית הגרדיאנט- מציאת נגזרות חלקיות של פונקציית הגוון, עבור כל נקודה בתמונה מתקבל וקטור נגזרת.
2. מציאת גודלו וכיוונו של ווקטור הנגזרת.
3. הנקודות שבהן גודל וקטור הנגזרת קטן מהסף-התחתון נפסלות מייד לשמש כקצוות ואלו שבהן הוא גדול מהסף העליון מייד מסומנות כקצוות.
4. הכיוון של וקטור-הנגזרת של כל נקודה מהנקודות שנותרו מקורב לאחד מארבעה כיוונים בדידים: אופקי, אנכי או אלכסוני (אחד משני הכיוונים האלכסוניים).
5. הנקודות שעוזרות ליצור רצף של נקודות בעלות אותו כיוון-נגזרת עם נקודות שכבר סומנו כקצה, מסומנות גם הן כקצוות.



שלב 5 - חיתוך איזורים לא רלוונטים בתמונה

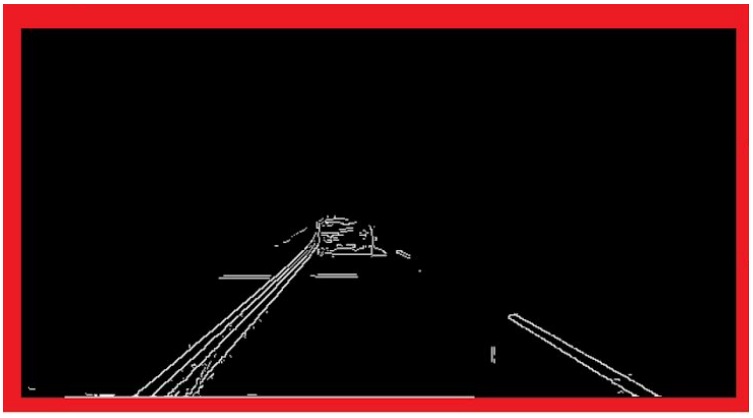
מכיוון מצלמת הרכב מצלמת עם עדשת Fisheye כך שככל שהנתבים קרובים יותר לרכב כך הם מתפרסים יותר לצדדים

הנתיבים במצלמת הרכב נראים לא מקבילים



אך מנקודת מבט שונה קל לראות שהם מקבילים

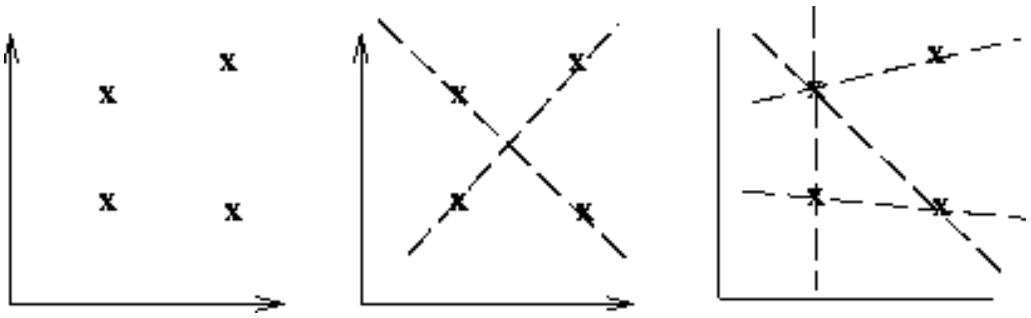
לכן עליי לקחת אך ורק משולש הלוקח רק שטח מצומצם מאוד מהתמונה (יהיה בשטח הזה רק את הקרקע ומכוניות.



שלב 6 - Hough Transform for detecting lines

לאחר קבלת תמונה בעלת גבולות בוררים והסרת כל החלקים הלא רלוונטים אפשר לעשות את אלגוריתם אף

מטרת האלגוריתם הוא מציאת מקרים לא מושלמים של אובייקטים בתוך סוג מסוים של צורות על ידי תהליך "הצבעה". הליך הצבעה זה מתבצע במרחב פרמטרים, שממנו אובייקטים "מועמדים" מתקבלים כמקסימום מקומי במרחב סוכם (Accumulator) הנבנה על ידי האלגוריתם. אלגוריתם האף בהגדרתו הראשונית עסק בזיהוי של קווים בתמונה את אלגוריתם התמרת האף כדי למצוא קווים במרחב.



שלב 7 - סינון false positives

בתהליך זה בנוסף למימוש הקלאסי של אף הוספתי סינון שמאפשר לי למחוק קווים לא רצויים ,בעיקר קווים מאונכים וקווים שהופיעו על רכבים. וכך להסיר קווים שאינם נתיבים בכביש.

שלב 8 - הצגת התמונה למשתמש

באמצעות פונקציה פשוטה של OpenCV הצגתי את התמונה עם פסי הנתיבים למשתמש וחזרתי חלילה לשלב 1 על הפריים הבאה עד לסוף הסרטון.

תוצאות

מה הצלחתי להשיג

שולי הכביש לאורך כל הנסיעה מסומנים בצבע ירוק כפי כשרציתי. הקווים המקווקווים לאורך רוב הנסיעה מסומנים גם כן. אך, ניתן לראות שיש קושי לזהותם לעומת השוליים.
לאחר ניסוי וטעיה באמצעות הפעלת Histogram Equalization, מצלמת הרכב מזהה נתיבים גם בלילה וגם ביום.
באמצעות חתיכת התמונה זיהוי הנתיבים מאוד ממוקד ויש מספר מצומצם של false positive .

נקודות שטרם הושגו:

-העלאת FPS - שיפור מהירות האלגוריתם על ידי מניעת צוואר בקבוק של משאבי המחשב (עבודה עם מערכות הפעלה מבוססות לינוקס) .
-הפחתת הבהובי הנתיבים המקווקווים לטובת חווית המשתמש.

מקורות

https://he.wikipedia.org/wiki/%D7%90%D7%9C%D7%92%D7%95%D7%A8%D7%99%D7%AA%D7%9D_%D7%A7%D7%A0%D7%99?veaction=edit&section=2

https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html

http://www.cs.technion.ac.il/~mic/ISP/ch4.pdf

/https://alyssaq.github.io/2014/understanding-hough-transform

https://github.com/oskarboer/Hough-Transform-lines/blob/master/Hough_Transform_lines.py

https://he.wikipedia.org/wiki/%D7%94%D7%AA%D7%9E%D7%A8%D7%AA_%D7%94%D7%90%D7%A3