שם התלמידה: איילה אילוז

מספר זהות: 326148921

מקום לימודים: סמינר בית יעקב רכסים

שם המנחה: המורה רבקה נילסון

תאריך הגשה: יוני 2024

תוכן עניינים

1. הצעת פרוייקט

רקע תאורטי:

רכב אוטונומי זהו רכב שלרוב פועל בכוחות עצמו ואינו נזקק לעזרתו של הנהג.
המכונית האוטונומית משתמשת בחיישנים שונים שקולטים מידע על הסביבה, מעבדים אותו באמצעות אלגוריתמים מתקדמים של בינה מלאכותית כגון yolo ומנווטים את המכונית בהתאם
תוך שמירה על בטיחותו של הנהג ובטיחות הסובבים אותו.
חיישנים אלה כוללים: מד מהירות על הגלגל, GPS, מצלמות, לידאר ורדאר.
כ-23 מיליון בני אדם עובדים בתעשיית הרכב העולמית, ורק כרבע מהם מועסקים ביצרניות הרכב
עצמן. השאר הם עובדי ספקיות המשנה שמייצרות רכיבים שונים לתעשייה – בלמים, צמיגים, מערכות מחשוב וכדומה.
לאלה מצטרפות בשנים האחרונות חברות הייטק, על רקע פיתוח המכוניות האוטונומיות.
שרשרת האספקה של יצרניות המכוניות מורכבת ממאות חברות.
חברות השלב הראשון (טיר 1) מוכרות את המוצרים שלהן ישירות לחברות הרכב,
וחברות השלב השני (טיר 2) הן ספקות רכיבים לשלב הראשון וליצרניות הגדולות.
מטרתה של המצאת הרכב האוטונומי היא לשפר את הבטיחות בדרכים- כיוון שהמכונית מתוכנתת עפ"י חוקי התנועה ומצייתת להם, לצמצם את הגודש בכבישים, להקל על פעילות הנהג ולשמור על איכות הסביבה.

יתרונות רכב אוטונומי:

- **פינוי שטחי חניה רבים לטובת בניה-** לפי הערכות עד 2030 עשרה אחוזים מהרכבים שמיוצרים בעולם יהיו אוטונומיים, וסביר להניח שעד 2035 מחצית מכלי הרכב כבר יהיו אוטונומיים. סביר להניח שמספר כלי הרכב הנעים בכבישים יופחת ורבים יוותרו על רכישת ואחזקת מכונית פרטית. השחרור מצורכי חניה יפנה מיליוני מטרים של נדל"ן.
מכונית אחת תוכל לשמש הרבה אנשים. וזאת בגלל שכאשר אדם צריך לנסוע, הוא מזמין מכונית,
והמכונית הכי קרובה אליו בסביבתו תגיע אליו. כך שלו היה לו רכב הוא היה חונה המון שעות סתם ותופס שטח חניה לחינם.
- **מניעת תאונות וטעויות אנוש בנהיגה-** על פי נתוני משרד התחבורה האמריקאי 9 מתוך 10 תאונות מתרחשות בשל טעויות נהג, אבל המחקר קובע כי כשליש מהתאונות נובע מצירוף נסיבות שגם תגובה מהירה ומדויקת לא הייתה יכולה למנוע את התרחשותן. למשל הולך רגל שקופץ פתאום לכביש קרוב מדי מכדי שרכב יספיק לבלום. או מצבים אחרים שבהם אי-אפשר

היה לצפות את ההתנהגות של נהג אחר. בוחני המכון, שמבצעים גם מבחני ריסוק למכוניות חדשות, בחנו 5,000 תאונות.

לדבריהם, רכב אוטונומי לא יסבול מהסחת דעת, עייפות או שכרות שמהם סובל נהג אנושי, ותמיד יוכל להגיב במהירות למצבי חירום, כמו כן מציית לחוקי התנועה, אבל תאונות רבות מתרחשות בנסיבות שגם מכ"ם, מצלמה וחיישנים אחרים לא יכולים לצפות, או דווקא בשל העובדה שלעיתים נדרשת חשיבה יצירתית ועקיפת החוקים דבר שקיים רק אצל בן אנוש.

- **שמירה על סביבה נקייה-**מכוניות אוטונומיות רבות מונעות על ידי מנועים חשמליים, המייצרים אפס פליטות בנקודת השימוש. זה מפחית את זיהום האוויר ואת פליטת גזי החממה, ועוזר להילחם בשינויי האקלים.

כמו כן, מכוניות אוטונומיות יכולות להשתמש באלגוריתמים מתקדמים כדי לייעל את המסלולים שלהן, להפחית עומס ופקקים ולשמור על מהירות נסיעה קבועה. זה עוזר למזער את צריכת הדלק והפליטה על ידי הימנעות מתנועת סרק מיותרת ותנועת עצירה וסע.

- **צמצום העומס בכבישים-** כל אדם שירצה לנסוע, יזמין אליו מכונית דרך האפליקציה והמכונית הפנויה בסביבתו תגיע אליו באופן אוטומטי. דבר זה יצמצם את כמות הרכבים כיוון שהרבה יעדיפו להשתמש במודל חדשני זה, ללא עלויות אחזקת הרכב.

חסרונות רכב אוטונומי:

- **הסקת מסקנות שגויה-**רכב אוטונומי מגיב מהר יותר מרכב המופעל אנושית, ועלול גם לנוע בפתאומיות. כמו כן, הוא מתוכנת לציית תמיד לחוקי התנועה והתמרורים, אך קיימים מצבים שבהם התנהלות חוקית 'לפי הספר' עלולה להוביל לתאונה.

כשמשתלבים בכביש מהיר, לדוגמה, בעוד שאר המכוניות נעות במהירות גבוהה יותר מהמותרת, מצפים שגם האוטונומית תתאים את מהירותה, כדי למנוע מנהג שנע לא מרוכז מאחור להתנגש בה.

בעולם הבינה המלאכותית אין ביכולת המתכננים לצפות מראש כיצד תגיב המערכת הממוחשבת ובהתאם אין באפשרותן לצפות כיצד יפעל הרכב בכל מצב משתנה.

- **הגברת העומס בכבישים-**עלות הנסיעה ברכב זה תהיה זולה כיוון שהמכונית נוסעת ללא נהג, וכך בכל נסיעה אנו חוסכים תשלום לנהג, ואז רבים יעדיפו לנסוע במכונית האוטונומית ולא בתחבורה ציבורית.דבר זה יגרום דווקא לעליית מספר כלי הרכב בכבישים ולא הפחתתם.
- **חשיפה לפרצות ומתקפות סייבר-** שליטה על מערכת הניווט, סנכרון מדיה, ביצוע שיחות טלפון, המזגן מתחיל לעבוד מעצמו, לוח המחוונים מתחיל להבהב ולייצר התרעות שונות ומשונות, חוסר יכולת לצאת מהרכב מכיוון שהדלתות ננעלות, שליטה על מערכות בקרת כלי הרכב, השבתת הבלמים והפסקת פעולת המנוע, הפעלת כריות אוויר תוך נסיעה מהירה, הקרנת תמונות המציגות דמויות על הכביש או על שלטי חוצות אשר אלגוריתמי ניתוח תמונה

ממצלמות המכונית מזהים אותם כבני אנוש ובעקבות כך נעצרת נסיעת הרכב או פונה לנתיב אחר, השתלטות על רכב המוביל חומרים מסוכנים ופיצוצו במקום הומה אדם.

מנגנונים חכמים:

על הרכב האוטונומי מותקנים מגוון רחב של חיישנים העוזרים לחקות את הראיה האנושית, המוח האנושי, לקלוט את מפת הסביבה ולבצע חישובים רלוונטים כגון מרחק מעצמים, מהירות וכו'...

LIDAR: חיישני לייזר שמזהים אובייקטים שנמצאים לפני הרכב ומכוונים אותו. LIDAR הוא שילוב של המילים אור – LIGHT ומכ"ם – RADAR . בכלי רכב אוטונומיים משתמשים ב LIDAR כדי למפות מכשולים בתוואי הנסיעה וכדי לנווט את הרכב בצורה בטוחה על הכביש.

טכנולוגיית מדידת המרחק של ה- LIDAR מבוססת על הארת המטרה בקרן לייזר.

ה- LIDAR משגר בעצמו את האור הממוקד לכמה כיוונים בסביבת המכונית, הקרניים פוגעות בעצמים סביב המכונית וחלקן מוחזרות מהם אל החיישן לצורך חישוב מרחק העצם מהרכב. כמו כן הוא מחזיר ענן נקודות מהסביבה המנותח ע"י אלגוריתם לפילוח ענן נקודות וכך מזהים מכשולים בדרך, שפת מדרכה, גדרות ביטחון ורכבים אחרים. אין לו בעיה בחושך והוא טוב ממצלמה באור חזק, אבל הוא לא רואה צבעים. החיישן מסוגל למדוד את המרחק של הרכב מעצמים בסביבתו בדיוק של עד שני סנטימטר באמצעות הבזקי אור קצרים שמשכם חלקיקי שנייה.

חיישני LIDAR משלימים את אמצעי הצפייה והזיהוי ומאפשרים לרכב להישאר בנתיב ולבלום מייד כשמאותר מכשול. בנוסף, גלגלי המכונית מצוידים בחיישני אולטרה-סאונד שמזהים מכשולים גם בגובה הקרקע ומאפשרים לרכב לחנות בבטחה.

מעבד האות: מעבד האות הוא מרכיב קריטי בביצועי המכ"ם , ותפקידו לבצע עיבודים מתמטיים כמו חילוץ האות מתוך רעשי רקע, או להפריד את האנרגיה הנקלטת לפי תדרי הדופלר כדי לחלץ את האות החוקי מהמטרה.

מצלמה- מצלמה משוכללת היא חלק ממערך החיישנים ומסוגלת לזהות רמזורים ואת צבע האור ברמזור, תמרורים בסביבה, סימונים על הכביש, הולכי רגל, מכשולים ואף לחשב את המרחק אליהם. אולם היא רגישה לאור חזק ומתקשה גם בחשיכה. המצלמות, ובכללן ההיקפיות, משמשות לרוב לזיהוי אובייקטים במרחקים קצרים יחסית ובסיטואציות לא מורכבות במיוחד. כדי לחשב את המרחק היחסי מהמכונית שנוסעת לפני הרכב האוטונומי, מחשב המצלמה מסיק את המרחק באמצעות גיאומטריה. הוא לוקח בחשבון את אורך המוקד שבו היא מצולמת, את הגובה מפני הקרקע שהמצלמה מותקנת ואת המרחק היחסי של גלגלי הרכב המצולם. פרמטרים אלה מאפשרים לחשב את המרחק בין המצלמה למכונית שהיא מצלמת.

באמצעות אלגוריתם yolo-המנתח תמונה בזמן אמת ניתן לזהות את העצמים במרחב.

מכ"ם- מכ"ם אינו רגיש למזג אוויר אבל סובל מדיוק נמוך. הרדאר, המבוסס על גלי RF , מיועד לזהות אובייקטים בטווחים הרחוקים ובמזג אוויר קשה, בעיקר באזור המצוי לפני הרכב.

יצרני הרכב בחרו להפעיל את המכ"מים בתדר 77 גיגה הרץ, ושואפים לשפר את הרזולוציה שיכולה ממש לייצר תמונה. זה קריטי כדי לזהות אופנוע בכביש מהיר, גם כשהוא מתמזג מבחינת החיישנים עם רכב אחר, וכדי לזהות הולכי רגל ואופניים בקרבת מכוניות. האתגר הכי קשה של הרדארים הוא לזהות הולכי רגל ורוכבי דו-גלגלי, גם על רקע כלי רכב אחרים. כמו כן, לזהות את הגובה של חפץ במרחק של 150 מטר, כדי לדעת אם זו פיסת קרטון שלא מפריעה לנו או ארגז שלא נרצה להתנגש בו.

בעולם קיימים רכבים בעלי רמות שונות של אוטונומיה:

- **רמה 0: ללא אוטומציה בנהיגה –** הנהג מחויב בכל פעולות הנהיגה, כפי שאנו מכירים זאת בעשורים האחרונים. המערכת האוטומטית מציגה אזהרות ועלולה להתערב לפרקי זמן קצרים, אך אין לה שליטה מתמדת על הרכב. אין לו אפילו בקרת שיוט, ובוודאי לא בקרת שיוט אדפטיבית.
- **רמה 1: סיוע לנהג** – קיימת ברכב מערכת סיוע נקודתית לנהג. כלי רכב ברמה 1 יצאו לשוק החל מסביבות שנת 2010. הנהג והמערכת האוטומטית חולקים שליטה ברכב. לדוגמה מערכת שבה הנהג שולט על ההיגוי, והמערכת האוטומטית שולטת על כוח המנוע כדי לשמור על מהירות מוגדרת (בקרת שיוט), כדי לשמור על המהירות ולשנותה (בקרת שיוט מסתגלת), סיוע בחניה, שמירה על נתיב הנסיעה ובלימת חירום למניעת התנגשויות. בישראל קיימת כבר מתאריך 1.1.2018 חובת התקנת מערכת בטיחות של בקרת סטייה מנתיב והתרעת שמירת מרחק.
- **רמה 2: אוטומציה חלקית** – מכונית שבה פעילות שתי מערכות או יותר של בטיחות ואוטונומיה, אשר מסוגלות, למשל, לטפל בו זמנית בהיגוי בתאוצה ובבלמים. יצאו לשוק החל משנת 2016, וברמה זו נדרש עדיין הנהג לבחון את תנאי הדרך ולהגיב בהתאם. על הנהג לפקח על הנהיגה ולהיות מוכן להתערב בכל עת אם המערכת האוטומטית. אפשר לעקוב אחר עיניו של הנהג באמצעות מצלמות כדי לאשר שתשומת ליבו מופנית לנסיעה.

  למערכות החיישנים החכמים ולמצלמות ההיקפיות ברמה זו של אוטומציה חלקית מתווספות מערכות נרחבות יותר, שנכללים בהן חיישנים המקיפים את השטח המת בשדה הראייה, מערכות התרעה על עקיפה מימין ואפילו אמצעי חניה אוטומטיים. לעיתים קרובות הרכב מצויד בבקרת שיוט ובשליטה חלקית על ההיגוי והבלימה.
- **רמה 3: אוטומציה מותנת** – המערכות של המכונית ברמה זו בשלות מספיק כדי שהנהג לא יידרש לפקח על תנאי הדרך "עיניים סגורות", אולם נדרש להיות זמין לנהיגה במקרה חירום. ברמה זו מערכות משולבות נהג-מחשב. הרכב יטפל במצבים המצריכים תגובה מיידית כגון בלימת חירום. שווק לראשונה ע"י חברת אודי באוקטובר 2018. בספר הרכב של הדגמים כתוב במפורש שהתכונות המופעלות בדגם דורשות פיקוח אקטיבי של הנהג ולא הופכת את הרכב לאוטונומי. אפשר

לחשוב על המערכת האוטומטית כעל נהג שותף שמתריע באופן מסודר לנהג שתורו לנהוג. הנהג יכול להפנות את תשומת ליבו בבטחה ממשימות הנהיגה. הוא יכול לשלוח הודעות טקסט או לצפות בסרט. המערכת עומדת בתקנות הבינלאומיות של מערכת לשמירת נתיבים אוטומטית – ALKS . מערכות השליטה ברכב עוברות ממחשב הרכב אל הנהג על פי המקרה, מפגעים בכביש ותנאי מזג אוויר. לדוגמא, בתנאים שאינם מצריכים תשומת לב מיוחדת, כגון כביש מהיר, יוכל המחשב להיכנס לפעולה, עם זאת לעיתים קרובות עדיין נדרשת התערבות אנושית.

- **רמה 4: אוטומציה ברמה גבוהה** – יכולת נהיגה אוטונומית מלאה בתנאים מסוימים או בנסיבות מיוחדות. יש צורך בנהג זמין למצבי חירום ואת תשומת ליבו לבטיחות. נהיגה עצמית של הרכב נתמכת רק באזורים מרחביים מוגבלים. מחוץ להם על הרכב להיות מסוגל לתפעל את הנסיעה ואת החניה בבטחה. רמה 4 מתאימה לדוגמא למונית רובוטית או שירות משלוחים רובוטי הפועלים במיקומים מוגדרים. מאחר וברמה זו הנהג יכול לשלוט על מערכות הרכב, נשאלת השאלה המשפטית – מתי המכונית הורתה לו להתערב, עד כמה בכלל יש לו שיקול דעת והאם מלכתחילה ההתערבות הייתה נחוצה. ברמה זו מחשב הרכב יכול לפקד על כל משימות הנהיגה. הוא יעבור למצב זה בשטחים ללא תנועה כבדה של רכבים והולכי רגל או בכבישים מהירים. בכל שלב הנהג יכול לשלוט בכל משימות הנהיגה.
- **רמה 5: אוטומציה מלאה** – הרכב יהיה מסוגל להחליף את הנהג בכל תנאי הדרך, ואין צורך במעורבות נהג למצבי חירום. מדובר באמת בטייס אוטומטי שמסוגל לשלוט לבדו במכונית. דוגמה לכך היא מונית רובוטית שעובדת בכל תנאי מזג האוויר ובכל הדרכים ללא מגבלות. הנהג לא חייב לשבת ברכב. התערבותו אינה נדרשת כלל, ולכן אין צורך בהגה או בדוושות. אפשר להגדיר תרחישים קבועים ולשלוח את המכונית עם הילדים לבית הספר, להזמין את הרכב לאיסוף ממקום מסוים ועוד משימות. השליטה ברכב נעשית באמצעות טכנולוגיות LOT שמאפשרות להזמין אותו לכל נקודה דרך אפליקציה ייעודית.

כיום ניתן לתכנת רכב אוטונומי בעל אוטומציה ברמה גבוהה (רמה 4). וזאת כיוון שעם כל הטכנולוגיה המתקדמת והמשוכללת, עדיין יש מחסור במכשירים משוכללים מספיק שיוכלו לעמוד בכל התנאים הנדרשים.

לדוגמא:

- **מצלמה-**

✓ עדשות מצלמות שאינן נקיות מעבירות אינן באמינות את המידע הוויזואלי.

✓ למצלמות יש מגבלות הדומות לאלה של העין האנושית, כלומר הן עשויות "לראות" גרוע יותר למשל בשלג או גשם עזים, ערפל סמיך, סופות אבק כבדות ופתיתי שלג. בתנאים כאלה, התפקודים של מערכות תלויות מצלמה עלולים להיות מופחתים

במידה משמעותית או להתנתק באופן זמני. אור מתקרב בעצמה גבוהה, השתקפויות בכביש המהיר, שלג או קרח על פני הכביש, משטחי כביש מלוכלכים או סימוני נתיבים לא ברורים יכולים גם הם להפחית במידה משמעותית את תפקוד המצלמה כשמשתמשים בה לסריקת הכביש לזיהוי הולכי רגל, רוכבי אופניים, בעלי חיים גדולים וכלי רכב אחרים.

- **לידאר-**

✓ ביצועי LiDARיכולים להיות מושפעים מתנאי מזג האוויר, כגון גשם, ערפל ושלג. זה יכול להגביל את השימוש בו ביישומים חיצוניים באקלים מסוימים.

- **רדאר-**

✓ היכולת של יחידת הרדאר לגלות רכב לפנים פוחתת במידה רבה אם המהירות של הרכב שלפנים שונה במידה רבה מהמהירות של המכונית שלך.

✓ שדה ראייה מוגבל-במצבים מסוימים כלי רכב אחר אינו מזוהה, או שהזיהוי נעשה מאוחר מהצפוי.

✓ במקרה של גשם כבד, או שלג או קרח על הסמל, ייתכן שתהיה הפחתה בתפקודי יחידת הרדאר, הם יושבתו כליל או יספקו תגובת תפקוד שגויה.

תיאור הפרויקט:

- **מבנה הפרויקט:**

הפרויקט נכתב בשפות:

python 3.9 המתאימה במיוחד לאלגוריתמים של למידת מכונה כמו yolo, OpenCV, coco.

C++ לכתיבת האלגוריתם.

כמו כן מורץ על סביבת Anaconda דרך Visual code .

- **קלט מהמשתמש:**

אני קולטת מהמשתמש את נקודת היעד שאליה הוא מעוניין לנסוע.

- **חיישנים:**

בפרויקט שלי נשתמש בחיישנים הבאים:

מצלמה- על הרכב יותקנו מצלמות ב-4 כיוונים.

קדימה-כדי לראות כל הזמן את המרחב מלפני הרכב.

אחורה- כדי לראות כל הזמן את המרחב מאחורי הרכב.

מימין- כדי לראות כל הזמן את המרחב הימני של הרכב.

משמאל- כדי לראות כל הזמן את המרחב השמאלי של הרכב.

המצלמות יהיו מיוחדות עבור רכב אוטונומי- שמצלמות גם בלילה ע"י שימוש באינפרה אדום.

IMU– מכיל מד תאוצה, גירוסקופ-מודד מהירות זוויתית, שהיא קצב השינוי של מיקום או סיבוב זוויתי. הוא מזהה תנועות סיבוב סביב שלושת הצירים ומספק נתונים על קצב הסיבוב של האובייקט, כך נוכל לנווט את הרכב במהירות המותרת עפ"י חוקי התנועה וכן לדעת לאיזה כיוון הרכב נוסע.

