The Andrew & Erna Viterbi Faculty of Electrical Engineering Technion — Israel Institute of Technology

Electronics
Computers
Communications

VISL Vision and Image Sciences Laboratory

הטכניון - מכון טכנולוגי לישראל
הפקולטה להנדסת חשמל ע"ש אנדרו וארנה ויטרבי
המעבדה לראייה ומדעי התמונה

**דו"ח סיכום פרויקט שנתי**

# Real-time track detection via LiDAR technology for an autonomous formula car

מבצעים:

**אביב רגב** **Aviv Regev**

**אבינועם ברהום** **Avinoam Barhom**

מנחה: **דני וייכרמן** **Danny Veikherman**

**סמסטר רישום: אביב תשע"ח**

**תאריך הגשה: אפריל, 2019**

**בשיתוף עם: פורמולה אוטונומית טכניון 2018**

FORMULA TECHNION DRIVERLESS

# תוכן עניינים

# רשימת איורים

# רשימת טבלאות

# תקציר

פרויקט מיפוי המסלול עבור פרויקט רכב הפורמולה האוטונומי נולד באוקטובר 2018, לאור פתיחת מקצה אוטונומי באיגוד הפורמולה הסטודנטיאלי (Formula Student Automotive Society – FSAE). איגוד זה מארח שורה של תחרות בינלאומיות בין סטודנטים להנדסה, אשר אחראים על בנייה, עיצוב והפעלה של רכב אוטונומי מקצה לקצה.

כחלק מהצורך הבסיסי הקיים ברכב אוטונומי לזהות את המיקום שלו ביחס למסלול (אשר לרוב אינו ידוע מראש למתחרים), דרוש היה למצוא פתרון לזיהוי הקונוסים אשר תוחמים את מסלול התחרות בזמן אמת. הפתרון שיוצג בפרויקט זה, מסתמך על טכנולוגיית ה- LiDAR (Light Detection And Ranging), אשר מספקת פלט בדמות מרחק וזווית (אזימוט) עבור כל נקודה במרחב סביב החיישן.

כפי שיוסבר בפירוט בהמשך ספר זה, זיהוי הקונוסים במסלול יעשה באמצעות חיישן VLP-16 של חברת Velodyne, באמצעות שימוש בחלק קטן מתוך 16 לייזרים הקיימים, כאשר לאורך סריקת הפריים של כל לייזר אנו למעשה מחפשים קפיצה/שינוי משמעותי בקריאת המרחק מהנקודות בסביבה אותן "רואה" החיישן, וזאת על מנת לזהות את הקונוסים הפזורים במסלול במערכת הצירים של חיישן ה- LiDAR.

לאחר מכן, על מנת ליצור מיפוי בזמן אמת של המסלול (תוך כדי תנועת התקדמות הרכב) אנו מבצעים אינטגרציה של פלט חיישן ה- LiDAR עם חיישן אינרציאלי (Inertial Measurement Unit- IMU), VN-200 של חברת VectorNav, וזאת על מנת לקבל את הנקודות החשודות כקונוסים במערכת הצירים של העולם, ובכך למעשה אנו מבטלים את המריחה המתקבלת מהפלט של ה-LiDAR .

בשלב האחרון, אנו נעזרים באלגוריתם קיבוץ הנקודות (clustering) לשם מציאת מרכזי המסה של כלל קבוצות הנקודות שקיבלנו במערכת ייחוס הגלובלית. האלגוריתם שנבחר הינו K-means אשר מספק K מרכזי מסה לפי דרישת המפעיל (למעשה מדובר על מספר הקונוסים הצפויים להיות בפריים). כאמור פעולה זו נעשית על מנת לקבץ כל קבוצה כזו לנקודה בודדת שתציג קיומו של קונוס על גבי המסלול הנבנה תוך כדי תנועת הרכב, ובכך לקבל מיפוי מלא של מסלול הרכב.

# Abstract

This real-time track detection project, for an autonomous Formula race car was born on October 2018, as part of an international series of racing competitions of the Formula Student Automotive Society (FSAE).

"FSAE" organization is challenging engineering students to design and build a one-of-a-kind racecar to compete in international competitions against top-tier universities.

As part of the track detection and foresight needed to navigate the car along the unknown track, a sensing system able to detect the sideway cones, is needed as so path planning can be achieved.

To do so, a track detection algorithm is needed, which will collect its data using LiDAR sensor, which able to extract the exact distance and angel to the cone's position along an unknown track.

Our cone detection method relies on VLP-16 LiDAR sensor (Velodyne), while in each of its scans we are using few of its 16 lasers, in purpose of searching a specific sudden rise in the radius values which emphasize that we are "looking" at a cone object in the LiDAR's coordinate system.

Afterwards, to achieve real-time route mapping (while the vehicle continues driving throw the track), we preform integration between the LiDAR's output to an Inertial Measurement Unit (IMU) VN-200 (VectorNav), to receive all the suspected points on the route in the world's coordinate system.

In this way we dismiss the spreading received by the continues LiDAR's output.