GPS – חיישן לאיתור המיקום הנוכחי של הרכב. חיישן זה מדויק רק ב-96%.

- הפרויקט יעשה תוך הנחה מראש של חוסר התחשבות בבלתמי"ם-מצבים בלתי מזוהים ובלתי צפויים.

בעיה אלגוריתמית:

- **פתיחת פרויקט:**

יצירת פרויקט חדש ב– Anaconda עם Python 3.9.

התקנת כל הספריות הדרושות לפרויקט.

התקנת כל התוספים למחשב כגון:Cuda, Pytorch..

- **רקע**:

כדי לקבל את כל הרקע על מכוניות אוטונומיות: איך הם בנויות, מאיזה חיישנים הם מורכבות, איזה אלגוריתמים מפעילים אותם אני אעשה קורס ב- coursera על מכוניות אוטונומיות וכל המידע הדרוש לי עליהם.

- **קבלת מידע מהחיישנים:**

מידע על הסביבה-

כדי לקבל מידע שוטף ועדכני על כל המרחב סביב הרכב האוטונומי והעצמים הנמצאים בו, נצטרך חיישן שבעצם יחקה את הראיה האנושית של הנהג. לצורך כך נתקין מצלמות ב-4 כיוונים: קדימה, אחורה ומשני הצדדים. שקולטות למרחק של עד 10 מטר.

כדי לדעת מה קורה בתמונות ולפעול בהתאם ננתח את המרחב באמצעות כמה סוגים של אלגוריתמים וספריות:

<u>אלגוריתם yolo</u>- שמתאים במיוחד לניתוח תמונות עבור רכב אוטונומי.

בחרתי את yolo v5 כיוון שיש לו תוצאות מדויקות יותר וטובות יותר מ-yolo v3 .

אלגוריתם yolo הוא אלגוריתם שמנתח תמונות בזמן אמת.

לאלגוריתם מכניסים dataset של חפצים/תמונות/דברים שמעוניינים לזהות אותם מתוך התמונה בזמן אמת.

מערכי נתונים ניתן להוריד מאתר Kaggle או ממקומות אחרים.

עבור המכונית לפרויקט שלי בחרתי לזהות באמצעות yolo :

רמזורים- רמזור אדום, רמזור ירוק, תמרורי כיוון מעל הרמזור שאומרים לאן עלי לפנות.

תמרור עצור- שמסמן לרכב לעצור.

לאחר הכנסת מערך הנתונים, מתחיל שלב האימון של האלגוריתם-train.
מגדירים את קטגוריות הדברים שברצוננו לזהות, לדוגמא: רמזור אדום ימינה-1, רמזור אדום שמאלה-2 וכו'... ומתחילים להריץ.
באמצעות תיבות סימון בצבעים שונים המתאימים לכל קטגוריה, נסמן כל פעם את אזור החפץ בתמונה. ככל שנאמן את האלגוריתם על מגוון גדול יותר של תמונות הוא ינפיק לנו בזמן אמת את התוצאות הנכונות ביותר עם פחות טעויות בזיהוי.
לאחר מכן עושים לאלגוריתם test-מבחן ובודקים מה רמת הדיוק שלו.
עבור מכונית חשוב שיהיו לנו תוצאות מדויקות ביותר כדי שלא יגרמו תאונות, ולכן יש לאמן אותו עד לדיוק מרבי.
בשביל ביצוע שלב זה יש ללמוד היטב את אלגוריתם yolo v5 ואת דרכי העבודה איתן.
כמו כן חיפוש מערך נתונים המכיל כמות נכבדת של תמונות העונות על הקריטריונים הנ"ל.
<u>ספריית OpenCV-</u> Opencv (ספריית ראיית מחשב בקוד פתוח) היא ספריית תוכנת ראייה ממוחשבת ולימוד מכונה בקוד פתוח שפותחה ע"י intel. היא מספקת מגוון רחב של פונקציות ואלגוריתמים לעיבוד תמונה ווידאו, זיהוי תכונות, זיהוי אובייקטים ועוד.
לספריה זו פונקציות מיוחדות ביניהם פונקציה שיכולה לקלוט צבעים מסוימים מתמונה ולסמן את האזור בצבע. נעשה שימוש בפונקציה זו כדי לזהות את הנתיב הנוכחי שהרכב נוסע בו. זה יעיל ביותר כיוון שלמחשב קל לקלוט את שילוב הצבעים של הנתיב-שחור לבן וע"י סינון הרעש מתמונה- כל מה שלא עונה על הצבע הזה, ועוד כמה פעולות נוכל לקבל את אזור הנתיב באופן ממש מדויק.
לצורך זה יש לכתוב אלגוריתם לזיהוי נתיב המשתמש בספריה זו, ולאמן אותו על תמונות של נתיבים ולבדוק שזה אכן מזהה בצורה מדויקת וטובה.
<u>ספריית COCO-</u> ספריית COCO (Common Objects in Context) היא מערך נתונים ומדד הערכה בשימוש נרחב עבור משימות זיהוי, פילוח וכתוביות של אובייקטים בראייה ממוחשבת. הוא מספק מערך נתונים בקנה מידה גדול עם תמונות מוערות, תיבות תוחמות אובייקט, מסכות פילוח וכיתובים. מערך הנתונים של COCO מכיל קבוצה מגוונת של אובייקטים נפוצים הנמצאים בסצנות יומיומיות, מה שהופך אותו למתאים לאימון והערכת מודלים של ראייה ממוחשבת .ספריית COCO כוללת גם מדד הערכה הנקרא COCO AP (Average Precision), המודד את הדיוק של אלגוריתמי זיהוי אובייקטים ופילוח. מדד זה לוקח בחשבון ערכי דיוק וזיכרון כדי לספק הערכה מקיפה של ביצועי המודל.
ספריית COCO כבר מאומנת בזיהוי של אנשים, רכבים ואופנועים ולכן אני אעדיף להשתמש בה.

מה שנותר לעשות הוא לחפש קוד שמזהה רכבים, אופנועים ואנשים באמצעות ספריית COCO ולהטמיע אותו בפרויקט.

כאשר נרצה לקבל מידע על סביבת הרכב בצד מסוים, נגש למצלמה שנמצאת בצד הרצוי על הרכב, ננתח את הנתונים באמצעות האלגוריתמים שהזכרנו, והרי לנו מפה מדויקת של סביבת הרכב.

מרחק- (מותנה במידה ואני יספיק, אחרת אני מסתמכת על כ שהמכונית נמצאת בתוך מערכת שמנהלת מרחק בטוח מכלי הרכב סביבה באופן אוטומטי)

כדי לדעת מה המרחק של הרכב מהרכבים שבסביבת הרכב או ממכשולים נשתמש בחיישן לידאר שמחזיר לנו את המרחק של העצם מהרכב.

חיישן זה שולח גלי אור ומחזיר ענן נקודות של הסביבה.

אפשר לנתח ענן זה ע"י אלגוריתמים שונים כמו: פילוח ענן נקודות ולהגיע למיפוי כולל של הסביבה.

אני מעדיפה לזהות את העצמים בסביבה דרך מצלמות, ולהשתמש בלידאר לזיהוי מרחק. עבור כל נקודה מהענן נקודות שמחזיר הלידאר יש לנו 3 פרמטרים:

המרחק של העצם מהמכונית והמיקום בציר ה-X ובציר ה-Y קואורדינטות קווי הרוחב והאורך של העולם.

באמצעות מהירות האור הידועה, שהיא קבועה, חיישן הלידר מחשב את המרחק לעצם על ידי הכפלת זמן הטיסה במהירות האור ולאחר מכן חלוקתו בשניים (מאז הדופק נוסע אל האובייקט ובחזרה).

לצורך כך יש ללמוד התעסקות בסיסית עם ענן נקודות שפולט לידאר,

וכן לכתוב קוד שמקבל את הנתונים מהחיישן ומחזיר לי את המרחק מהעצם לרכב.

מהירות-

נתקין על הגלגלים **IMU-** הפלט של IMU הוא מהירות נוכחית של הגלגל בזווית מסויימת. וכך נוכל לדעת ע"י מספר חישובים, בכל זמן נתון מה המהירות הנוכחית של הרכב, וכן האם צריך להאט או להגביר את המהירות.

חקירה איך עובד מד מהירות ומה הפלט שלו.

כתיבת קוד שהפלט מקבל את הפלט של מהירות זוויתית של הגלגל ומחזיר מהירות נוכחית של הרכב.

מיקום וניווט-

זיהוי מיקום נוכחי של הרכב יעשה באמצעות חיישני GPS.

חיישן GPS מספק מידע מיקום גיאוגרפי מדויק. הוא משתמש באותות מלוויינים מרובים כדי לקבוע את קואורדינטות קווי הרוחב והאורך של המכונית.

ניתן להשתמש במידע זה כדי לאתר את המיקום המדויק על פני כדור הארץ.

באמצעות אלגוריתם מסנן קלמן נוכל לשערך את המיקום הנוכחי בצורה אופטימלית של הרכב.

כמו כן כדי לדעת לאן לנווט את הרכב נצטרך לעשות שימוש במידע שמגיע מספריות ה Google Maps ולדעת בכל שלב לאן הרכב צריך להמשיך.

- **חישוב מסלול:**

בחלק זה יכנס גם הצד לקוח של המשתמש.

למשתמש יהיה מסך שבו הוא יקליד את כתובת היעד שהוא מעוניין לנסוע אליו.

נעשה זאת באמצעות התחברות לGoogle Maps.

Google מאפשרת לנו שימוש בספריית Google Maps שמכילה את כל מפות הדרכים של הכבישים בעולם.

כמו כן מציעה מגוון רחב של ספריות שונות שמציעות ממשק משתמש אטרקטיבי ומנגד גם ספריות שמסייעות למתכנת-כמו חישוב המסלול הקצר ביותר מכתובת המוצא לכתובת היעד תוך התחשבות בגורמים משתנים כמו: עמס בכבישים, תאונות, חסימות כבישים וכו', זה נעשה ע"י שימוש באלגוריתמים שונים של גרפים כמו דייקסטרה וכו'...

יש להשיג מפתח API זהו מפתח שע"י ניתן להתחבר לGoogle Maps דרך קוד.

זה יהיה פרויקט מסוג- WEB כיוון שזה ממשק משתמש.

כמו כן ללמוד הייטב את ספריות Google Maps.

לאחר מכן-כתיבת קוד שקולט מהמשתמש את נקודת היעד, ניתן לו גם אפשרות של השלמה אוטומטית ותיבת חיפוש, מזהה מיקום נוכחי של הרכב באמצעות חיישני GPS, ומחב את המסלול הקצר ביותר לכתובת היעד.

בנוסף מציג על המסך את מפת הדרכים ומסמן את המסלול שעל הרכב לנסוע.

- **היתוך חיישנים:**

היתוך חיישנים ברכב אוטונומי מתייחס לתהליך של שילוב נתונים ממספר חיישנים כדי לקבל הבנה מדויקת ואמינה יותר של סביבת הרכב. בכלי רכב אוטונומיים מותקנים בדרך כלל חיישנים שונים, כגון מצלמות, מכ"ם, לידר וחיישנים קוליים, כל אחד מספק סוגים שונים של מידע על הסביבה.

אלגוריתמי היתוך חיישנים לוקחים את הנתונים הגולמיים מחיישנים אלו ומשלבים אותם יחד כדי ליצור תפיסה מקיפה והוליסטית של הסביבה. על ידי שילוב החוזקות של חיישנים שונים ופיצוי על המגבלות האישיות שלהם, היתוך חיישנים משפר את יכולות התפיסה של הרכב ומאפשר קבלת החלטות חזקה יותר.

תהליך ההיתוך כולל יישור וסנכרון של נתוני החיישן, הסרת רעשים וחריגים, כיול מדידות חיישנים, ולאחר מכן איחוי הנתונים יחד באמצעות טכניקות שונות כמו סינון,

מעקב אחר אובייקטים, מיפוי ושיטות אומדן הסתברותיות.
נתוני החיישן המותכים המתקבלים מספקים ייצוג מדויק יותר של הסביבה, ומאפשרים לרכב האוטונומי לתפוס טוב יותר עצמים, לזהות מכשולים, לעקוב אחר עצמים נעים, להעריך את מיקומם ומהירותם ולקבל החלטות מושכלות לגבי ניווט, תכנון נתיב, הימנעות מהתנגשות ופונקציות נהיגה אוטונומית אחרות. היתוך חיישנים ממלא תפקיד קריטי בשיפור האמינות הכוללת, הבטיחות והביצועים של כלי רכב אוטונומיים.
החלטתי לעשות את היתוך החיישנים באמצעות אלגוריתם מסנן קלמן מורחב-EKF.
מסנן קלמן זהו אלגוריתם שמקבל מידע מהחיישנים של המכונית כגון: מיקום ומהירות, אך כיוון שהמידע שמגיע לא נכון במאה אחוז, וכן לכל חיישן יש מצבים מסוימים שבהם הוא לא מספיק מדויק, או לא עובד בכלל, יש צורך לבצע חישובים ע"י סטטיסטיקה ופונקציות של מודל התנועה וע"י סנכרון המידע נוכל לשערך בצורה הנכונה ביותר הנתונים מהחיישנים וכך לנווט את הרכב בצורה הטובה ביותר.
הוא מתבסס על הרעיון של הסתברות בייס- שבהינתן מאורע וסיגמא ניתן לחשב מאורעות נוספים.
לביצוע שלב זה:
למידת קורס אלגוריתמי ניווט ושיערוך מקום.
חקירה יסודית של מסנן קלמן.
כתיבת פונקציית מסנן קלמן רגילה.
שדרוג הפונקציה להיתוך חיישנים.
בדיקה שהפונקציה עובדת הייטב.
פונקציה זאת היא בעצם הפונקציה הבסיסית והכי חשובה של הרכב.

בכל זמן נתון הפונקציה מאגדת לנו את כל המידע הנדרש, ע"י כיול והיתוך החיישנים

וכך נוכל לשלוף ממנה את המידע שאנחנו צריכים בצורה נוחה ונגישה.

- **פונקציית ההפעלה:**

פונקציית ההפעלה מאגדת את כל האלגוריתמים השונים והמידע העצום שמגיע מהחיישנים.
היא בעצם "המוח" של הרכב, במקום המוח של הנהג.
פונקציה זו כוללת בתוכה את החלטות הנהיגה- מה עושים בכל מקרה, מתי לנסוע ומתי לעצור.
היא תהיה מחולקת כך:
<u>ניווט</u>: פונקציית ההפעלה צריכה לכלול אלגוריתמים לתכנון וביצוע מסלול המכונית.

זה כרוך בהתחשב בגורמים כמו יעד, תנאי תנועה, חוקי כביש וכלי רכב אחרים.
קביעת המיקום המדויק של הרכב- יעשה באמצעות מסנן קלמן, קבלת מידע מחיישן- GPS ו-Google Map.

<u>תפיסה</u>: תכלול את המידע המגיע מהאלגוריתמים למשימות תפיסה, כגון זיהוי אובייקטים, זיהוי נתיב, זיהוי כלי רכב, זיהוי תמרורים וזיהוי הולכי רגל. זה מאפשר למכונית להבין את סביבתה ולקבל החלטות מושכלות.
יעשה באמצעות מסנן EKF וכתיבת פונקציות בשפת C++.

<u>בקרה</u>: פונקציית הבקרה צריכה לכלול אלגוריתמי בקרה לטיפול בתאוצה, בלימה והיגוי של המכונית. אלגוריתמים אלו מבטיחים שהמכונית מתמרנת בצורה בטוחה וחלקה בהתבסס על המידע המגיע מפונקציית התפיסה והמסלול המתוכנן.
כתיבת פונקציות בשפת C++.

<u>קבלת החלטות</u>: פונקציית ההפעלה צריכה לכלול אלגוריתמי קבלת החלטות הלוקחים בחשבון גורמים שונים, כמו מצב המכונית הנוכחית, תנאי הדרך, חוקי התנועה והתנהגות משתמשי דרך אחרים. אלגוריתמים אלו קובעים את פעולות המכונית, כגון החלפת נתיב, עקיפה או עצירה בצמתים.

באמצעות מסנן EKF נוכל לדעת את המידע המדויק באותו זמן על החיישנים- כי הוא עושה היתוך חיישנים.

ניצור טבלת SQL. לטבלה נכניס בכל פעם את מה שהתקבל מפונקציית התפיסה:
הולך רגל בכביש, רמזור אדום, לפנות ימינה וכו', עפ"י עדיפות מסוימת,
הפונקציה תשלוף ממסד הנתונים את שם הפונקציה שצריך לבצע במקרה הנ"ל,
או שיש צורך לעבור לבדיקות נוספות במסד נתונים אחר ותבצע אותם.
טבלאות ה– data base ינהלו את סדר העדיפות ודרכי הפעולה של הרכב.

תהליכים עיקריים בפרויקט:

מקבלים מהמשתמש את נקודת היעד אליה הוא רוצה להגיע.
מחשבים את המסלול הקצר ביותר לנקודת היעד.
מתחילים לנסוע.
במשך הנסיעה- קולטים עם החיישנים מידע על הסביבה, עצמים, מרחק, מהירות וכו'...
ומעבדים אותו באמצעות האלגוריתמים השונים.
מעבירים את המידע לפונקציית קבלת ההחלטות שמחזירה לנו את שם הפונקציה שאנחנו צריכים להפעיל כדי שהרכב יתפקד כנדרש.
מפעילים את הפונקציה.
מבצעים את השלבים שוב ושוב עד שמגיעים אל היעד!

הסתייגויות:

- הפרויקט עובד על כבישים חד סיטריים או עם 2 נתיבים בלבד.
- הפרוייקט לא עוסק בהתעסקות עם מצבי בלתמ"ים.
- כאשר אני צריכה לקבל מידע מלידאר- אני מפעילה אותו למיקום ספציפי והוא מחזיר לי את המרחק. אני מניחה שאני יכולה להפעיל למיקום מסויים וכן שלא יופיע לי במהלך המדידה אובייקט שלא צפיתי.
- כאשר אני מזהה רמזור אני מניחה שאני צריכה להאט ולעצור.
- המכונית לא תעבוד במזג אוויר קיצוני.
- לא התייחסתי לכיכרות.
- זיהוי הנתיב לא יעבוד על כבישים מפותלים במיוחד.

2. מבוא

2.1 הרקע לפרוייקט

אני מאוד מתחברת לנושא של טכנולוגיה ופיתוחים טכנולוגים חדשים, כאשר נדרשתי לחשוב על רעיון לפרויקט מאוד רציתי משהו שיעניין אותי לעבוד עליו.

לאחר מחשבה החלטתי שאני עושה רכב אוטונומי למרות כל הקשיים, אני מוכנה לעבוד חזק בשביל זה. מאז ומעולם עניין אותי נהיגה ואני חושבת שפרויקט זה הזדמנות מצוינת לשלב תכנות עם דברים שאוהבים. רכב אוטונומי זהו רכב שלרוב פועל בכוחות עצמו ואינו נזקק לעזרתו של הנהג. מספר רעיונות קדמו לפיתוח רכב אוטונומי. אחת הסיבות הייתה כדי לחסוך בתאונות דרכים. בגלל שהכל יתנהל בצורה מסודרת עפ"י חוקי התנועה היות והרכב מציית להוראות המתכנת שלו. הבעיה העיקרית בנושא היא שדברים שלא מתוכנתים כגון: הולכי רגל יכולים להשפיע על התנהגות הרכב ולעיתים יש צורך לעשות ההפך מחוקי התנועה כדי לשמור על חיי אדם.

המכונית האוטונומית משתמשת בחיישנים שונים שקולטים מידע על הסביבה, מעבדים אותו באמצעות אלגוריתמים מתקדמים של בינה מלאכותית כגון yolo ומנווטים את המכונית בהתאם תוך שמירה על בטיחותו של הנהג ובטיחות הסובבים אותו.

בין החיישנים: מד מהירות על הגלגל, GPS, מצלמות, לידאר ורדאר.

רכב אוטונומי בעצם מחקה את המוח האנושי הוא מקבל מידע מהסביבה כגון: איזה צבע רמזור, מזהה תמרורים, מזהה את הנתיב שהרכב אמרו לנסוע בו, מעברי חציה, מכוניות, אנשים....

ואז הוא מתנהל בהתאם.

לדוגמא: אם הרכב זיהה רמזור אדום הוא צריך להאט את הרכב למהירות 0, במקרה של ילד קופץ לכביש הוא חייב לעצור במהירות וכו'...

לאט לאט עם הזמן הרכב לומד ע"י למידת מכונה מה עליו לעשות במצבים קשים, למשל: כאשר יש סיכון שהרכב יתהפך או ח"ו לדרוס ילד..

הפרוייקט יקבל מידע מהסביבה בזמן אמת על עצמים ותפיסת הסביבה, ינתח אותו ויעביר אותו לפונקציית קבלת ההחלטות שמחליטה מה לעשות עם המידע שהגיע.

כמו כן יהיה צד לקוח שבו המשתמש מכניס את נקודת היעד אליה הוא רוצה להגיע וחישוב המסלול הקצר ביותר לנקודת היעד, כדי שהלקוח יגיע במהירות האפשרית ליעד שלו.

2.2תהליך המחקר

כיום קיימים בשוק רכבים אוטונומיים עם יכולת נהיגה אוטונומית מלאה (רמה 4). בתנאים מסוימים או בנסיבות מיוחדות יש צורך בנהג זמין למצבי חירום ואת תשומת ליבו לבטיחות. נהיגה עצמית של הרכב נתמכת רק באזורים מרחביים מוגבלים. מחוץ להם על הרכב להיות מסוגל לתפעל את הנסיעה ואת החניה בבטחה. רמה 4 מתאימה לדוגמא למונית רובוטית או שירות משלוחים רובוטי

הפועלים במיקומים מוגדרים. מאחר וברמה זו הנהג יכול לשלוט על מערכות הרכב, נשאלת השאלה המשפטית – מתי המכונית הורתה לו להתערב, עד כמה בכלל יש לו שיקול דעת והאם מלכתחילה ההתערבות הייתה נחוצה. ברמה זו מחשב הרכב יכול לפקד על כל משימות הנהיגה. הוא יעבור למצב זה בשטחים ללא תנועה כבדה של רכבים והולכי רגל או בכבישים מהירים. בכל שלב הנהג יכול לשלוט בכל משימות הנהיגה.

נכון לעכשיו, ניתן לתכנת רכבים אוטונומיים עד רמה 4 וזאת כיוון שעם כל הטכנולוגיה המתקדמת והמשוכללת, עדיין יש מחסור במכשירים משוכללים מספיק שיוכלו לעמוד בכל התנאים הנדרשים.

דוגמאות:

מצלמה-

- עדשות מצלמות שאינן נקיות מעבירות אינן באמינות את המידע הוויזואלי.
- למצלמות יש מגבלות הדומות לאלה של העין האנושית, כלומר הן עשויות "לראות" גרוע יותר למשל בשלג או גשם עזים, ערפל סמיך, סופות אבק כבדות ופתיתי שלג. בתנאים כאלה, התפקודים של מערכות תלויות מצלמה עלולים להיות מופחתים במידה משמעותית או להתנתק באופן זמני. אור מתקרב בעצמה גבוהה, השתקפויות בכביש המהיר, שלג או קרח על פני הכביש, משטחי כביש מלוכלכים או סימוני נתיבים לא ברורים יכולים גם הם להפחית במידה משמעותית את תפקוד המצלמה כשמשתמשים בה לסריקת הכביש לזיהוי הולכי רגל, רוכבי אופניים, בעלי חיים גדולים וכלי רכב אחרים.

לידאר-

ביצועי LiDARיכולים להיות מושפעים מתנאי מזג האוויר, כגון גשם, ערפל ושלג. דבר המגביל את השימוש בו ביישומים חיצוניים באקלים מסוימים.

רדאר-

- היכולת של יחידת הרדאר לגלות רכב לפנים פוחתת במידה רבה אם המהירות של הרכב שלפנים שונה במידה רבה מהמהירות של המכונית שלך.
- שדה ראייה מוגבל-במצבים מסוימים כלי רכב אחר אינו מזוהה, או שהזיהוי נעשה מאוחר מהצפוי.
- במקרה של גשם כבד, או שלג או קרח על הסמל, ייתכן שתהיה הפחתה בתפקודי יחידת הרדאר, הם יושבתו כליל או יספקו תגובת תפקוד שגויה.

3.2 סקירה ספרותית

לצורך לימוד תיאורטי:

www.new-techonline.com

www.zivud.com

www.data.gov.il

www.new-techonline.com

www.techtime.co.il

www.hamichlol.org.il

www.stayinformedgroup.com

www.fnx.co.il

dwo.co.il

www.digikey.co.il

wheel.co.il

https://bennyaviad.com/articles/vehicle-and-transportation-trends-and-innovations

www.mrcoral.co.il

למידת מכונה:

www.oracle.com

מסנן קלמן:

https://he.wikibooks.org/wiki/%D7%A4%D7%99%D7%96%D7%99%D7%A7%D7%94_%D7%AA%D7%99%D7%9B%D7%95%D7%A0%D7%99%D7%AA/%D7%9E%D7%9B%D7%A0%D7%99%D7%A7%D7%94/%D7%A7%D7%99%D7%A0%D7%9E%D7%98%D7%99%D7%A7%D7%94/%D7%9E%D7%A9%D7%95%D7%95%D7%90%D7%95%D7%AA_%D7%94%D7%AA%D7%A0%D7%95%D7%A2%D7%94

ניווט:

www.gov.il

https://developers.google.com/maps/documentation/javascript/overview?hl=he#Inline

https://softauthor.com/google-maps-api-realtime-tracking-javascript

https://www.gov.il/he/departments/legalInfo/speed_regulations

אלגוריתם yolo ולמידת מכונה:

https://www.youtube.com/watch?v=cPOtULagNnI&t=4858s

https://www.v7labs.com/blog/yolo-object-detection

https://hashdork.com/iw/%D7%A8%D7%A9%D7%AA-/%D7%A2%D7%A6%D7%91%D7%99%D7%AA

https://reshetech.co.il/machine-learning-tutorials/yolo-you-only-look-once-object-detection

https://www.classcentral.com/classroom/youtube-object-detection-on-custom-dataset-with-yolo-v5-fine-tuning-with-pytorch-and-python-tutorial-126510

https://www.v7labs.com/blog/mean-average-precision

https://www.analyticsvidhya.com/blog/2021/12/how-to-use-yolo-v5-object-/detection-algorithm-for-custom-object-detection-an-example-use-case

https://medium.com/@shahriar.rezghi.sh/using-yolo-in-c-55d55419a947

https://wandb.ai/onlineinference/YOLO/reports/YOLOv5-Object-Detection-on-Windows-Step-By-Step-Tutorial---VmlldzoxMDQwNzk4

חיישן רדאר:

https://engineering.tau.ac.il/Engineering-Faculty-radar-research-Prof.Ginzburg

קורסים:

https://www.coursera.org/learn/motion-planning-self-driving-cars

https://www.coursera.org/learn/visual-perception-self-driving-cars

https://www.classcentral.com/subject/autonomous

/https://campus.gov.il/course/ariel-acd-rfp4-ariel-nivut

https://backup.campus.gov.il/course/ariel-acd-rfp4-ariel-robot-he

דאטאסטים:

https://paperswithcode.com/dataset/bdd100k

https://www.kaggle.com/datasets/sakshaymahna/kittiroadsegmentation/download?datasetVersionNumber=1

זיהוי נתיב:

https://paperswithcode.com/task/lane-detection

לידאר וענני נקודות:

https://www.mysun.co.il/post/%D7%9E%D7%94%D7%95-%D7%A2%D7%A0%D7%9F-%D7%A0%D7%A7%D7%95%D7%93%D7%95%D7%AA

https://www.youtube.com/watch?v=JbUNsYPJK1U

אלגוריתם:

/https://cplusplus.com/reference/thread/thread

https://math.haifa.ac.il/ronn/advprog/unix_c/docs/htm/pthreads.htm

https://www.damada.co.il/topics/physics/db/kinematics_accilirate_velocity/kinematics_accilirate_velocity.shtml

https://docs.python.org/3/extending/extending.html

3.מטרות ויעדים

מטרת הפרוייקט היא לתכנת רכב אוטונומי שמקבל מידע ע"י חיישנים מידע על תפיסת הסביבה, מיקום של הרכב בזמן אמת ע"י חיישן GPS, מהירות של הרכב ע"י חיישן IMU.

להחלפת רכבים מאויישים ע"י נהגי ברכבים אוטונומיים יש מספר יתרונות:

- **פינוי שטחי חניה רבים לטובת בניה-** לפי הערכות עד 2030 עשרה אחוזים מהרכבים שמיוצרים בעולם יהיו אוטונומיים, וסביר להניח שעד 2035 מחצית מכלי הרכב כבר יהיו אוטונומיים. סביר להניח שמספר כלי הרכב הנעים בכבישים יופחת ורבים יוותרו על רכישת ואחזקת מכונית פרטית. השחרור מצורכי חניה יפנה מיליוני מטרים של נדל"ן.

מכונית אחת תוכל לשמש הרבה אנשים. וזאת בגלל שכאשר אדם צריך לנסוע, הוא מזמין מכונית, והמכונית הכי קרובה אליו בסביבתו תגיע אליו. כך שלו היה לו רכב הוא היה חונה המון שעות סתם ותופס שטח חניה לחינם.

מניעת תאונות וטעויות אנוש בנהיגה- על פי נתוני משרד התחבורה האמריקאי 9 מתוך 10 תאונות מתרחשות בשל טעויות נהג, אבל המחקר קובע כי כשליש מהתאונות נובע מצירוף נסיבות שגם תגובה מהירה ומדויקת לא הייתה יכולה למנוע את התרחשותן. למשל הולך רגל שקופץ פתאום לכביש קרוב מדי מכדי שרכב יספיק לבלום. או מצבים אחרים שבהם אי-אפשר היה לצפות את ההתנהגות של נהג אחר. בוחני המכון, שמבצעים גם מבחני ריסוק למכוניות חדשות, בחנו 5,000 תאונות.

לדבריהם, רכב אוטונומי לא יסבול מהסחת דעת, עייפות או שכרות שמהם סובל נהג אנושי, ותמיד יוכל להגיב במהירות למצבי חירום, כמו כן מצויית לחוקי התנועה, אבל תאונות רבות מתרחשות בנסיבות שגם מכ"ם, מצלמה וחיישנים אחרים לא יכולים לצפות, או דווקא בשל העובדה שלעיתים נדרשת חשיבה יצירתית ועקיפת החוקים דבר שקיים רק אצל בן אנוש.

- **שמירה על סביבה נקייה-**מכוניות אוטונומיות רבות מונעות על ידי מנועים חשמליים, המייצרים אפס פליטות בנקודת השימוש. זה מפחית את זיהום האוויר ואת פליטת גזי החממה, ועוזר להילחם בשינויי האקלים.

כמו כן, מכוניות אוטונומיות יכולות להשתמש באלגוריתמים מתקדמים כדי לייעל את המסלולים שלהן, להפחית עומס ופקקים ולשמור על מהירות נסיעה קבועה. זה עוזר למזער את צריכת הדלק והפליטה על ידי הימנעות מתנועת סרק מיותרת ותנועת עצירה וסע.

- **צמצום העומס בכבישים-** כל אדם שירצה לנסוע, יזמין אליו מכונית דרך האפליקציה והמכונית הפנויה בסביבתו תגיע אליו באופן אוטומטי. דבר זה יצמצם את כמות הרכבים כיוון שהרבה יעדיפו להשתמש במודל חדשני זה, ללא עלויות אחזקת הרכב.

4.אתגרים

התקנת ספריות מורכבות ב-python , מציאת הגירסה המתאימה של python וכן סביבת הרצה קלה ונוחה לשימוש עם ספריות מורכבות.

חילוק הפרוייקט הענק לתתי משימות קטנות.

למידת ספריות של Google maps וכתיבת קוד שמחשב מסלול קצר ביותר.

מסנן קלמן- להבין לעומק את הקורס שלמדתי, לחקור על זה עוד ולכתוב את זה בקוד- להבין בצורה יסודית ומעמיקה מה המסנן מקבל, מה הוא מחשב ומה הוא מחזיר ואז מה אני עושה עם הנתונים האלה.

התקנת כל התוספים והתוכנות למחשב בכדי להריץ את אלגוריתם yolo.

מציאת database של תמונות לאימון עבור yolo.

לתכנן בצורה נכונה ויסודית את כל התהליכים שמתבצעים בצורה אסינכרונית וכן לנהל גישה למשאבים משותפים.

5. מדדי הצלחה

- ניהול מוצלח של התהליכים שמתבצעים בצורה אסינכרונית
- התקנת הספריות המורכבות של python ומציאת הגרסה מתאימה לכל הספריות
- נתונים אמיתיים בסימולציה של החיישנים
- שימוש במבנה נתונים יעיל
- קוד נקי וברור
- הדמיה טובה של הפרוייקט ע"י הדפסה ל-console
- סנכרון נכון בין הנתונים שמתקבלים מהחיישנים
- פרויקט עובד
- זיהוי נכון של עצמים בסיבה
- זיהוי הנתיב של הרכב שלי גם כשהוא מתפתל
- חיבור ל-python , קבלת נתונים מ-python ושליחת נתונים ל-python מ-++C
- חיבור נכון ל- react ושליחת נקודת היעד ל-++C

6.רקע תאורטי על הפרוייקט

בתחילה נפתח צד לקוח. על המסך יש למשתמש אפשרות להכניס את נקודת היעד אליה הוא מעוניין לנסוע, לאחר שהוא מכניס הוא לוחץ על כפתור שמחשב את המסלול הקצר ביותר ליעד ושולח את ההוראות שלו לאלגוריתם של פונקציית קבלת ההחלטות ב-C++.

האלגוריתם מפעיל כל פעם את הפונקציה המתאימה ובו זמנית מפעיל את אלגוריתם yolo לזיהוי עצמים בסביבה, זיהוי נתיב, חיישן GPS, חיישן IMU ומפעיל פונקציות בהתאם.

האלגוריתם פועל עד שהמכונית מגיעה ליעדה.

7.מצב קיים

כיום קיימים בעולם רכבים בעלי רמות שונות של אוטונומיה:

- **רמה 0: ללא אוטומציה בנהיגה –** הנהג מחויב בכל פעולות הנהיגה, כפי שאנו מכירים זאת בעשורים האחרונים. המערכת האוטומטית מציגה אזהרות ועלולה להתערב לפרקי זמן קצרים, אך אין לה שליטה מתמדת על הרכב. אין לו אפילו בקרת שיוט, ובוודאי לא בקרת שיוט אדפטיבית.
- **רמה 1: סיוע לנהג** – קיימת ברכב מערכת סיוע נקודתית לנהג. כלי רכב ברמה 1 יצאו לשוק החל מסביבות שנת 2010. הנהג והמערכת האוטומטית חולקים שליטה ברכב. לדוגמה מערכת שבה הנהג שולט על ההיגוי, והמערכת האוטומטית שולטת על כוח המנוע כדי לשמור על מהירות מוגדרת (בקרת שיוט), כדי לשמור על המהירות ולשנותה (בקרת שיוט מסתגלת), סיוע בחניה, שמירה על נתיב הנסיעה ובלימת חירום למניעת התנגשויות. בישראל קיימת כבר מתאריך 1.1.2018 חובת התקנת מערכת בטיחות של בקרת סטייה מנתיב והתרעת שמירת מרחק.
- **רמה 2: אוטומציה חלקית** – מכונית שבה פעילות שתי מערכות או יותר של בטיחות ואוטונומיה, אשר מסוגלות, למשל, לטפל בו זמנית בהיגוי בתאוצה ובבלמים. יצאו לשוק החל משנת 2016, וברמה זו נדרש עדיין הנהג לבחון את תנאי הדרך ולהגיב בהתאם. על הנהג לפקח על הנהיגה ולהיות מוכן להתערב בכל עת אם המערכת האוטומטית. אפשר לעקוב אחר עיניו של הנהג באמצעות מצלמות כדי לאשר שתשומת ליבו מופנית לנסיעה. למערכות החיישנים החכמים ולמצלמות ההיקפיות ברמה זו של אוטומציה חלקית מתווספות מערכות נרחבות יותר, שנכללים בהן חיישנים המקיפים את השטח המת בשדה הראייה, מערכות התרעה על עקיפה מימין ואפילו אמצעי חניה אוטומטיים. לעיתים קרובות הרכב מצויד בבקרת שיוט ובשליטה חלקית על ההיגוי והבלימה.
- **רמה 3: אוטומציה מותנת** – המערכות של המכונית ברמה זו בשלות מספיק כדי שהנהג לא יידרש לפקח על תנאי הדרך "עיניים סגורות", אולם נדרש להיות זמין לנהיגה במקרה

חירום. ברמה זו מערכות משולבות נהג-מחשב. הרכב יטפל במצבים המצריכים תגובה מיידית כגון בלימת חירום. שווק לראשונה ע"י חברת אודי באוקטובר 2018. בספר הרכב של הדגמים כתוב במפורש שהתכונות המופעלות בדגם דורשות פיקוח אקטיבי של הנהג ולא הופכת את הרכב לאוטונומי. אפשר לחשוב על המערכת האוטומטית כעל נהג שותף שמתריע באופן מסודר לנהג שתורו לנהוג. הנהג יכול להפנות את תשומת ליבו בבטחה ממשימות הנהיגה. הוא יכול לשלוח הודעות טקסט או לצפות בסרט. המערכת עומדת בתקנות הבינלאומיות של מערכת לשמירת נתיבים אוטומטית – ALKS . מערכות השליטה ברכב עוברות ממחשב הרכב אל הנהג על פי המקרה, מפגעים בכביש ותנאי מזג אוויר. לדוגמא, בתנאים שאינם מצריכים תשומת לב מיוחדת, כגון כביש מהיר, יוכל המחשב להיכנס לפעולה, עם זאת לעיתים קרובות עדיין נדרשת התערבות אנושית.

- **רמה 4: אוטומציה ברמה גבוהה** – יכולת נהיגה אוטונומית מלאה בתנאים מסוימים או בנסיבות מיוחדות. יש צורך בנהג זמין למצבי חירום ואת תשומת ליבו לבטיחות. נהיגה עצמית של הרכב נתמכת רק באזורים מרחביים מוגבלים. מחוץ להם על הרכב להיות מסוגל לתפעל את הנסיעה ואת החניה בבטחה. רמה 4 מתאימה לדוגמא למונית רובוטית או שירות משלוחים רובוטי הפועלים במיקומים מוגדרים. מאחר וברמה זו הנהג יכול לשלוט על מערכות הרכב, נשאלת השאלה המשפטית – מתי המכונית הורתה לו להתערב, עד כמה בכלל יש לו שיקול דעת והאם מלכתחילה ההתערבות הייתה נחוצה. ברמה זו מחשב הרכב יכול לפקד על כל משימות הנהיגה. הוא יעבור למצב זה בשטחים ללא תנועה כבדה של רכבים והולכי רגל או בכבישים מהירים. בכל שלב הנהג יכול לשלוט בכל משימות הנהיגה.
- **רמה 5: אוטומציה מלאה** – הרכב יהיה מסוגל להחליף את הנהג בכל תנאי הדרך, ואין צורך במעורבות נהג למצבי חירום. מדובר באמת בטייס אוטומטי שמסוגל לשלוט לבדו במכונית. דוגמה לכך היא מונית רובוטית שעובדת בכל תנאי מזג האוויר ובכל הדרכים ללא מגבלות. הנהג לא חייב לשבת ברכב. התערבותו אינה נדרשת כלל, ולכן אין צורך בהגה או בדוושות. אפשר להגדיר תרחישים קבועים ולשלוח את המכונית עם הילדים לבית הספר, להזמין את הרכב לאיסוף ממקום מסוים ועוד משימות. השליטה ברכב נעשית באמצעות טכנולוגיות LOT שמאפשרות להזמין אותו לכל נקודה דרך אפליקציה ייעודית.

8.ניתוח חלופות מערכתי

בתחילה הפרוייקט היה נראה לי הר ענק שאין דרך לעבור אותו.

בשלב ראשון התחלתי לחקור את הנושא של הפרוייקט, קראתי מאמרים ונכנסתי לכל מיני אתרים כדי ללמוד ולהבין איך עובד רכב אוטונומי ומה אמור להיות בו.

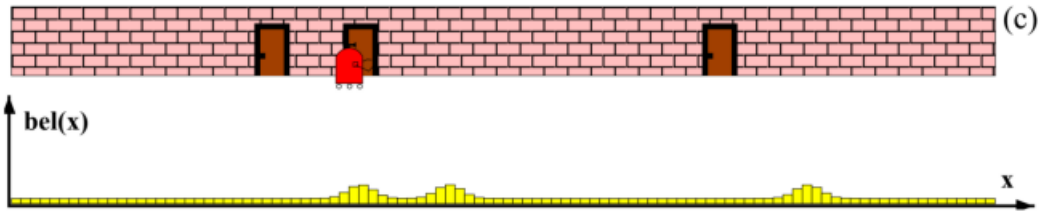
כשראיתי שעוד לא קיבלתי מספיק ידע ברמה שאני יכולה לדעת מאיפה להתחיל את הפרוייקט, נרשמתי למספר קורסים ב-coursera ושם למדתי על קינמטיקה של הרכב, אלו חיישנים מרכיבים את הרכב, ומה התפקיד של כל אחד מהם, על אלו דברים צריך לשים דגש כשניגשים לכתוב את הקוד וכו'...

אחרי שהרקע היה ברור לי למדתי קורס אלגוריתמי ניווט ושיערוך מיקום באוניברסיטת אריאל כדי שאוכל לעקוב אחרי המיקום של הרכב בזמן אמת ולסנכרן את הנתונים עם מה שמגיע מחיישן GPS כיוון שבמקומות סגורים כגון חניונים ומנהרות חיישני ה-GPS לא עובדים ויש צורך לשערך בצורה עצמאית את המיקום של הרכב כדי לנווט אותו.

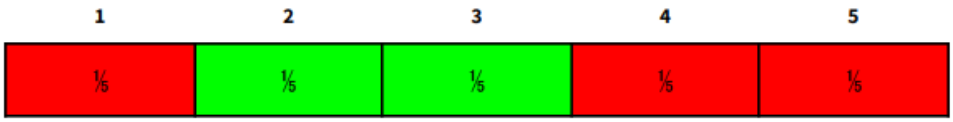
עשיתי גם קורס על רובוטים אוטונומיים כדי להבין את הרעיון של רובוטיקה ולקבל על זה קצת רקע.