Finally, we are using K-means clustering algorithm to convert each cluster in the world's coordinate system to one point on the map, which is being built with the car's advance throw the runway.

# תיאור מטרות הפרויקט

פרויקט זה בא לענות על הצורך הבסיסי הקיים ברכב אוטונומי לזהות את המיקום שלו ביחס למסלול אינו ידוע מראש, על מנת לגבש יכולת ניווט ובקרה בהתאם לזיהוי נקודות הציון המבוקשות בדרך.

בפרויקט זה נתמקד במציאת פתרון עבור זיהוי הקונוסים אשר תוחמים את מסלול תחרות הפורמולה האוטונומית. להלן המטרות העיקריות עליהן נשים דגש בפרויקט זה:

א. בניית יכולת זיהוי קונוסים מתוך פלט ענן הנקודות המתקבל מחיישן המרחק הייעודי שנבחר.

ב. המרת כל נקודה מבוקשת ממערכת הייחוס של החיישן אל מערכת ייחוס גלובלית, באמצעות אינטגרציה עם חיישן אינרציאלי (IMU).

ג. בניית יכולת שמירת נתוני פלט האינטגרציה בזמן אמת וסינון מידע לא נחוץ.

ד. מתן פלט גרפי של מיפוי מסלול הרכב בזמן אמת, תוך שימוש באלגוריתם k-means למציאת k מרכזי מסה של נקודות החשודות כקונוסים.

# השוואת חיישני מרחק

זיהוי קונוסים במסלול מוטורי בזמן אמת על ידי חיישן למדידת מרחק מהווה אתגר בעיקר בשל מהירות התנועה הגבוהה הצפויה בתחרות הפורמולה. אתגר הזיהוי בזמן אמת ובמהירות גבוהה מצריך בראש ובראשונה בדיקה מקפה של חיישני מרחק הקיימים בשוק המסוגלים לספק מענה עבור אתגר זה.

באפיון התחלתי של בעיית זיהוי העצמים נמצא כי התכונה הקריטית מבחינתנו בבחירת החיישן הינה קצב הדגימות לשנייה אשר נובע מכמות שליחת "נקודות האינפורמציה" לשנייה ( points per second) אותו מספק החיישן אל צד שלישי (במקרה שלנו, מדובר על מערכת עיבוד המידע). לכן בהשוואה שלפניכם אחד הדגשים העיקריים יהיה על תכונה זו, וכמו כן יבדקו המאפיינים הבאים של החיישן: מס' ערוצי מידע, טווח, מידת דיוק (מרחק), שדה ראייה (מאוזן/מאונך), רזולוציה זוויתית, תדר עבודה, הספק צריכה, מתח הזנה, טמפרטורת עבודה, משקל ומאפייני מפתח נוספים.

להלן החיישנים העיקריים אותם בדקנו במהלך חקר הנושא:

## FR 55-RLAP

**FR**- photo reflective, 55(housing size), **R**ed **L**aser, **A**nalog output, high **P**recision

חיישן מרחק באמצעות לייזר אדום (655 nm) אשר נועד למדידת מרחקים גדולים למטרת זיהוי מטרות, בקרת תנועה ומניעת התנגשויות. החיישן עובד באמצעות טכנולוגיית מדידת מרחק שמחשבת את זמן החזר קרן הלייזר מרגע שידורה. מדידת המרחק מתבצעת באמצעות קרן לייזר אחת (ליניארי), כאשר מדידת המרחק מתבצעת באופן אנלוגי בהתאם לכמות הפוטונים שנקלטו בגלאי (זרם).

יתרונות עיקריים: זול, משקל נמוך, צריכת הספק נמוכה.
חסרונות עיקריים: מפיק מידע אנלוגי מאוד מצומצם (כמות החזר פוטונים- נמדד ביחידות mA), כמות אינפורמציה לשנייה מאוד נמוכה (קצב עבודה נמוך).

*איור 1 - חיישן FR 55-RLAP*

## MB7583 SCXL-MaxSonar- WR

**S**elf-**C**leaning, **H**igh **R**esolution, **W**eather **R**esistant, Ultrasonic Sensor

חיישן מרחק באמצעות גל קול, אשר נועד למדידת מרחקים בינוניים/קטנים עבור מטרות זיהוי מטרות. החיישן עמיד מאוד לתנאי מזג האוויר קשים, ובעל אלגוריתם המקנה יכולת גבוהה לביצוע סינון רעשים סביבתיים.
קיימת אפשרות לבחירה בין שני סוגי חיישנים עיקריים, כאשר הראשון מטרתו לזהות את המרחק מהאובייקט הגדול ביותר בסביבה, והסוג השני מאפשר למדוד מרחק מהאובייקט הקרוב ביותר לחיישן.
מדידת המרחק מתבצעת בפועל על ידי שידור פולס (גל קול, תדר לא מפורט) מהחיישן ברוחב אלומה ספציפי (כתלות במתח אספקה), כאשר המרחק עצמו נמדד לפי מדידת

הזמן שעבר מרגע השידור עד רגע קבלת ההחזר של גל הקול. מדד המוצא הינו מתח אנלוגי התלוי ב- Time of Flight.