ואז התחלתי את הפרוייקט. בחרתי להתחיל עם החלק של השיערוך מיקום.

שיערוך מיקום הוא טכניקה המשמשת ברובוטיקה ולוקליזציה להערכת המיקום והכיוון של רובוט בסביבה המבוססת על מדידות חיישנים ומודלים של תנועה.
בתחילה אלגוריתם ניווט באסיאני שמחלק את המרחב למשבצות ובודק מה ההסתברות להיות בכל משבצת.
נניח שזה המרחב שלנו:



נניח והעולם שלנו מחולק ל5- משבצות, אז ההסתברות להיות בכל תא היא 1/5 כלומר:



כעת נניח ולרובוט שלנו יש חיישן שמזהה ירוק ב,100%- וקיבלנו חיווי לירוק אז ההסתברות לכל משבצת היא:

1	2	3	4	5
0	½	½	0	0

כעת נניח שהחיישן מזהה ירוק ב-3/5 ואדום ב-1/5 , מה נעשה עכשיו ?נשתמש בחישוב על פי הסתברות בייס.

$$\Pr(H \mid E) = \frac{\Pr(E|H)\cdot\Pr(H)}{\Pr(E)}$$

נסמן ב-H ההשערה שהרובוט בתא ירוק ו-E שקיבלנו מהחיישן שאנחנו בתא ירוק. סה"כ קיבלנו:

$$\Pr(H \mid E) = \frac{\Pr(E|H)\cdot\Pr(H)}{\Pr(E)} = \frac{0.6\cdot 0.4}{0.36} = \frac{2}{3}$$

ניתן לשערך בצורה טובה איפה הרובוט נמצא, אבל בעולם האמיתי כאשר הרכב זז, הוא לא יודע בוודאות אם הוא זז בכלל וכמה הוא זז לכן הרכב ירצה לזוז בעולם בצורה הסתברותית .אם נחזור לדוגמה של המשבצות, כאשר הרובוט יזוז על המשבצות הרבה זמן, יכולות להתחיל להיות טעויות קטנות בזמן התנועה. ולכן הבנתי שזה לא מתאים עבור רכב אוטונומי אז עברתי לנסות את מסנן היסטוגרמה.

מסנן היסטוגרמה משערך מיקום מצויין עבור מימד אחד אך כאשר מוסיפים לו ממדים, הבעיה נהיית מורכבת חישובית בצורה מעריכית (אקספוננציאלית) ועובר רכב אני צריכה מסנן דו-מימדי שישערך לי מיקום בציר ה-X ובציר ה-Y.

כמו כן מסנן היסטוגרמה הוא בגדול מסנן בדיד -בדוגמאות שראינו, המשמעות היא שהרובוט יכול להיות רק באחת מתוך המשבצות האפשריות. יתרה מזאת, אם גודל המשבצת הוא 10 מ"ר, אזי ההסתברות להיות בכל נקודה בתוך השטח היא שווה. ואילו רכב יכול להיות גם על השטח שבין לבין כי הוא נוסע.

אז עברתי לבדוק מהו מסנן קלמן.

הגדרה:

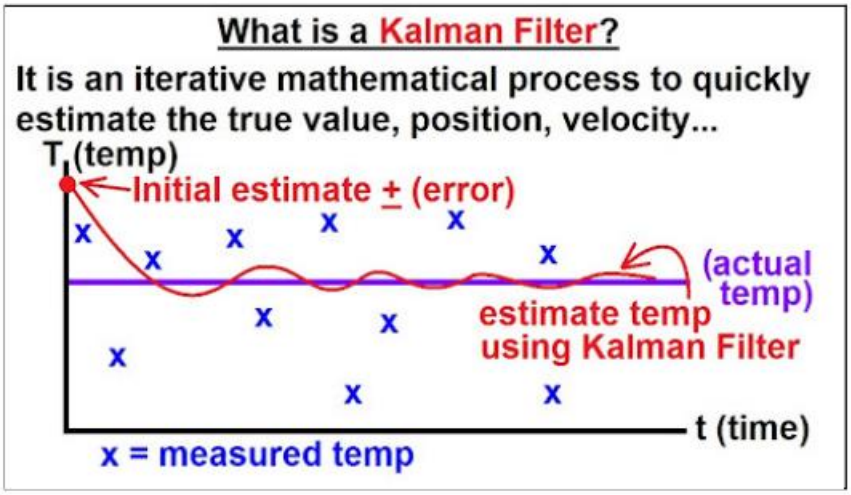
- מסנן קלמן זה תהליך מתמטי איטרטיבי שמשתמש בסט של משוואות ונתונים שמגיעים ברציפות מהחיישנים כדי לשערך במהירות את הערך האמיתי של אובייקט, כאשר הערכים הנמדדים מכילים חוסר וודאות, רעש ואקראיות
- ניתן לחשוב על מסנן קלמן כעל ממוצע משוקלל בין המדידה הנוכחית מהחיישן ובין הידע הקודם שלנו לגבי מצב המערכת

- הרכיב שקובע את המשקל של כל אחד מהרכיבים הוא הגבר קלמן.

יתרונות של מסנן קלמן:

1.שיערוך עקיף - שיערוך על פי כמה פרמטרים שונים כמו מהירות ומיקום

2. היתוך חיישנים - תהליך מתמטי של חיבור מידע מחיישנים שונים.



המסנן שיערך לי בצורה נכונה וטובה ולכן בחרתי להשתמש בו.

מסנן קלמן זהו אלגוריתם שמקבל מידע מהחיישנים של המכונית כגון: מיקום ומהירות,

אך כיוון שהמידע שמגיע לא נכון במאה אחוז, וכן GPS אינו קולט בחניונים, קניונים או ליד בניינים גבוהים מאוד יש צורך לבצע חישובים ע"י סטטיסטיקה ופונקציות של מודל התנועה וע"י סנכרון המידע נוכל לשערך בצורה הנכונה ביותר את המיקום של הרכב וכך לנווט אותו בצורה הטובה ביותר. הוא מתבסס על הרעיון של הסתברות בייס- שבהינתן מאורע וסיגמא ניתן לחשב מאורעות נוספים.

המסנן עובד בצורה כזו:

נגדיר למעלה את המטריצות הבאות:

F- מכילה את משוואות התנועה

DT- חוסר הוודאות של התנועה

H- מסמנת את מרחב החיישן- איזה חיישנים פועלים

R- מטריצת הרעש של המערכת

I-מטריצת היחידה

B- מטריצת הבקרה של שינויים לא ידועים של המערכת-פיתאומיים

המסנן מקבל:

```
def filter(x,P, u, measurements):
```

X- זהו וקטור המצב-מכיל את הפרמטרים שמעוניינים לבדוק במערכת: מיקום ומהירות

בשלב ראשון היא מאותחלת מיקום בציר X=0, מיקום בציר Y=0, מהירות=0

$$\vec{x}_k = \begin{pmatrix} Position \\ Velocity \end{pmatrix}$$

P- מטריצת חוסר הוודאות-הסיגמא.
באלכסון הראשי שלה יש את הסיגמא של כל חיישן בריבוע.
זוהי מטריצה סימטרית: $X_{I,J}=X_{J,I}$ המכילה את חוסר הוודאות של המערכת- עד כמה החיישנים לא מדויקים וכו'...

היא נראית כך:

$$P = \begin{pmatrix} \sigma_{pos}^2 & \sigma_{pos}\sigma_{vel} \\ \sigma_{vel}\sigma_{pos} & \sigma_{vel}^2 \end{pmatrix}$$

U-וקטור הבקרה של שינויים ידועים של המערכת

MEASURMENT- המדידות מהחיישנים. כל מדידה תכלול מיקום ומהירות.

```
for Z in range(len(measurements)):
```

עוברים בלולאה על כל הווקטור של המדידות ובכל פעם מבצעים את הפעולות הבאות:

<u>שלב ה-prediction/(ניבוי מיקום הרכב):</u>

```
X =np.dot(F,x)+np.dot(B,u)
```

- נכפיל את מטריצת F – משוואות התנועה, במיקום ומהירות הנתונים
- נוסיף את ההכפלה של מטריצת B עם ווקטור U ששניהם מכילות את חוסר הוודאות שיש להתחשב בו בחישוב המקום של הרכב.

$$X_k = F \cdot X_{k-1} + B \cdot \vec{u}$$

כעת יכנס ל-P ממוצע השיערוך החדש של המיקום ומהירות.

```
P =np.dot(np.dot(F,P),F.T)
```

- נכפיל את מטריצת F -משוואות התנועה בשערוך החדש ואז נכפיל במטריצת F.TRANSPOZE()

כדי לקבל את מטריצת הסיגמא החדשה.

$$P_k = F_k \cdot P_{k-1} \cdot F_k^T + Q_k$$

<u>שלב ה- Measurement/(עדכון מיקום הרכב):</u>

```
S =R + np.dot(H, np.dot(P,H.T))
K = np.dot(np.dot(P,H.T), np.linalg.inv(S))
```

- חישוב הגבר קלמן- מעבר ממרחב המצב למרחב החיישן H עם תוספת רעש R.

$$\left(\begin{array}{c} K = \dfrac{P_K \cdot H^T}{H \cdot P_{k_p} \cdot H^T + R} \\ Y_k = X_{k_p} + K \cdot \left[Z_k - H \cdot X_{k_p} \right] \end{array} \right)$$

```
y =Z-np.dot(H,X)
```

- קבלת המדידה החדשה במרחב החיישן.

$$Y_1 = C \cdot Y_{1_m} + z_m$$

```
x=x+np.dot(k,np.linalg.inv(y-np.dot(H,x)))
```

- חישוב מצב נוכחי

$$X_1 = X_{1_p} + K \cdot \left[Y_1 - H \cdot X_{1_p} \right]$$

```
P =np.dot(I-np.dot(K,H),P)
```

- עדכון מטריצת covariance.

$$P_k = (I - K \cdot H) \cdot P_k$$

לאחר שסיימתי לכתוב את הקוד מצאתי ספרייה בפייתון בשם pykalman שמחשבת את מסנן קלמן,
אך ברמה נמוכה יותר.

```
from pykalman import KalmanFilter
```

ניתן ללמוד על ספרייה זו בקישור: https://pykalman.github.io/

לאחר מכן החלטתי לעבור לזיהוי הנתיב. בתחילה חשבתי להשתמש באלגוריתם yolo לזיהוי הפסים הלבנים המגבילים את הנתיב ואז לשערך באיזה מיקומים על הכביש זה במציאות ע"י חישובים מתמטיים ואז לאפשר למכונית לנסוע בין 2 הגבולות.

זה דרש חישובים ברמה גבוהה כמו כן לבצע כיול מצלמה וחשב את המרחק של המצלמה מהעצם שהיא פוגשת ולכן חיפשתי דרך אחרת לזהות את הנתיב.

מצאתי באתר Github קוד לזיהוי נתיב שהפלט שלו הוא סרטון שמסמן את הנתיב הנוכחי של הרכב בירוק ומחזיר את מרכז הנתיב לכל פריים.

הקוד משתמש בספריית OpenCV- Opencv (ספריית ראיית מחשב בקוד פתוח) היא ספריית תוכנת ראייה ממוחשבת ולימוד מכונה בקוד פתוח שפותחה ע"י intel. היא מספקת מגוון רחב של פונקציות ואלגוריתמים לעיבוד תמונה ווידאו, זיהוי תכונות, זיהוי אובייקטים ועוד. לספריה זו פונקציות מיוחדות ביניהם פונקציה שיכולה לקלוט צבעים מסוימים מתמונה ולסמן את האזור בצבע. נעשה שימוש בפונקציה זו כדי לזהות את הנתיב הנוכחי שהרכב נוסע בו. זה יעיל ביותר כיוון שלמחשב קל לקלוט את שילוב הצבעים של הנתיב-שחור לבן וע"י סינון הרעש מתמונה- כל מה שלא עונה על הצבע הזה, ועוד כמה פעולות נוכל לקבל את אזור הנתיב באופן ממש מדויק.

הורדתי את הקוד וניסיתי להריץ אותו בפייתון, אבל היה שם הרבה ספריות מורכבות שלא הצלחתי להתקין והיו לי שגיאות. אז התחלתי לפתור את השגיאות ושחיפשתי אודות שגיאה מסוימת היה כתוב שאולי הסיבה היא שלא מותקן לי על המחשב python בגרסה מתאימה.

ניסיתי ובדקתי את כל סוגי ה-python והגעתי למסקנה ש- python3.9 מתאים לכל הספריות שאני צריכה. אבל משום מה זה לא רץ לי על ה- pycharm שהיה מותקן לי על המחשב.

החלפתי גרסאות אבל זה לא עזר. עד שמצאתי ב- google שקיימת סביבת הרצה וירטואלית בשם Anaconda . התקנתי אותה על המחשב ואז הרצתי את הקוד והוא ב"ה עבד.

אחרי שהזיהוי נתיב היה מוכן, עברתי לזיהוי עצמים. מאוד נהנתי מהחלק הזה.

כשלב ראשון היה להחליט אלו עצמים הרכב יזהה, בהכוונת המורה הוחלט שהרכב יזהה את המקרים הבאים: רמזור אדום, רמזור אדום ישר, רמזור אדום ימינה, רמזור אדום שמאלה, רמזור ירוק ישר, רמזור ירוק ימינה, רמזור ירוק, רמזור ירוק שמאלה, מעבר חצייה, תמרור עצור, תמרור הגבלת מהירות ל-80.

א"ח בהוראת המורה התחלתי לחקור על אלגוריתם שמנתח נתונים בזמן אמת.

לשם כך נבחר אלגוריתם yolo. בחרתי את yolo v5 כיוון שיש לו תוצאות מדויקות יותר וטובות יותר מ-yolo v3 . לאלגוריתם מכניסים dataset של חפצים/תמונות/דברים שמעוניינים לזהות אותם מתוך התמונה בזמן אמת.

לאחר הכנסת מערך הנתונים, מתחיל שלב האימון של האלגוריתם-train.

מגדירים את קטגוריות הדברים שברצוננו לזהות, לדוגמא: רמזור אדום ימינה-1, רמזור אדום שמאלה-2 וכו'... ומתחילים להריץ.

באמצעות תיבות סימון בצבעים שונים המתאימים לכל קטגוריה, נסמן כל פעם את אזור החפץ בתמונה. ככל שנאמן את האלגוריתם על מגוון גדול יותר של תמונות הוא ינפיק לנו בזמן אמת את התוצאות הנכונות ביותר עם פחות טעויות בזיהוי.

לאחר מכן עושים לאלגוריתם test-מבחן ובודקים מה רמת הדיוק שלו.

עבור מכונית חשוב שיהיו לנו תוצאות מדויקות ביותר כדי שלא יגרמו תאונות, ולכן יש לאמן אותו עד לדיוק מרבי.

בתחילה לא היו תוצאות טובות לאימון, אבל אימנתי אותו כמה פעמים עד לקבלת תוצאות טובות.

לאחר זה, התחלתי לחשוב על הניווט.

בשלב ראשון צריך למצוא את המסלול הקצר ביותר לנקודת היעד. חיפשתי ומצאתי יש ספריות של Google maps שמחשבות את זה.

רציתי לעשות שהתצוגה תתעדכן בזמן אמת על המיקום שלי אבל זה ספריות שעולות כסף אז העדפתי שלא כי זה גם ככה לא הרעיון המרכזי בפרוייקט שלי.

מה שנשאר הוא ללמוד עליהם . למדתי וכתבתי קוד ב-web זה השפת צד לקוח שהכרתי, א"ח הבנתי שעדיף לכתוב ב– React אז שיניתי את הקוד במקמות הנדרשים כך שיתאים ל-React.

מה שהייתי צריכה מספריות אלו זה בעצם את קאורדיננטות המיקומים לאורך הנסיעה ואז להעביר אותם למסנן קלמן שיבצע חישוב של המיקום המשוערך יחד עם המיקום שהתקבל מחיישן ה-GPS ואת זה להעביר לפונקציית קבלת ההחלטות.

כשסיימתי את זה והתחלתי לכתוב את פונקציית קבלת ההחלטות ב-C++ חשבתי עם המורה על צורה לנתח את זה, בעצם קיבלתי מערך בגודל עצום של כל הקאורדיננטות שהרכב צריך לנסוע בהם. כדי לנתח את זה צריך לדעת כל כמה מילישניות להתקדם קאורדיננטה ואיך נדע מתי זה פניה ימינה/ שמאלה רק עפ"י מספרים של קאורדיננטות, ואז המורה לא הסכימה לי להיכנס לזה כי הפרוייקט שלי גם ככה גדול. אז חיפשתי אפשרות אחרת לחשב את המסלול הקצר. ומצאתי ששינוי קטן בקוד שלי יביא לי הוראות במילים ואת המרחק במטרים.

לדוגמא: פנו **שמאלה** לכיוון **הרקפות** (29מ')

אחרי כל זה סוף סוף התחלתי בכתיבת האלגוריתם.

אלגוריתם yolo מחזיר נתונים או בסרטון עם תיבות תוחמות על העצמים שהוא זיהה ואת שם העצם עם אחוז הדיוק של הזיהוי, או בקובץ אקסל שעל כל פריים הוא פותח קובץ txt ורושם שם את מה שהוא זיהה.

בתחילה עשיתי אלגוריתם שמחפש את התקיה האחרונה שהאלגוריתם יצר, פותח את קובץ האקסל וקורא משם את הנתונים.

לאחר שהרצתי התברר שקיימת כאן בעיית הקוראים כותבים ש"א לקרוא ולכתוב לקובץ באותו זמן.

אז היה צורך לשנות את הקוד ועבדתי עם קבצי טקסט שנוצר עבור כל פריים.

בתחילה חשבתי לעשות גם לידאר שיזהה מרחק לעצם וכן בין רכב לרכב כדי לאפשר החלפת נתיב בטוחה. כשחקרתי איך עושים את זה גיליתי שישנם מס' אלגוריתמים שעושים את זה,

אך כדי לזהות מרחק מעצם יש צורך לזהות מהו מתוך מפת עננים נקודות שמתקבלת כפלט מהחיישן, זה הצריך עבודה רבה במיוחד אחרי שאלגוריתם yolo היה מוכן והמורה אמרה לי לא לעשות את זה בפרוייקט, כי לא יהיה לי זמן לגמור אותו, אז במקום זה עשיתי פונקציה שמקבלת מערך נקודות ובוחרת את הנקודה המינימלית בהסתמך על זה שהנקודה המינימלית היא האובייקט הרצוי.

בסוף כשראיתי שאני לא עושה החלפת נתיב אז הורדתי גם את זה.

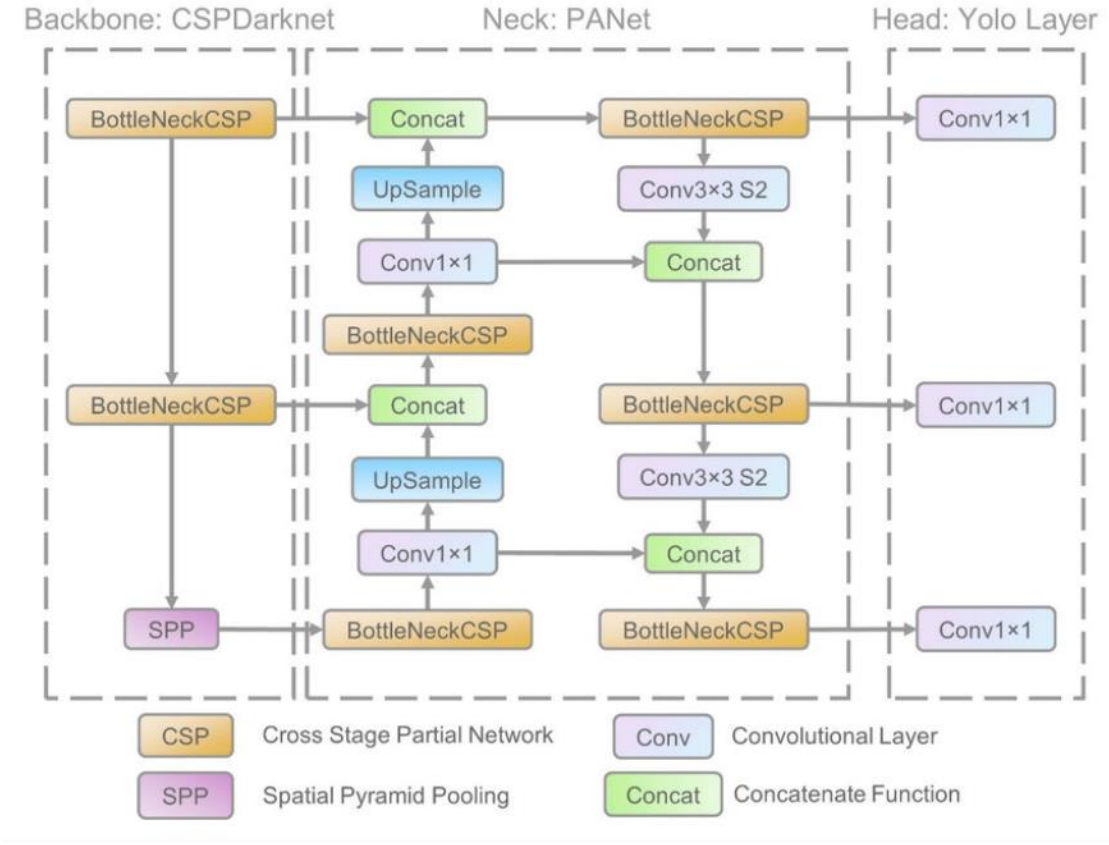
את החלק של זיהוי הנתיב לא הצלחתי לחבר לפרוייקט. ניסתי המון סוגים שונים של קודים ושימוש בספריות שונות מה שהתברר בסוף זה שא"א להפעיל מ-C++ קובץ Jupiter וכשניסיתי להעביר אותו לpython זה גם לא עבד.

להמשיך לחפור לא היה לי זמן אז הוא לא מחובר לפרוייקט אבל הוא עובד.

כשחשבתי איך לעשות את האלגוריתם של הניווט וקבלת ההחלטות רציתי לעשות פונקציה עבור כל מצב שתופעל במקרה שהיא אמורה לפעול. לדוגמא: פונקציה בשם GreenLightLeft תופעל במקרה שאלגוריתם yolo זיהה רמזור ירוק לכיוון שמאל וגם זה הכיוון שהרכב צריך לפנות אליו. ואז ראיתי שחבל לעשות את זה כך כי זה עושה כפילות של קוד כיוון שבמקרה של רמזור ירוק ימינה או ישר צריך לבצע את אותו דבר רק בשינוי הכיוון. אז צמצמתי את הפונקציות שיכלתי כדי שיהיו יותר גנריות.

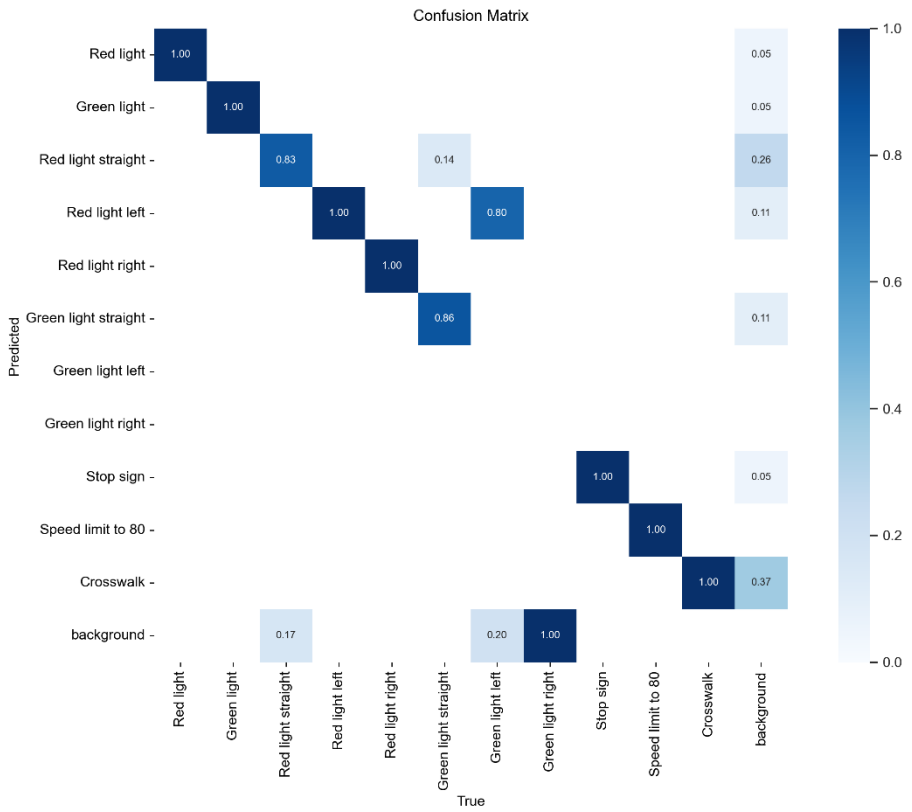
9. תיאור החלופה הנבחרת

נתחיל מאלגוריתם yolo . yolo הוא ראשי תיבות של You Only Look Once. אני השתמשתי בגרסה 5, שהושקה על ידי Ultralytics ביוני 2020 וכעת היא האלגוריתם המתקדם ביותר לזיהוי אובייקטים. זוהי רשת עצבית קונבולוציונית חדשה (CNN) המזהה עצמים בזמן אמת בדיוק רב. האלגוריתם משתמש ברשת עצבית אחת כדי לעבד את התמונה כולה, ואז מפריד אותה לחלקים ומנבא תיבות תוחמות והסתברויות עבור כל רכיב. תיבות תוחמות אלו משוקללות לפי ההסתברות הצפויה. השיטה "חיים רק פעם אחת" על התמונה במובן זה שהיא עושה תחזיות לאחר התפשטות אחת קדימה בלבד דרך הרשת העצבית. לאחר מכן הוא מספק פריטים שזוהו לאחר דיכוי לא מרבי (מה שמבטיח שאלגוריתם זיהוי האובייקט מזהה כל אובייקט פעם אחת בלבד).

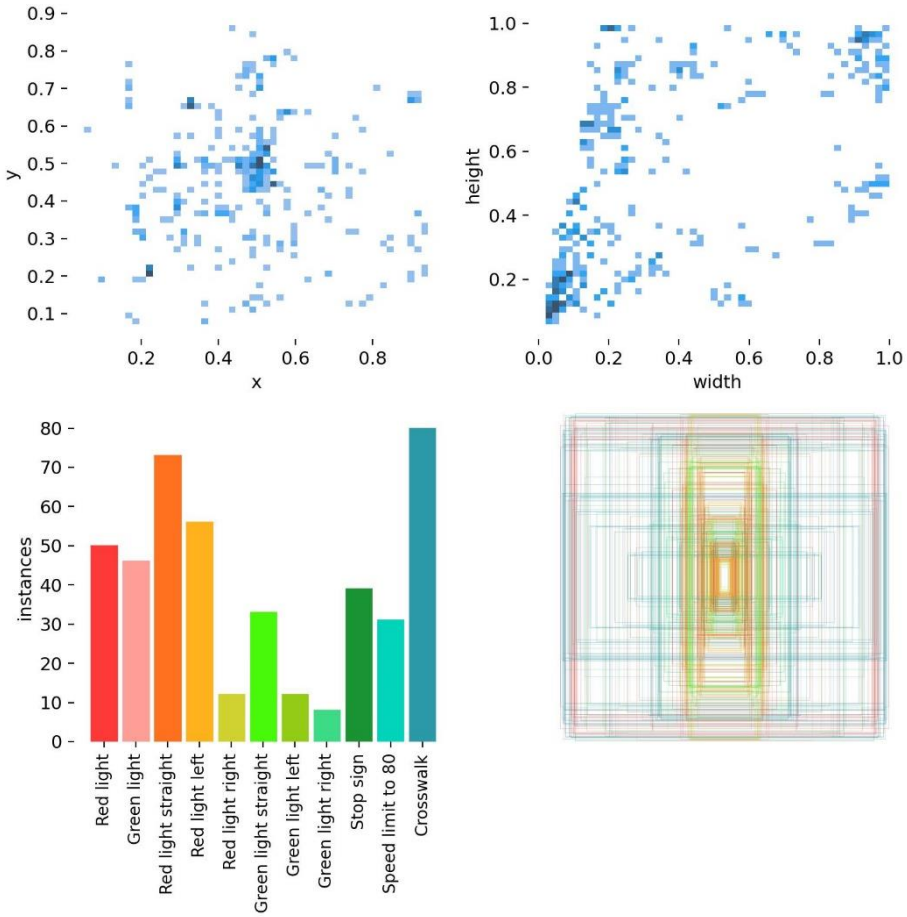


בשלב האימון קיבלתי את מטריצת הבלבול. מטריצת בלבול היא כלי להערכת ביצועים המשמש להערכת הדיוק של זיהוי אובייקטים. זוהי טבלה המשמשת לעתים קרובות לתיאור הביצועים של מודל סיווג על קבוצה של נתוני בדיקה שעבורם ידועים הערכים האמיתיים.

באלגוריתם YOLO, מטריצת הבלבול יכולה לעזור להבין עד כמה המודל מתפקד במונחים של זיהוי נכון של אובייקטים בתמונה. זה מציג את מספר התוצאות החיוביות האמיתיות, השליליות האמיתיות, החיוביות השגויות והשליליות השגויות עבור כל מחלקה של אובייקטים שזוהה. על ידי ניתוח מטריצת הבלבול, מפתחים יכולים לכוונן את המודל כדי לשפר את הדיוק והביצועים שלו.



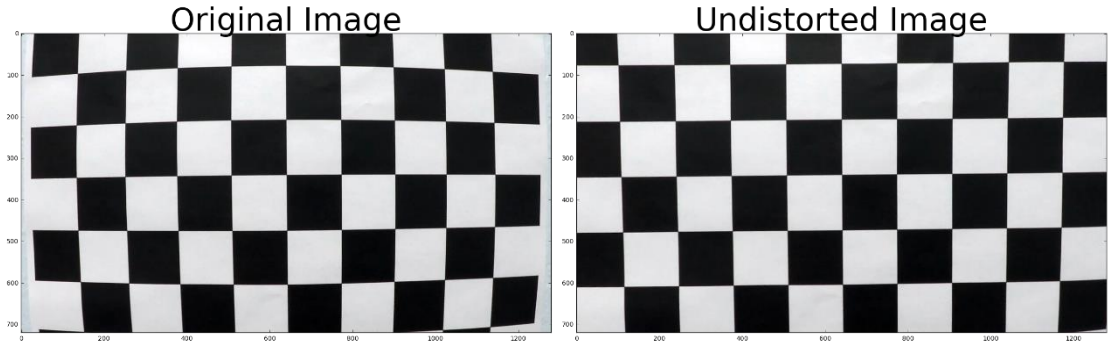
המקרים שהאלגוריתם אומן עליהם והתגיות שלהם:



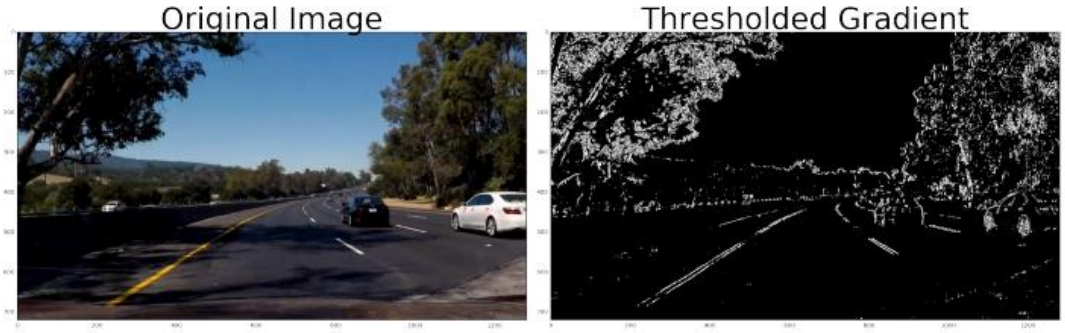
דוגמא להרצה לאחר האימון:

נעבור לאלגוריתם זיהוי הנתיב. האלגוריתם מורץ על סביבת Anaconda .

הוא מבצע כיול מצלמה בשיטה מעניינת. מצלמים לוח שחמט של משבצות שחור לבן ונותנים לו מטריצה של ריבועים שחור לבן. את ההפרש הקעור בין התמונה למטריצה האלגוריתם מזהה ומכייל את המצלמה לפי זה:

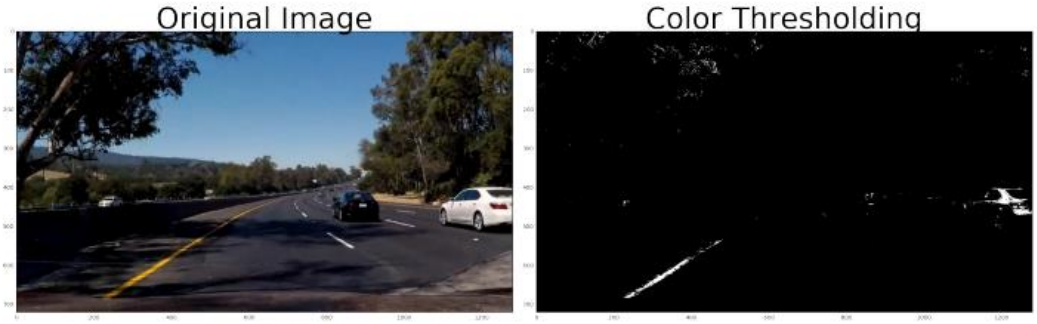


הופך את תמונת הנתיב לשחור לבן:



מחדד את התמונה שנוצרה:

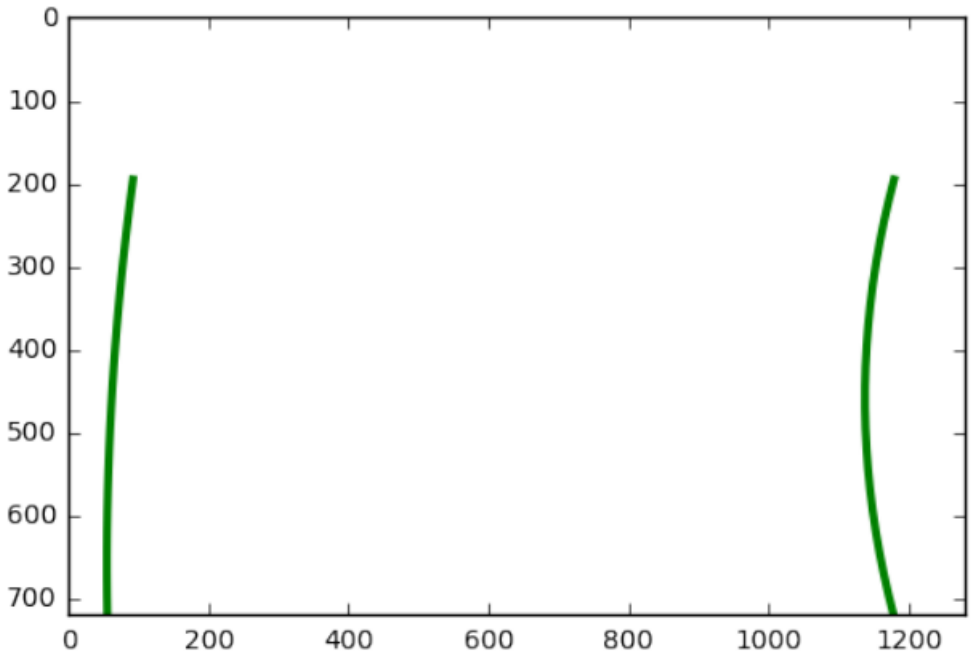
מסמן את פס הנתיב השמאלי:



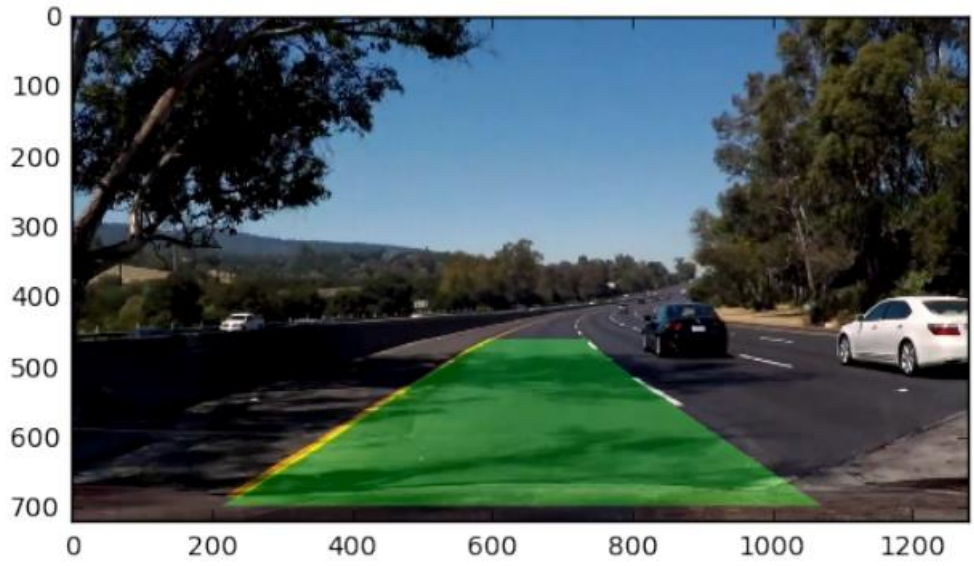
מסמן טרפז ברוחב נתיב מהשוליים שהתקבלו:

מאמנים אותו על זיהוי שוליים:

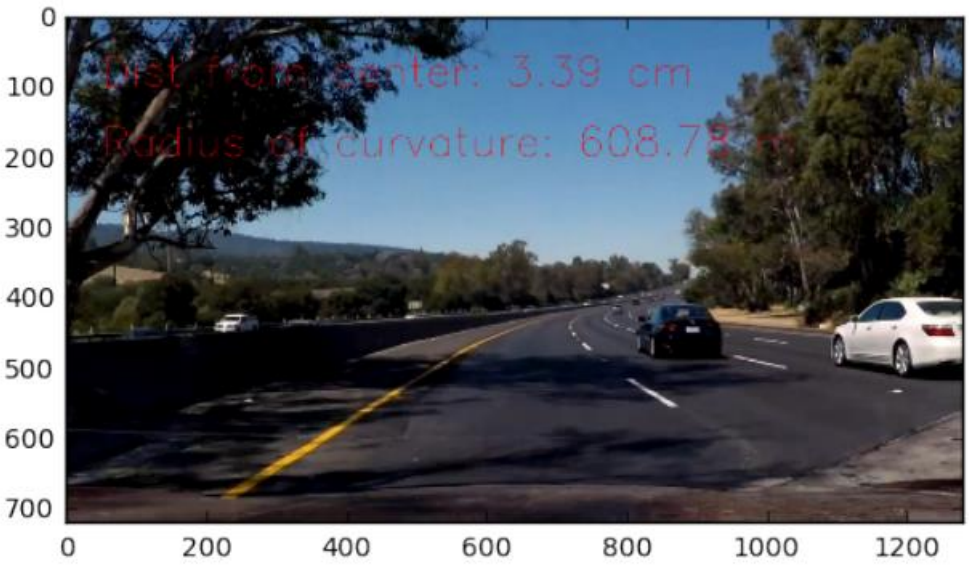
האלגוריתם מסמן את צורת הנתיב שזיהה:



התמונה שהתקבלה:



מרכז הנתיב ועיקול הנתיב:



חיבור בין C++ ל-python:

```
void DrivingScenarios::ConnectKalmanFilter()
{
    string result = "";
    try
    {
        // Calling the Kalman filter from a Python file with the parameters we recorded
        string command = "python.exe kalmanFilter.py " + to_string(dt) + " " + to_string(GetoldvelosityX()) + " " + to_string(GetoldvelosityY()) + " " +
            to_string(GetvelosityX()) + " " + to_string(GetvelosityY());
        pipe = _popen(command.c_str(), "r");
        while (!feof(pipe))
        {
            if (fgets(buffer, 128, pipe) != NULL)
                result += buffer;
        }


        stringstream ss(result);
        double pos;
        while (ss >> pos)
        {
            filtered_positions.push_back(pos);
        }


        // Access and manipulate the filtered positions as needed
        cout << "Filtered positions as double array:" << endl;
        for (double position : filtered_positions)
        {
            cout << position << " ";
        }
        SetoldvelosityX(filtered_positions[2]);
        SetoldvelosityY(filtered_positions[3]);
        _pclose(pipe);
    }
    catch (const exception& e)
    {
        cerr << "Exception caught: " << e.what() << endl;
    }
```

אלגוריתם הניווט וקבלת ההחלטות:

תכננתי את הקוד בצורה שפונקציה תהיה גנרית ככל האפשר.

המקרים שהתייחסתי אליהם:

- red light
- Red light straight
- Red light right
- Red light left

- green light
- Green light straight
- Green light right
- Green light left
- Stop sign
- Crosswalk
- Speed limit to 80
- Right
- Left
- Straight

כשלב ראשון חשבתי מה תהיה הדרך להפעלת הפונקציות והחלטתי שאני יעשה את זה בצורה של hash table מסוג מצביע לפונקציה כדי שהקוד יהיה יעיל ומהיר ככל האפשר.

מיינתי את המקרים ל-2 טבלאות גיבוב:

טבלה ראשונה בשם HashFunctionDrivingScenarios בגודל 6

וטבלה שניה בשם HashFunctionDirection בגודל 3

הנה טבלה המתארת כיצד מספר פונקציות יכולות להיות מופעלות מאותו מיקום בטבלת HashFunctionDrivingScenarios:

0		1		2		3		4		5		מיקום ב-Hash table
RedLightStraight		RedLightRight		RedLightLeft		GreenLight		Stop		SpeedLimitSignFor80		שם הפונקציה המופעלת
מצבים בהם תופעל הפונקציה וקוד הזיהוי החוזר מ- yolo:												
red light	0	Red light right	5	Red light left	4	green light	1	Stop sign	9	Speed limit to 80	10	
Red light straight	3					Green light straight	6	Crosswalk	11			
						Green light right	8					
						Green light left	7					

בגלל שפונקציות אלו מתבצעות תוך כדי נסיעה רק במקרים הנ"ל, יש את טבלת HashFunctionDirection שממנה מופעלות כל הזמן פונקציות Straight right או left

ואז כאשר רוצים להפעיל פונקציה מהטבלה הקודמת יש לבדוק שהטבלה לא מגבילה את זה. כמו לדוגמא לשם הפעלת פונקצית red right מטבלת HashFunctionDrivingScenarios צריך לבדוק שהפונקציה המופעלת כעת בטבלת HashFunctionDirection היא Straight .

להלן טבלה המתארת את המקרים בהם יוגבלו פונקציות מטבלת HashFunctionDrivingScenarios :

0		1		2		מיקום ב-Hash table
Left		Right		Straight		שם הפונקציה המופעלת
פונקציות מטבלת HashFunctionDrivingScenarios שיוגבלו:						
GreenLight	3	GreenLight	3	RedLightStraight	0	
במקרה של: Green light left		במקרה של: Green light Right		במקרים: red light • Red light straight •		
RedLightLeft	2	RedLightRight	1	GreenLight	3	
במקרה של: Red light left		במקרה של: Red light right		במקרים: green light • Green light straight •		

פונקציית הגיבוב לטבלת HashFunctionDirection:

כאשר מנתחים הוראת ניווט נקבל אחד מ-3 הכיוונים: ישר ימינה או שמאלה, נכניס את המילה שהתקבלה לפונקציות מאקרו הבאות:

```
#define CHECK_STRAIGHT(direction) ((direction!= "right" && direction!= "left")?2:0
#define CHECK_DIRECTION(direction) (direction == "right" ?1:0)
```

על כל מילת כיוון נזמן את שני הפונקציות הללו עם חיבור ביניהם ואז נקבל:

עבור left- יוחזר הערך 0 ותופעל הפונקציה **HashFunctionDirection[0]**

עבור right– יוחזר הערך 1 ותופעל הפונקציה **HashFunctionDirection[1]**

עבור כל כיון אחר ישר, כיכר, סיבוב, מיזוג כיוון שהתייחסתי רק ל-3 מצבים יוחזר הערך 2 ותופעל הפונקציה **HashFunctionDirection[2]**

פונקציית הגיבוב לטבלת HashFunctionDrivingScenarios **:**

כאשר מנתחים את מקרי הזיהוי מאלגוריתם yolo , זה מגיע כמספר כאשר כל מספר מייצג

עצם מסויים שזוהה.

להלן טבלת העצמים ומספר הזיהוי שלהם:

תיאור העצם שזוהה:	מספר הזיהוי:
Red light	0
Green light	1
Red light straight	2
Red light left	3
Red light right	4
Green light straight	5
Green light left	6
Green light right	7
Stop sign	8
Speed limit to 80	9
Crosswalk	10

מספר הזיהוי נשלח לכל פונקציות המאקרו הבאות וע"י חיבור התוצאה ביניהם נגיע למיקום בטבלת HashFunctionDrivingScenarios עם הפונקציה שיש להפעיל:

```
#define check_RedLightStraight(state)((state ==0||state==3)?0:1);
#define check_RedLightRight(state)(state ==5?0:1);
#define check_RedLightLeft(state)(state ==4?0:1);
#define check_GreenLight(state)((state ==1||state==6||state==7||state==8)?0:1);
#define check_Stop(state)((state ==9||state==11)?0:1);
#define check_SpeedLimitSignFor80(state)(state ==10?0:1);
```

הפונקציה שתופעל:	מספרי הזיהוי:
HashFunctionDrivingScenarios[0]	0,3
HashFunctionDrivingScenarios[1]	5
HashFunctionDrivingScenarios[2]	4
HashFunctionDrivingScenarios[3]	1,6,7,8
HashFunctionDrivingScenarios[4]	9,11
HashFunctionDrivingScenarios[5]	10

10.יעילות

- **דיוק**: יש אפשרות להוסיף לפקודת ההרצה conf 0.7-- ואז מתקבלים תוצאות עם אחוזי זיהוי גבוהים. המודל מאומן על 92% הצלחה.
- **מהירות**: מהירות האלגוריתם תלויה בצורת הקלט. אם זה מתקבל כל פריים בתמונה- האלגוריתם מזהה במהירות. לעומת זאת על סרטון האלגוריתם מזהה יותר לאט.
- **מורכבות**: לאמן את אלגוריתם yolo זה דווקא נחמד, מאוד פשוט וקל להבנה.
- **שימוש במשאבים**: האלגוריתם תלוי בספריות ובתוכנות שמותקנות על המחשב. מלבד זאת כאשר הוא מופעל בזמן אמת הוא צריך מקום בזיכרון של המחשב עבור צורת הזיכרון שנבחרה. ישנה אפשרות לבחור קובץ אקסל, קבצי טקסט- עבור כל פריים, סרטון וידאו עם תיבות תוחמות עבור האובייקטים שזוהו והתגיות שלהם.

11. אפיון המערכת

חומרה: מעבד i7, RAM 16GB

עמדת פיתוח: מחשב Asus

מערכת הפעלה: windows 11

שפות תוכנה: python, react ,C++

כלי תוכנה לפיתוח המערכת:visual code, visual studio 2022, anaconda navigator

מסד נתונים: אין

עמדת משתמש מינימלית:

- חומרה: מעבד i3 , 4GB ram
- מערכת הפעלה: windows 11 ומעלה
- חיבור לרשת: נדרש
- תוכנות:PyTorch, cuDNN
- chrom, CUDA
- מערכת לא מרובת מעבדים

11.1 מודול המערכת

נושאים באחריות המערכת:

- חישוב המסלול הקצר ביותר לנקודת היעד
- בדיקה כל פעם לאיזה כיוון צריך לנסוע: ישר, ימינה או שמאלה
- נסיעה בכביש בעל נתיב אחד בלבד
- זיהוי עצמים בזמן אמת
- תפיסת הסביבה
- קבלת החלטות בזמן אמת
- זיהוי מיקום
- חישוב מהירות בזמן אמת ע"י קבלת נתונים מחיישני IMU
- זיהוי נתיב הנסיעה והחזרת מרכז הנתיב

נושאים שאינם באחריות המערכת:

- זיהוי עצמים ש-yolo לא אומן עליהם
- נסיעה בכביש בעל 2 נתיבים
- התעסקות עם מצבי בלתמ"ים
- חיזוי עצמים ותפיסת הסביבה בזמני מזג אוויר קיצוניים

11.2 אפיון פונקציונלי

פונקציות עיקריות:

צד שרת:

()Receivinginformation- יוצרת שרת ב-C++ שמקבל מ-react וקטור עם הוראות ניווט

(SpeedCar(int maxSpeed -האצת מהירות הרכב עד למהירות שהתקבלה

(SlowdownCar(int MinSpeed – האטת מהירות הרכב עד למהירות שהתקבלה

(getLastCreatedFolder(const string& path- קבלת התקייה הנוכחית שאלגוריתם yolo מכניס לשם נתונים בנסיעה הנוכחית

()UpdateStateFromYolo- בודקת כל הזמן אם זוהו עצמים חדשים. אם כן מעדכנת את המערכת אלו עצמים זוהו

()ConnectKalmanFilter- חיבור מ-C++ ל-python. שליחת הנתונים הנדרשים למסנן קלמן וקבלת המיקום הנוכחי המשוערך החדש

()processFile- בודקת איזה פונקציה צריך להפעיל בהתאם לעצם שזוהה

(int placeinhash, double dist)PlayHashFunctionDirection – מפעילה בטבלת הגיבוב את הפונקציה המתאימה בהתאם לכיוון שהתקבל מ-Google maps

(int placeinhash)PlayHashFunctionDrivingScenarios- מפעילה בטבלת הגיבוב את הפונקציה המתאימה בהתאם לעצמים שזוהו

(DrivingScenarios& carpoint)calculateSpeed- עדכון מסימולציה של חיישן IMU על המהירות הנוכחית של הרכב

(DrivingScenarios& carpoint)UpdatePossion- עדכון מסימולציה של חיישן GPS על המיקום הנוכחי של הרכב

צד לקוח:

()InitMap- מזהה מיקום נוכחי של הרכב. מציגה על המסך את מפת הדרכים מנקודת המוצא אל היעד וסימון הנתיב הנבחר באדום

()InitAutocomplete- השלמה אוטומטית של הצעות למקומות יעד

()calculateRoute- חישוב המסלול הקצר ביותר ליעד

()ShowTurns- חישוב הוראות הנסיעה ושליחתם לאלגוריתם ב-C++

11.3 ביצועים עיקריים

- הכנסת נקודת היעד
- לחיצה על כפתור "SEND"
- זיהוי עצמים בזמן אמת
- זיהוי נתיב בזמן אמת
- קבלת החלטות בזמן אמת לגבי פעילות הרכב

11.4 אילוצים

מכשיר עם מצלמה עובדת. זיהוי מיקום אוטומטי של המחשב.

12. תיאור הארכיטקטורה

12.1 הארכיטקטורה של הפתרון המוצע

בצד שרת:

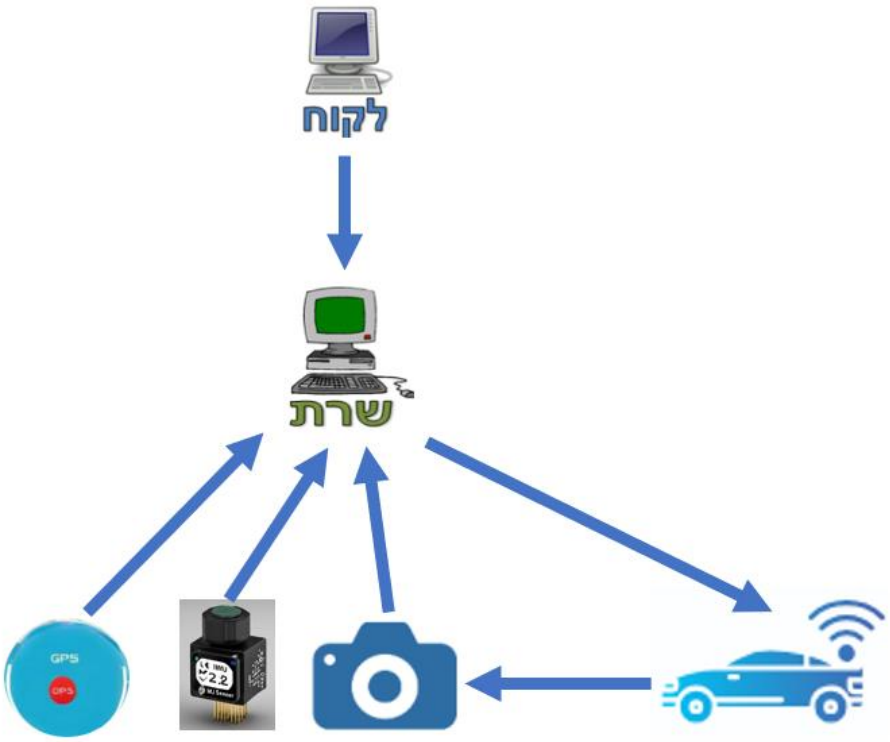
- קובץ KalmanFilter בסיומת py -מסנן קלמן שמבצע את השיערוך מיקום
- קובץ IMU בסיומת cpp – סימולציה של חיישן IMU שמחשב מהירות נוכחית של הרכב
- קובץ GPS בסיומת cpp- סימולציה של חיישן GPS שמחזיר מיקום נוכחי של הרכב בזמן אמת
- קובץ DrivingScenarios בסיומת cpp - מבצע את קבלת ההחלטות של המערכת. מכיל פונקציות להפעלה בזמן אמת וכן פונקציה שמתחברת ל-yolo ומעדכנת נתונים בזמן אמת.
- קובץ last.pt מריץ את אלגוריתם yolo בזמן אמת ושומר את תוצאות החיזוי
- קובץ P1 בסיומת ipynb מריץ את אלגוריתם זיהוי הנתיב בזמן אמת והחזרת מרכז הנתיב

בצד לקוח:

קבצים בסיומת js המייצגים קומפוננטות וקבצים בסיומת css לעיצוב.

- קומפוננטת Map לחישוב המסלול הקצר ביותר מנקודת המוצא לנקודת היעד ושליחת הוראות הניווט לשרת ב-C++

12.2 תיאור הרכיבים בפתרון



בארכיטקטורה זו הלקוח מכניס את היעד אליו הוא רוצה להגיע, הוראות הנסיעה נשלחות על השרת- האלגוריתם המרכזי, השרת מפענח את ההוראה הראשונה שולח למכונית מידע, המכונית מתחילה לנסוע, בו זמנית החיישנים מופעלים: מצלמה, חיישן GPS, חיישן IMU הם מעבירים נתונים לשרת, השרת מפענח אותם ונותן הוראות לרכב בהתאם. זה עובד בצורה אסינכרונית למשך כל זמן הנסיעה עד שהמכונית מגיעה ליעדה.

12.3 תיאור פרוטוקולי תקשורת

HTTPS- פרוטוקולי HTTP/HTTPS חיוניים לתקשורת עם שירותי Google דרך ממשקי API עבור משימות כמו ניווט במפה, קידוד גיאוגרפי, עדכוני תנועה בזמן אמת וכו.'

TCP/IP- פרוטוקול תקשורת בסיסי של האינטרנט והוא עשוי לשמש לתקשורת בין המערכות המשולבות של הרכב האוטונומי לשירותים חיצוניים כמו ממשקי ה API-של גוגל.

RTSP- ישמש להזרמת נתוני וידאו בזמן אמת מהמצלמה ליחידות העיבוד המשולבות או למטרות ניטור מרחוק.

NMEA- משמש לתקשורת בין מכשירי GPS ומערכות אחרות להחלפת נתונים כמו מיקום, מהירות, זמן וכו'.

12.4 שרת- לקוח

צד שרת נכתב בשפות: C++ בשילוב python

צד לקוח: javascript, css, html בטכנולוגית react

שליחת הנתונים מלקוח לשרת בפורמט json

תקשורת לקוח משתמשת בטכנולוגית API

13. ניתוח ותרשים Use cases /UML של המערכת המוצעת

הפרוייקט שלי בעקרון הוא שרת. צד הלקוח מיועד רק כדי לקבל את היעד.

13.1 Use cases

משתמש:

- **תכנון נסיעה:** המשתמש יכול להכניס את היעד אליו הוא רוצה להגיע
- **נסיעה אוטונומית:** הרכב של המשתמש ייסע אל היעד אליו הוא מעוניין להגיע

13.2 תיאור Use cases

UC1

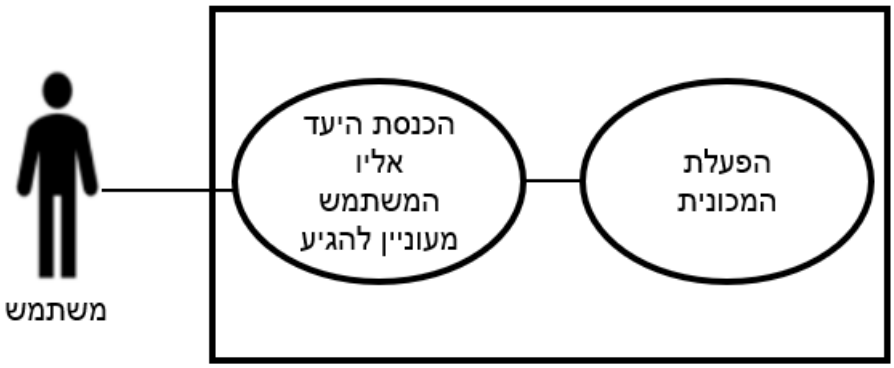
- **Identifier**: UC1
- **Name**: בחירת היעד
- **Description**: המשתמש יכניס את היעד אליו הוא מעוניין לנסוע, ניתנת לו עזרה של הצעות השלמה אוטומטית של חיפוש מקומות
- **Actors**: משתמש
- **Frequency**: פעם אחת בנסיעה
- **Condition-post**: היעד קיים במערכת

UC2

- **Identifier**: UC2
- **Name**: הפעלת המכונית האוטונומית
- **Description**: האלגוריתם יפעיל את הרכב לכיוון היעד שהתקבל. הרכב ייסע עד שיגיע אל היעד
- **Actors**: משתמש
- **Frequency**: במשך כל זמן הנסיעה

Condition-post: חיישנים עובדים כראוי

13.3 Use Case Diagram



13.4מבני נתונים בפרויקט

לאורך הפרוייקט השתמשתי במבני נתונים הבאים:

- **מערכים רב ממדיים-** מטריצות. באו לידי שימוש בפונקציית מסנן קלמן. כחלק מתהליך החישוב שם יש הכפלת מטריצות בווקטור ולכן נדרשתי למבנה נתונים זה.

```
np.array([[1,        0,        dt,       0],
          [0,        1,        0,        dt],
          [0,        0,        1,        0],
          [0, 0, 0, 1]])
```

- **וקטור-** חלק מחישובים במסנן קלמן

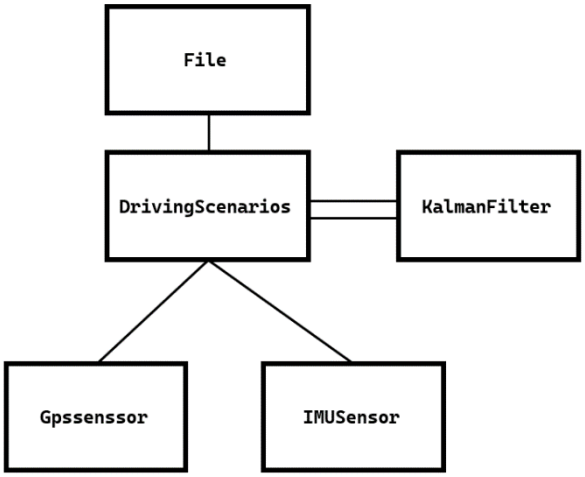
```
initial_state = np.array([0, 0, 0, 0])
```

- **Hash table-** כל אלגוריתם הניווט מבוסס על שימוש בטבלת גיבוב מסוג מצביע לפונקציה.זה חסך לי המון כפילויות של קוד וכן שכל האלגוריתם המרכזי של התוכנית מתבצע ביעילות ובמהירות בסיבוכיות O(1) .

```
public:
    void (DrivingScenarios::*HashFunctionDrivingScenarios[6])() = {&DrivingScenarios::RedLightStraight,&DrivingScenarios::RedLightRight,
                                                                   &DrivingScenarios::RedLightLeft,&DrivingScenarios::GreenLight,
                                                                   &DrivingScenarios::Stop,&DrivingScenarios::SpeedLimitSignFor80};
    void (DrivingScenarios::*HashFunctionDirection[3])(double) = { &DrivingScenarios::Left,&DrivingScenarios::Right,
                                                                   &DrivingScenarios::Straight };
```

13.5 מחלקות

תרשים המחלקות:



הפרוייקט ב-C++ מורכב ממחלקות שונות שיפורטו להלן:

מחלקת IMUSensor:

```
#pragma once
#include "DrivingScenarios.h"
#include <thread>
using namespace std;

class IMUSensor
{
public:
    IMUSensor();
    void calculateSpeed(DrivingScenarios& carpoint);
    void startIMUSensor(DrivingScenarios& carpoint);
    void stopIMUSensor();
    double getCurrentSpeed();
    int GettimeSensor();
    void SettimeSensor(int second);
private:
    thread imuThread;
    mutex mtxtimeSensor;
    bool isRunning;
    int timeSensor;
    double speedX, speedY, time, speed, distanceInOneSecond, distance, Dataprocessing, currentSpeed;
};
```

תפקיד:

מבצעת סימולציה של חיישן IMU המודד את כוחות התאוצה לאורך 2 הצירים. למעשה הוא מחזיר בכל רגע נתון את המהירות של הרכב בציר-X ובציר-Y וע"י חישוב `sqrt((speedX * speedX) +( speedY * speedY))` מעדכן את המהירות הנוכחית של הרכב , התאוצה, והמרחק שעבר עד כה.

מחלקת Gpssenssor:

```
#pragma once

#include "DrivingScenarios.h"


class Gpssenssor
{
public:
    Gpssenssor();
    void UpdatePossion(DrivingScenarios& carpoint);
};

```

תפקיד:

מבצעת סימולציה של חיישן GPS המחזיר מיקום נוכחי של הרכב בכל רגע בציר-X ובציר- Y, המיקום נשלח למסנן קלמן המחזיר את המיקום החדש המשוערך של הרכב.

מחלקת File:

```
#pragma once
#include <iostream>
#include <string.h>
using namespace std;
class File
{
public:
    File();
    string GetWordAfterLastDash(const std::string& sentence);
    double ExtractLastWordToDouble(const string& sentence);

};

```

תפקיד:

מממשת פונקציות לעבודה וניתוח קבצים. כיוון שסימולציות החיישנים ממושמשות ע"י קריאה מקבצים מחלקה זו הכרחית.

מחלקת DrivingScenarios:

```
#pragma once
#include <string>
#include <mutex>
#include <vector>
#include <array>
#include <iostream>
#include <stdbool.h>
#include "File.h"

using namespace std;
class DrivingScenarios
{
private:
    mutex mtxCurrentSpeed;
    string line;
    mutex mtxAccelerationSpeed;
    mutex mtxTimeCar;
    mutex mtxdistance;

    mutex mtxPathOfSpeed;
    mutex mtxstate;
    mutex mtxTrafficLightColor;
    mutex mtxvelosityX;
    mutex mtxvelosityY;
    mutex mtxoldvelosityX;
    mutex mtxoldvelosityY;
    mutex mtxplay;
    double accelerationSpeed;
    double currentSpeed;
    bool degel;
    string str;
    string directiontotravel;
    string signal;
    string PathOfSpeed;
    string direction;
    int temp;
```

```
    double distance;
    int timeCar;
    bool onyolo;
    string state;
    string TrafficLightColor;
    char buffer[128];
    FILE* pipe;
    vector<double> filtered_positions;
    double velosityX ;
    double velosityY;
    double oldvelosityX ;
    double oldvelosityY;
    double dt;
    double distancetoturn;
    array<bool,8> arrState;
    bool play;
    int maxspeed;

public:
    void (DrivingScenarios::*HashFunctionDrivingScenarios[6])() = {&DrivingScenarios::RedLightStraight,&DrivingScenarios::RedLightRight,
                                                                   &DrivingScenarios::RedLightLeft,&DrivingScenarios::GreenLight,
                                                                   &DrivingScenarios::Stop,&DrivingScenarios::SpeedLimitSignFor80};
    void (DrivingScenarios::*HashFunctionDirection[3])(double) = { &DrivingScenarios::Left,&DrivingScenarios::Right,
                                                                   &DrivingScenarios::Straight };

    void PlayHashFunctionDirection( int placeinhash, double dist);
    void PlayHashFunctionDrivingScenarios(int placeinhash);
    DrivingScenarios();
    double GetvelosityX();
    void SetvelosityX(double newvelosityX);
    double GetvelosityY();
    void SetvelosityY(double newvelosityY);
    double GetoldvelosityX();
    void SetoldvelosityX(double newoldvelosityX);
    double GetoldvelosityY();
    void SetoldvelosityY(double newoldvelosityY);
```

```
    bool Getplay();
    void Setplay(bool newplay);
    string Getdirection();
    void Setdirection(string newdirection);
    string GetTrafficLightColor();
    void SetTrafficLightColor(string newTrafficLightColor);
    double Getdistance();
    void Setdistance(double newdistance);
    //string Getstate();
    double Getdistancetoturn();
    void SetaccelerationSpeed(double newaccelationspeed);
    double GetaccelerationSpeed();
    double GetcurrentSpeed();
    void SetcurrentSpeed(double speed);
    int GettimeCar();
    void SettimeCar(int second);
    string GetPathOfSpeed();
    int MaxSpeed();
    void Setmaxspeed(int newmaxspeed);
   // void Setstate(string s);
    void RedLightStraight();
    void RedLightRight();
    void RedLightLeft();
    void GreenLight();
    void SpeedLimitSignFor80();
    void Stop();
    void SignalLight(std::string direction);
    void SlowdownCar(int minSpeed=0);
    void SpeedCar(int maxSpeed=120);
   // double DistanceFromCarToObject(std::string filename);
    string ReadFromFile(std::string filepath);
    void Right(double distancetoturnRight);
    void Left(double distancetoturnLeft);
    void Setdistancetoturn(double newdistancetoturn);
    void Straight(double distancetoturnStraight=0);
    string getLastCreatedFolder(const string& path);
    void UpdateStateFromYolo();
    string extractFirstWord(const string& input);
    void ConnectKalmanFilter();
    void processFile(const std::string& filePath);
    void Offyolo();
};
```

תפקיד:

מחלקה זו היא בעצם המחלקה החשובה ביותר בפרוייקט. היא מכילה את תכונות הרכב כגון: מהירות, תאוצה, מיקום בציר-X, מיקום בציר-Y, וכן בגלל שהפונקציות מופעלות בצורה אסינכרונית ויש גישה למשאבים משותפים כי כל החיישנים משנים את תכונות הרכב היה צורך להשתמש ב- mutex - אמצעי סנכרון המשמש לשליטה בגישה למשאבים משותפים בתכנות במקביל. המטרה העיקרית של mutex היא למנוע משרשורים מרובים לגשת לנתונים משותפים בו זמנית, מה שעלול להוביל לנתונים שגויים. במחלקה זו ממומשות כל הפונקציות שמפעילות את הרכב: האצה, האטה, הגבל מהירות, עצירה, ימינה, שמאלה, ישר ועוד. בנוסף חיבורים לפיתון ועדכון נתונים מאלגוריתם yolo בזמן אמת מתבצעים במחלקה זו.

מחלקת Server:

```cpp
#pragma once
#include <vector>
#include <string>
class Server
{
public:
    Server() {};
    std::vector<std::pair<std::string, std::string>> Receivinginformation();
    void handle_get(int client_socket, char* request);
    void handle_post(int client_socket, char* request);
};
```

תפקיד:

יוצרת שרת ב-C++ שמחכה לקבל בקשות. כאשר בצד לקוח הלקוח בוחר מסלול, נשלחת הבקשה עם הוראות הניווט לשרת. השרת מעבד את הנתונים ומנווט את המכונית בהתאם.

מחלקת מסנן קלמן ב-python:

```python
import numpy as np
import sys

class KalmanFilter:
    def __init__(self, dt, process_noise, measurement_noise):
        self.dt = dt
        self.F = np.array([[1, 0, dt, 0],
                           [0, 1, 0, dt],
                           [0, 0, 1, 0],
                           [0, 0, 0, 1]])
        self.H = np.array([[1, 0, 0, 0],
                           [0, 1, 0, 0]])
        self.Q = np.eye(4) * process_noise
        self.R = np.eye(2) * measurement_noise
        self.P = np.eye(4)

    def predict(self):
        self.x = np.dot(self.F, self.x)
        self.P = np.dot(self.F, np.dot(self.P, self.F.T)) + self.Q

    def update(self, z):
        y = z - np.dot(self.H, self.x)
        S = np.dot(self.H, np.dot(self.P, self.H.T)) + self.R
        K = np.dot(self.P, np.dot(self.H.T, np.linalg.inv(S)))
        self.x = self.x + np.dot(K, y)
        I = np.eye(4)
        self.P = np.dot(I - np.dot(K, self.H), self.P)
```

```python
# Example of using the Kalman filter
input1 = float(sys.argv[1])
input2 = float(sys.argv[2])
input3 = float(sys.argv[3])
input4 = float(sys.argv[4])
input5 = float(sys.argv[5])
initial_state = np.array([0, 0, 0, 0])  # Initial state: x, y, vx, vy
process_noise = 0.1
measurement_noise = 0.1
dt = input1
kf = KalmanFilter(dt, process_noise, measurement_noise)
kf.x = initial_state

# Measurements of x, y position
measurements = np.array([[input2,input3],[input4,input5]])

# Run the Kalman filter
filtered_positions = []
for measurement in measurements:
    kf.predict()
    kf.update(measurement)
    filtered_positions.append(kf.x[:2])
for p in filtered_positions:
    print(" ".join(str(os) for os in p))
```

תפקיד:

מחלקה אחראית לשערך את המיקום הנוכחי של הרכב על סמך נתוני מיקום שהתקבלו מחיישן GPS ונתוני מהירות שהתקבלו מחיישן IMU. הנתונים החדשים נשלחים חזרה לפונקציה ב-++C ומתעדכנים על המכונית.

14.תיאור התוכנה

סביבת עבודה: vs code ,visual studio, anaconda

שפות תכנות: צד שרת- C++,python

צד לקוח- html, JavaScript, API בטכנולוגיית react

15. אלגוריתם מרכזי

אלגוריתם הפעלת המכונית:

```
#include "File.h"
#include "DrivingScenarios.h"
#include "IMUSensor.h"
#include "Gpssenssor.h"
#include <iostream>
#include <list>
#include <vector>
#include <string.h>
#include <functional>
#include <thread> // std::thread, std::this_thread::sleep_for
#include <chrono>
#include <string>
#include <fstream>
#include <sstream>
using namespace std;
```

```
#define CHECK_STRAIGHT(direction) ((direction!= "right" && direction!= "left")?2:0 )
#define CHECK_DIRECTION(direction) (direction == "right" ?1:0)
int main()
{

    //build object

    IMUSensor imuSensor;
    DrivingScenarios car;
    Gpssenssor gpsSenssor;
    File files;
    thread timerThread(timerFunction, ref(car), ref(imuSensor));
    imuSensor.startIMUSensor(ref(car));
    thread yoloThread(&DrivingScenarios::UpdateStateFromYolo, ref(car));
    thread Gpsthread(&Gpssenssor::UpdatePossion, ref(car),ref(imuSensor));
    //start
    //read Filestring direction;
    string direction;
    int placeinarr;
    double distance;
    string filename = "src/Instructions.txt";
    ifstream file(filename);
```

```
    if (!file.is_open())
    {
        cout << "לא ניתן לפתוח את הקובץ." << endl;
        return 1;
    }
    string line;
    while (getline(file, line))
    {
        car.Setmaxspeed(100);
        direction = files.GetWordAfterLastDash(line);
        distance = files.ExtractLastWordToDouble(line);
        placeinarr = CHECK_DIRECTION(direction) + CHECK_STRAIGHT(direction);
        car.PlayHashFunctionDirection(placeinarr,distance);
        cout << "distance: " << distance << endl;
        this_thread::sleep_for(chrono::seconds(1));
    }
    file.close();

    //join object
    car.Offyolo();
    yoloThread.join();
    imuSensor.stopIMUSensor();
    Gpsthread.join();
    timerThread.join();
}

```

תיאור:

האלגוריתם מפעיל טיימר לחישוב זמן הנסיעה, מפעיל את חיישני IMU,GPS , מפעיל את פונקצית עדכון נתונים מ-yolo, מקבל הוראת ניווט ומדליק את הפונקציה המתאימה.

16. קוד התוכנית

```
void IMUSensor::startIMUSensor(DrivingScenarios& carpoint)
{
    isRunning = true;
    imuThread = thread(&IMUSensor::calculateSpeed, carpoint);
}
```

```

void IMUSensor::calculateSpeed(DrivingScenarios& carpoint)
{
    double prevSpeed = 0.0,acceleration; // Initialize previous speed variable
    string line;
    while (isRunning)
    {
        ifstream inputFile("src/IMUsensor.txt"); // Open the text file for reading
        if (!inputFile.is_open())
        {
            cerr << "Error opening file." << endl;
        }
        getline(inputFile, line);

        istringstream iss(line);
        iss >> speedX >> speedY; // Extract speed X, speed Y, and time from the line
        speed = sqrt((speedX * speedX) +( speedY * speedY)); // Calculate the total speed

        acceleration = (speed - prevSpeed); // Calculate acceleration
        carpoint.SetaccelerationSpeed(acceleration);
        carpoint.SetcurrentSpeed(speed);
        distance += (speed * 1000 / 3600 * GettimeSensor()); // Calculate the total distance covered directly
        carpoint.Setdistance(distance);
        cout << "Distance covered in current iteration: " << distance << " meters" << endl;
        prevSpeed = speed; // Update previous speed to current speed
        inputFile.close(); // Close the file
        this_thread::sleep_for(chrono::seconds(1)); // Wait for 1 second between iterations
    }
    //    cout << "Total distance covered: " << distance << " meters" << endl;
}
```

```
void IMUSensor::stopIMUSensor()
{
    isRunning = false;
    imuThread.join();
}
```

תיאור:

האלגוריתם המרכזי מפעיל את חיישן IMU. החיישן מופעל בצורת thread כי הוא אמור לפעול יחד עם חיישנים נוספים במקביל. קריאה לפונקציה calculateSpeed, הפונקציה קוראת בכל שניה שורה מקובץ הסימולציה השורה מכילה 2 ערכים מופרדים ברווח ביניהם, הערך הראשון הוא מהירות בציר X והערך השני מהירות בציר Y.

מחשבים את המהירות הכוללת ע"י נוסחה:
מהירות כוללת=מהירות בציר X בריבוע+ מהירות בציר Y בריבוע

מחשבים תאוצה ע"י נוסחה:

תאוצה=מהירות קודמת - מהירות נוכחית

מחשבים את המרחק שעבר משניה קודמת ע"י הנוסחה:

מרחק משניה קודמת=זמן הנסיעה שעבר עד כה* 1000/3600 *מהירות נוכחית

מעדכנים את משתני המכונית בערכים העדכניים.

```
void Gpssenssor::UpdatePossion(DrivingScenarios& carpoint)
{
    std::ifstream file("src/GPS.txt");
   // std::vector<std::pair<double, double>> values;
    double pos1, pos2;
    try
    {
        std::string line;
        while (std::getline(file, line))
        {
            std::istringstream iss(line);
            iss >>pos1>>pos2;
            carpoint.SetvelosityX(pos1);
            carpoint.SetvelosityY(pos2);
            carpoint.ConnectKalmanFilter();
            this_thread::sleep_for(chrono::seconds(2));
        }
        file.close();
    }
    catch (const char* error)
    {
        std::cout << "Error: " << error << std::endl;
    }

}
```

תיאור:

הפונקציה מופעלת כ- thread בגלל המקביליות של הפונקציות. קוראת בכל פעם שורה מקובץ הסימולציה של חיישן GPS , השורה מכילה 2 ערכים מופרדים ברווח, הערך הראשון הוא מיקום בציר X והערך השני הוא מיקום בציר Y. מעדכנת את המיקומים של המכונית ומפעילה את מסנן קלמן. היא עובדת כל עוד יש ערכים לקריאה- ז"א שעדיין לא הגענו אל היעד.

```
void DrivingScenarios::UpdateStateFromYolo()
{
    string folderPath = "C:/Users/USER/Documents/project/final_modal/yolov5/runs/detect";
    string lastCreatedFolder = getLastCreatedFolder(folderPath)+"/labels";
    onyolo = true;
    while (onyolo)
    {
        for (const auto& entry : filesystem::directory_iterator(folderPath))
        {
            if (entry.path().extension() == ".txt")
            {
                processFile(entry.path().string());
            }
            //std::this_thread::sleep_for(std::chrono::seconds(2));
        }

        // Wait for 1 second before checking again
        this_thread::sleep_for(chrono::seconds(1));
    }
}
```

תיאור:

אלגוריתם yolo מייצר עבור כל הרצה תיקייה בשם exp_ ושומר בה את קבצי הזיהוי. בשלב ראשון פונקציה זו מחפשת את התקיה האחרונה שנוצרה שבה נוכל לקבל את הנתונים של ההרצה הנוכחית. מוסיפה את זה לנתיב שצוין ואז מוסיפה לנתיב /lables כי אני בחרתי לעבוד עם קבצי הטקסט והם ממוקמים שם. משתנה onyolo נועד כדי שנוכל להפסיק בסוף את קריאת הקבצים. ברגע שהפונקציה מזהה שקיימים בתקיה קבצי טקסט הוא שולחת אותם לפונקציה processfile שמטפלת בתוכן הקבצים ומוחקת אותם לאחר הטיפול בהם.

```
void DrivingScenarios::processFile(const string& filePath)
{
    ifstream inputFile(filePath);
    int state, placeinHashFunctionDrivingScenarios;
    if (!inputFile)
    {
        std::cerr << "Error opening the file: " << filePath << "\n";
        return;
    }

    string word;
    while (inputFile >> word)
    {
        state= stoi(word);
        placeinHashFunctionDrivingScenarios = check_RedLightStraight(state) + check_RedLightRight(state)
            + check_RedLightLeft(state) + check_GreenLight(state) + check_Stop(state) + check_SpeedLimitSignFor80(state);
        if (arrState[state] != true)
        {
            arrState[state] = true;
            thread t(&DrivingScenarios::PlayHashFunctionDrivingScenarios, this, state);
            t.detach();

        }
         // Ignore the rest of the line
         inputFile.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
    }

    inputFile.close(); // Close the file after processing

    // Close the file and then delete it
    std::remove(filePath.c_str());
    std::cout << "File " << filePath << " processed and deleted.\n";
}
```

תיאור:

הפוננקציה פותחת קובץ. כל שורה בקובץ מכילה מידע על העצם שזוהה: קוד הזיהוי וגבולות העצם בתמונה, מחלצים את קוד העצם מעבירים לפונקציית ה- hash לבדיקת המיקום של הפונקציה שצריך להפעיל במערך הפונקציות. ערך המיקום חוזר לתוך המשתנה state. קיים במחלקה מערך בוליאני בשם arrstate שמאותחל בערכי false. ברגע שזוהה עצם המערך arrstate[state] משתנה ל-true, וזאת כיוון ש– yolo מייצר קובץ טקסט בעבור כל פריים כל למשל שאם בפריים אחד זוהה רמזור אדום הוא בד"כ יזוהה גם בפריים 2, והפונקציות בנויות

בצורה שבמקרה של זיהוי הם עובדות בלולאה עד לסיום, אז אין צורך לזמן את אותה פונקציה המון פעמים, זה ייתן תוצאות שגויות. לכן כאשר פונקציה באמצע להתבצע תחסם האפשרות לזמן אותה עד שהיא גומרת. כשהיא תגמור הערך ישתנה ל-false ויהיה אפשר לזמן אותה שוב.

השרת ב-C++ שמקבל את המסלול מ-react:

```
#include <iostream>
#include <string>
#include <WS2tcpip.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <string>
#include <map>
#include <sstream>
#include <iomanip>
#include <regex>
#include <string>
#include <vector>
#pragma comment(lib, "ws2_32.lib")
using namespace std;
// Declaration of a function to handle GET requests
void handle_get(int client_socket, char* request);

// Declaration of a function to handle POST requests
void handle_post(int client_socket, char* request);


int main()
{
    WSADATA wsData;
    WORD ver = MAKEWORD(2, 2);

    int wsOK = WSAStartup(ver, &wsData);
    if (wsOK != 0) {
        std::cerr << "Can't initialize Winsock! Quitting" << std::endl;
        return -1;
```

```
    }

    SOCKET server_socket, client_socket;
    struct sockaddr_in server_address, client_address;
    socklen_t client_address_len;

    // Create socket
    server_socket = socket(AF_INET, SOCK_STREAM, 0);


    if (server_socket == -1) {
        cerr<<"Unable to create socket"<<endl;
        exit(EXIT_FAILURE);
    }

    // Bind socket to port 5000
    memset(&server_address, 0, sizeof(struct sockaddr_in));
    server_address.sin_family = AF_INET;
    server_address.sin_addr.s_addr = INADDR_ANY;
    server_address.sin_port = htons(54000);
    if (bind(server_socket, (struct sockaddr*)&server_address, sizeof(server_address)) == -1)
    {
        cerr<<"Unable to bind socket to port"<<endl;
        exit(EXIT_FAILURE);
    }

    // Listen for incoming connections
    if (listen(server_socket, 10) == -1)
    {
        cerr<<"Unable to listen for incoming connections"<<endl;
```

```
        exit(EXIT_FAILURE);
    }
    printf("Server is listening on port 54000...\n");

    while (1) {
        // Accept incoming connection
        client_address_len = sizeof(client_address);
        client_socket = accept(server_socket, (struct sockaddr*)&client_address, &client_address_len);
        if (client_socket == -1) {
            cerr<<"Failed to accept incoming connection"<<endl;
            continue;
        }

        cout<<"Accepted new connection from client:%d\n", ntohs(client_address.sin_port)<<endl;

        // Read HTTP request from client
        char request[1024];
        int length = recv(client_socket, request, sizeof(request),0);
        std::regex regex("maneuver=([^&]+)&distance=([^&]+)");
        std::smatch match;
        std::vector<std::pair<std::string, std::string>> result;

        std::string message = request;
        while (std::regex_search(message, match, regex)) {
            result.push_back(std::make_pair(match[1].str(), match[2].str()));
            message = match.suffix();
        }

        for (const auto& pair : result) {
            std::cout << "[" << pair.first << "," << pair.second << "]" << std::endl;
```

```
        }
        cout<<"the length:" <<length<<endl;
        if (length < 0)
        {
            cout<<"Failed to read from client"<<endl;
            closesocket(client_socket);
            continue;
        }


        cout<< message<<endl;
        // Extract the path from the message
        std::string path = message.substr(message.find_first_of('/') + 1, message.find('?') - message.find_first_of('/') - 1);

        // Extract the query string from the message
        std::string queryString = message.substr(message.find('?') + 1);

        std::map<int, std::string> objects;

        // Parse the query string to extract objects
        std::stringstream ss(queryString);
        std::string token;
        while (std::getline(ss, token, '&')) {
            std::string::size_type pos = token.find('=');
            int key = std::stoi(token.substr(0, pos));
            std::string value = token.substr(pos + 1);
            objects[key] = value;
        }
```

```
            std::string value = token.substr(pos + 1);
            objects[key] = value;
        }

        // Extract the title of the message
        std::string title = path;

        // Output the title and objects
        std::cout << "Title of the message: " << title << std::endl;  ≤ 23ms elapsed
        std::cout << "Objects:" << std::endl;
        // Determine the HTTP request method
        char http_method[10];
        int i = 0;
        while (request[i] != ' ') {
            http_method[i] = request[i];
            i++;
        }
        http_method[i] = '\0';

        // Determine the URI from the HTTP request
        char uri[1024];
        int j = 0;
        i++;
        while (request[i] != ' ') {
            uri[j] = request[i];
            i++;
            j++;
        }
        uri[j] = '\0';
```

```cpp
        cout<<"Request received: method="<< http_method<<"uri="<< uri<<endl;

        // Route the HTTP request
        if (strcmp(http_method, "GET") == 0) {
            handle_get(client_socket, uri);
        }
        else if (strcmp(http_method, "POST") == 0) {
            handle_post(client_socket, uri);
        }
        else {
            char error_message[] = "HTTP/1.1 400 Bad Request\r\nContent-Type: text/plain\r\nContent-Length: 0\r\n\r\n";
            send(client_socket, error_message, strlen(error_message),0);
        }

        // Close client socket
        closesocket(client_socket);
        printf("Connection closed\n");
    }

    // Close server socket
    closesocket(server_socket);

    return 0;



    /* ... */
}

void handle_get(int client_socket, char* request) {
    if (strcmp(request, "/") == 0) {
```

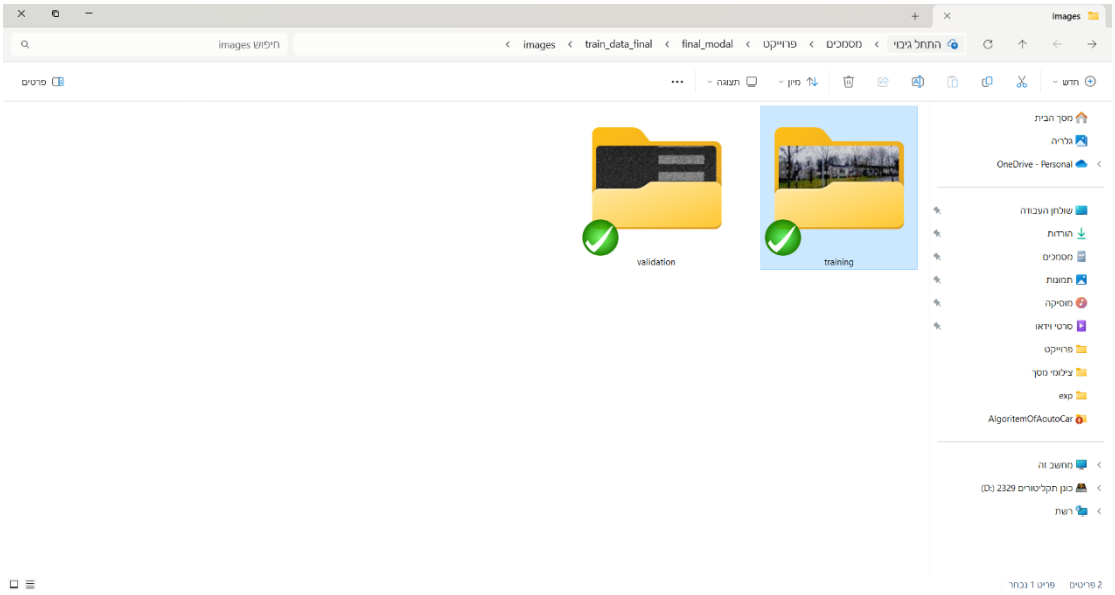
```cpp
void handle_get(int client_socket, char* request) {
    if (strcmp(request, "/") == 0) {
        char message[] = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nContent-Length: 77\r\n\r\n<html><body><h1>Welcome to my website</h1></body></html>";
        send(client_socket, message, strlen(message),0);
    }
    else if (strcmp(request, "/about") == 0) {
        char message[] = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nContent-Length: 72\r\n\r\n<html><body><h1>About me</h1><p>I'm a programmer</p></body></html
        send(client_socket, message, strlen(message),0);
    }
    else {
        char error_message[] = "HTTP/1.1 404 Not Found\r\nContent-Type: text/plain\r\nContent-Length: 0\r\n\r\n";
        send(client_socket, error_message, strlen(error_message),0);
    }
}

void handle_post(int client_socket, char* request) {
    char success_message[] = "HTTP/1.1 200 OK\r\nContent-Type: text/plain\r\nContent-Length: 15\r\n\r\nPost received\n";
    send(client_socket, success_message, strlen(success_message),0);
}
```

תיאור:

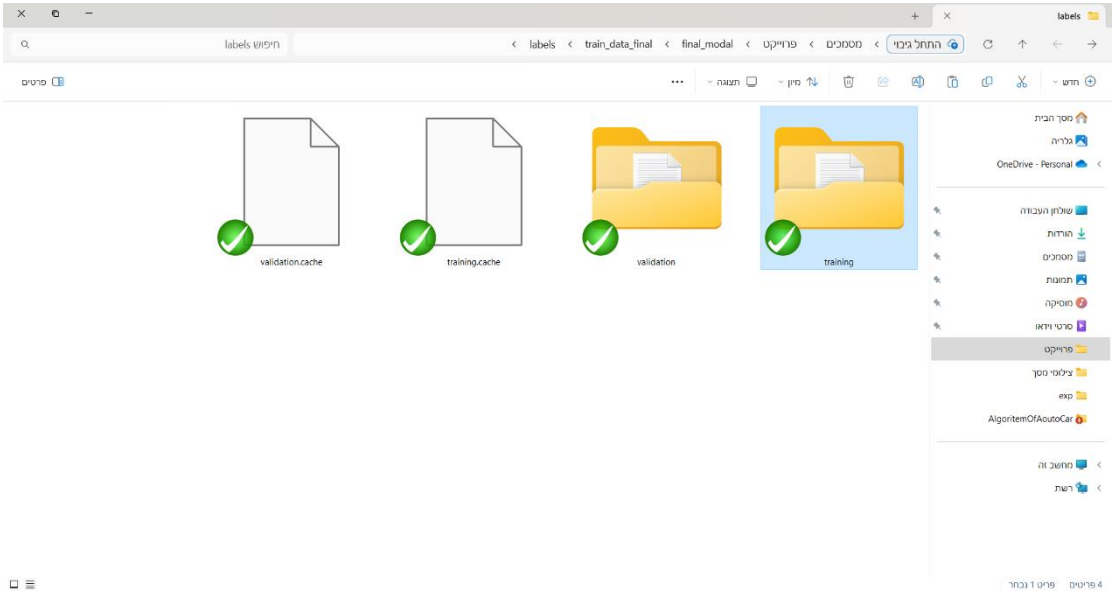
יוצרים שרת ב-C++. השרת ממתין שלקוח כלשהו יתחבר אליו. ברגע שהמשתמש לוחץ על היעד שאליו הוא רוצה להגיע נשלחת בקשה לשרת עם המסלול הקצר ביותר. השרת מקבל את הבקשה מנתח אותה, מחלץ נתונים ומעביר אותם לפונקציית ההפעלה המרכזית.

17. תיאור ה-dataset

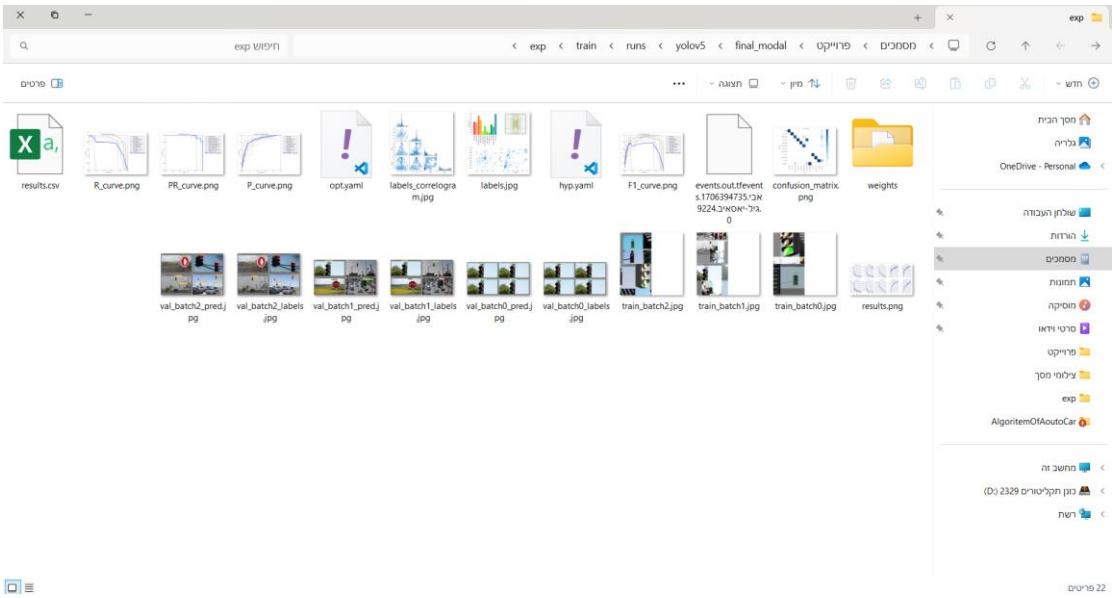
תיקיית האימון train של המודל, מכילה את מערך הנתונים שעליו אומן אלגוריתם yolo



תיקיית התגיות עבור כל תמונה מתיקיית train

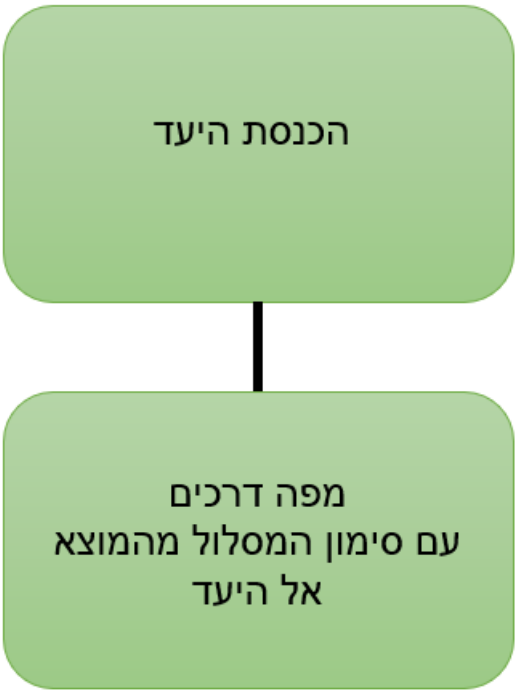


תוצאות האימון:



18. מסכים

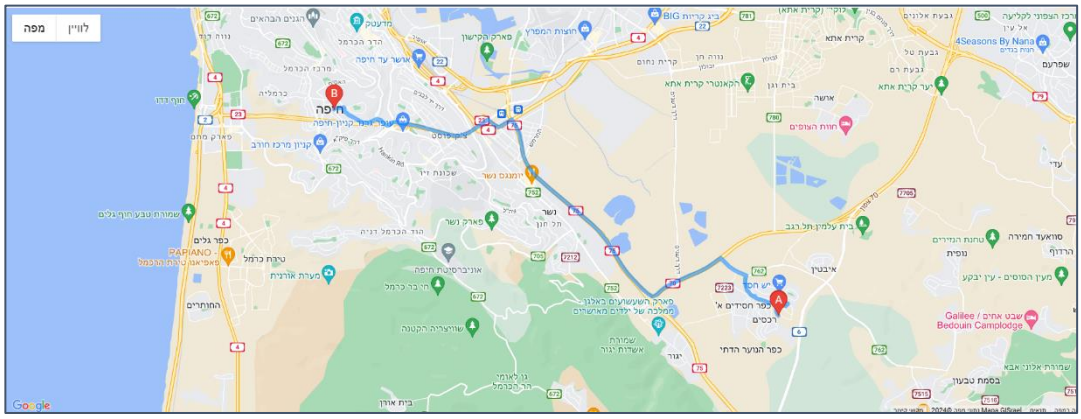
18.1 תרשים מסכים



18.2 צילום מסכים

הכנסת היעד וקבל עשרה מחיפוש אוטומטי.

סימון המסלול הקצר אל היעד לאחר לחיצה על הכפתור calculate route.



19. מדריך למשתמש

- הקלדה בתיבת החיפוש search place את שם היעד אליו מעוניינים להגיע
- בחירה מתוך הצעת אפשרויות להשלמה אוטומטית את המיקום
- לחיצה על כפתור calculate route לחישוב המסלול הקצר ביותר

20. פיתוחים עתידיים

- הוספת חיישן Lidar
- הוספת חיישן Radar
- ביצוע החלפת נתיב
- קישור פרוייקט זיהוי הנתיב לפרוייקט הנוכחי
- זיהוי עצמים נוספים
- התייחסות למצבי בלתמ"ים
- סימולציה ויזואלית
- עדכון תצוגה בזמן אמת

21. אבטחת מידע

הפרוייקט שלי לא מתעסק כ"כ עם צד לקוח. הדבר היחיד הוא שהוא מקבל מהלקוח זה את שם היעד. לאחר שהלקוח לוחץ על כפתור calculate route מתבצעת הפונקציה הזו:

```
ction calculateRoute() {
 var destination = document.getElementById('destination').value;

 var request = {
     origin: currentLocation,
     destination: destination,
     travelMode: 'DRIVING',
     drivingOptions: {
         departureTime: new Date(Date.now()),
     }
 };

 directionsService.route(request, function (result, status) {
     if (status === 'OK') {
         directionsDisplay.setDirections(result);
         showTravelTime();
         calculateAndDisplayRoute(directionsService, directionsDisplay, currentLocation, destination);
     }
 });
```

רק אם היעד נמצא ו– status=ok ישלחו הנתונים ל-C++, אחרת לא יתרחש שום דבר והתוכנית לא תיפגע במקרה שהוכנס יעד לא קיים.

22. בבליוגרפיה

new-techonline

zivud

Data

Techtime

Hamichlol

Stayinformedgroup

Fnx

Dwo

Digikey

Wheel

Bennyaviad

Mrcoral

Oracle

Wikibooks

Gov

Developers

Softauthor

Youtube

v7labs

hashdork

Reshetech

Classcentral

Analyticsvidhya

Medium

Wandb

Engineering

Coursera

Campus

Paperswithcode

Kaggle

Paperswithcode

Mysun

Cplusplus

Math

Damada

Python

Keras

Stackoverflow

Github

Geeksforgeeks

W3schrool

React

Opencv

23. סיכום ומסקנות

לכתוב בשלבים אלו את הספר פרוייקט בזמן שהפרוייקט לקראת סיום ואני רואה כבר את הסוף זה משמח מאוד. בתחילה כשהודיעו לנו על הפרוייקט רציתי לקחת פרוייקט קל ופשוט כי לא האמנתי מספיק ביכולות שלי.

כשבאה המורה נילסון והסבירה לנו על הדרישות של הפרוייקט היא נתנה דוגמאות של רעיונות לפרוייקטים ונהיה לי חשק להשקיע בזה כמה שיהיה צריך עד לקבלת פרוייקט מושלם.

עד לרגע הפרויקט לא יצרתי או הכרתי דברים דומים. הכל היה חדש וכל כך מעניין ומסקרן. במהלך הפרויקט התמודדתי לא מעט עם באגים ושגיאות בתחומים שונים, אולם כל אלו חישלו אותי ולימדו אותי הרבה מעבר לכל שיעור. כשהתחלתי לחשוב איך אני הולכת לעשות את הפרוייקט על רכב אוטונומי לא ידעתי מאיפה להתחיל. בהוראת המורה התחלתי לראות קורסים ב-Coursera שלימדו אותי המון רקע על מכוניות אוטונומיות, הארכיטקטורה שלהם, מאיזה חיישנים הם מורכבות ועוד דברים שלא ידעתי. לאחר שקיבלתי את הרקע התחלתי ללמוד קורס מעשי על ניווט, הייתי בטוחה שידריכו אותי שם שלב אחרי שלב איך לכתוב את הקוד אך זו הייתה תקוות שווא כי המרצה בקורס הסביר את הרעיון של ניווט ושיערוך מיקום ו...זרק אותי למים לכתוב לבד את אלגוריתם מסנן קלמן. מסנן קלמן היה הקוד הראשון שכתבתי בקשר לפרוייקט, זה לקח לי המון זמן גם כי זה היה שפה שלא הכרתי, ונדרשתי לתרגם מתמטיקה מהקורס לקוד ב-python ולאחוז ראש איזה מידע אני מקבלת מהחיישנים ואני מעבירה אותו למסנן ואיזה מידע המסנן צריך להחזיר. אחרי שסיימתי לכתוב אותו, הרצתי והוא עבד זה עשה לי סיפוק גדול וחשק להתקדם הלאה ולהצליח.

בדרך היו המון קשיים החל מהתקנת ספריות, גרסאות שונות לשפה, התקנת סביבות עבודה, Jupyter notebook– מושגים חדשים שלא שמעתי עליהם אף פעם, אך ב"ה למדתי מהטעויות ובסוף הצלחתי.

בנוסף רכשתי ידע נרחב ויקר מאוד בשפת C++, התקנת ספריות, פתרון בעיות ובאגים שזה חישל אותי ולימד אותי המון יותר מכל שיעור.

כמו כן במשך העבודה על הפרויקט פיתחתי את היכולת ללמוד חומר לבד, לדעת כיצד להגיע למידע נכון ורלוונטי לפי ערכים נכונים לחיפוש. זה היה מאתגר, חומרים רבים היו לא מוכרים כלל,

חלקם נגעו לנושאים שאף פעם לא נתקלתי בהם ,וברוב הפעמים היו בשפת האנגלית. הלמידה, מעבר לביצוע, הייתה חלק חשוב ובלתי נפרד מהתהליך וקידמה אותי המון. היו נושאים שעבור הלמידה שלהם חזרתי לקרוא גם בפעם השנייה והשלישית על מנת להבין בצורה הטובה ביותר. אין ספק כי למדתי מכך רבות – הרבה מעבר לחומר עצמו. רכשתי מיומנות חשובה של למידה עצמאית, של חיפוש חומרים רלוונטיים וסינון חומרים שאינם כאלה.

מיומנות נוספת וחשובה שרכשתי השנה ואין לי ספק שתלווה אותי בהמשך החיים זה היכולת לשבת מול מחשב שעות ארוכות, לא להתייאש כאשר יש באגים לנסות שוב ושוב כל פעם דרך אחרת עם הרבה סבלנות, כי בסוף יש פתרון לכל בעיה. כמו כן ניהול נכון ותקין של תהליכים מרובים שימוש ב-mutex ו- thread.

מסקנותיי מהפרוייקט:

חשוב להתחיל לעבוד עליו מייד בתחילת שנה, לחפש חומרים וללמוד טוב טוב את הרקע, זה מסייע המון במהלך כתיבת הפרוייקט ומחדד נקודות שלא בטוח הייתי חושבת עליהם.

בנוסף לפצל את הפרוייקט לתתי פרוייקטים ולעשות עבודת חקר באיזה צורה הכי נכון לבצע אותו וכמה זמן מתאפשר לי להקדיש לו.

כמו כן לפני כל צעד חשוב לדבר עם המורה לקבל אישור או הכוונה ורק אז לעשות.

במבט לאחור, כאשר הפרויקט היה נראה אתגר גדול כל כך, אני יודעת שעם השקעה גדולה , סבלנות ומאמץ – מגיעים להצלחה. העבודה הקשה שווה ומשתלמת ומעניקה תחושת סיפוק גדול וכן הכינה אותי טוב לקראת היציאה לשוק העבודה.

24. תודות

ראשית תודה לבורא עולם על היכולות והכישרונות שחנן אותי. לאורך כל הדרך הרגשתי איך ה' מלווה אותי בכל צעד ושעל, אף פעם לא הייתי לבד!

תודה ענקית למנחת הפרוייקטים – המורה רבקה נילסון על שתמיד האמינה בי, נתנה לי לעוף קדימה, עודדה בזמנים קשים, על הכוונה וייעוץ לאורך כל חלקי הפרוייקט (אפילו בזמני חופשה) והכל עם אווירה טובה וחיוך.

תודה לרכזת המגמה- המורה אלה גליק על הקשבה, עידוד, תמיכה ועזרה מקצועית לאורך כל חלקי הפרוייקט ותמיד מתוך כיף ושמחה.

למתרגלות המורה ציפי פולצ'ק והמורה תמר זקס על עזרה, סיוע והכוונה עם המון סבלנות ורוגע.

לצוות מורות תכנות המסור, על הנכונות הרבה, האווירה הטובה בשיעורים, על זה שלא ויתרתם לנו גם כשהיה קשה כדי להביא אותנו לתוצאות מקסימליות וכן על סבלנות אינסופית לייעץ לבדוק לתמוך וללמד והכול מתוך רוח טובה ונעימה .

לסמינר בית יעקב רכסים ולעומד בראשו הרב בן שלום על ההשקעה, המקצועיות והמסירות והכל כדי שנצליח.

להורים היקרים שלי ולמשפחתי על עידוד ותמיכה לאורך כל הדרך החל מתחילת הלימודים ועד סוף פרוייקט הגמר.

ואחרונות חביבות, בנות המגמה המהממת שלנו! על עידוד ותמיכה בזמנים קשים, על התעניינות בלתי פוסקת, על הקשבה, יעוץ, תמיכה ופירגון לאורך כל הדרך, על ההקפדה על אווירה טובה ונעימה גם בזמנים לחוצים. אין עליכם בעולם!!!!