חיישנים אלו אינם מושפעים מצבעו של האובייקט או ממאפיינים ויזואליים אחרים.

יתרונות עיקריים: זול, צריכת הספק נמוכה מאוד.

חסרונות עיקריים: כמות אינפורמציה לשנייה נמוכה, מוגבל מרחק.

*איור 2 - חיישן MB7583 SCXL-MaxSonar*

## VLP-16 Velodyne

חברת Velodyne מייצרת כמה דגמים של חיישני מרחק המתבססים על שידור אותות לייזר (903nm) ופועלים על בסיס מדידת הזמן שעבר מרגע תחילת השידור עד לקליטתו. היתרון היחסי של מוצרי החברה על פני שאר החיישנים הקיימים בשוק הינם קצב הדגימה הגבוה מאוד (המאפשר שימוש במוצרים על גבי רכבים אוטונומיים) והמידע הרחב המסופק בכל דגימה שכזו, שהינו: מרחק לכל נקודה במרחב, זווית לנקודה, רמת ההחזר (תלויה בסוג חומר האובייקט וצבעו) וחותמת זמן.

בנוסף, בשונה משאר החיישנים בשוק, בחיישנים אלו ניתן לבצע סריקה מרחבית של 360° בכל דגימה.

החיישנים העיקריים המוצעים כיום על ידי החברה הינם-

VLS-128, HDL-64E, HDL-32E, PUCK 16 (Hi-res\Lite).

המספר המצוין בדגמים אלו, למעשה מתאר את כמות הלייזרים שדוגמים בבת אחת את המרחב (בכל פקטת מידע), ולכן ביחס ישר ככל שכמות הלייזרים גדולה יותר, כך רזולוציית המפתח הזוויתי האנכי גדלה. בשל המחירים הגבוהים מאוד של הדגמים המתקדמים, נתמקד בהשוואה זו בדגם PUCK 16.

חיישן זה בעל 16 לייזרים, המאפשרים סריקת המרחב ב- °360, ובמפתח זוויתי של °15 מהציר הליניארי לחיישן. קצב העברת המידע בחיישן זה הינו 754 פקטות לשנייה, כאשר כמות הנקודות המסופקות בפקטה אחת הינה-

12 בלוקי מידע * 16 לייזרים * 2 דגימות בבלוק = 384 נקודות מידע לפקטה.

למעשה, בפועל בשנייה אחת אנו מגיעים ל- 300,000 נקודות מידע שמסופקות למערכת העיבוד.

יתרונות עיקריים: מרחק גבוה, כמות אינפורמציה לשנייה גבוהה מאוד, מידע איכותי ואמין.

חסרונות עיקריים: מחיר יקר מאוד, מידות, הספק גבוה יחסית.


*איור 3 - חיישן VLP-16 Velodyne*

**להלן סיכום כלל המאפיינים שנבדקו בחיישני המרחק:**

| **Sensor** | **VLP-16 Velodyne** | **FR 55-RLAP** | **MB7583 - SCXL** |
|---|---|---|---|
| Work method | Laser- 903 nm wavelength | Laser- 655 nm wavelength | Time of flight for sound |
| Num of channels | 16 | 1 | 1 |
| Range(m) | 1-100 | 0.3-70 | 0.3-5 / 0.5-10 |
| Accuracy(cm) | 3± | 8 (µA) | 5mm / 10mm |
| Field of view (horizontal) | °360 | linear | 5° to -5°+ |
| Field of view (vertical) | 15° to -15°+ | - | - |
| Angular resolution | 5 HZ- 0.1°<br>10 HZ- 0.2°<br>20 HZ- 0.4° | - | $\frac{1\mu s}{mm}$<br>Pulse width resolution |
| Rotation rates(Hz) | 5 - 20 | - | - |
| Power consumption (w) | 8 | 1.8 - 3 | 340 mw |
| Operating voltage (VDC) | 9 - 32 | 18-30 | 2.7 - 5.5 |
| Operating temperature | -10° to + 60° | -30° to + 60° | -40° to + 65° |
| Weight (g) | 830 | 42 | - |
| Output (points per sec) | 18,750 per laser | 1 | 1,066 |
| Repetition Rate | 6 ns (duration)<br>1.44µs * 16 lasers per pattern for a period of 46.1 µs = 21.7KHz | 10ms response time = 100Hz | Serial mode- 9600 baud rate, with package size of 9 bits. |
| Special features | Contains UDP packets, which includes: distances, Calibrated reflectivity, Rotation angles, time stamps, $GPRMC NMEA sentence | Analog measurement, 4 distance positions with 2 switching visible laser ,outputs light | Self-cleaning algorithm, Internal temperature compensation, 2 operational modes (largest/closest) |
| Price | 8000$ | 300€ | 175$ |

*טבלה 1 - השוואת מאפייני חיישני מרחק*

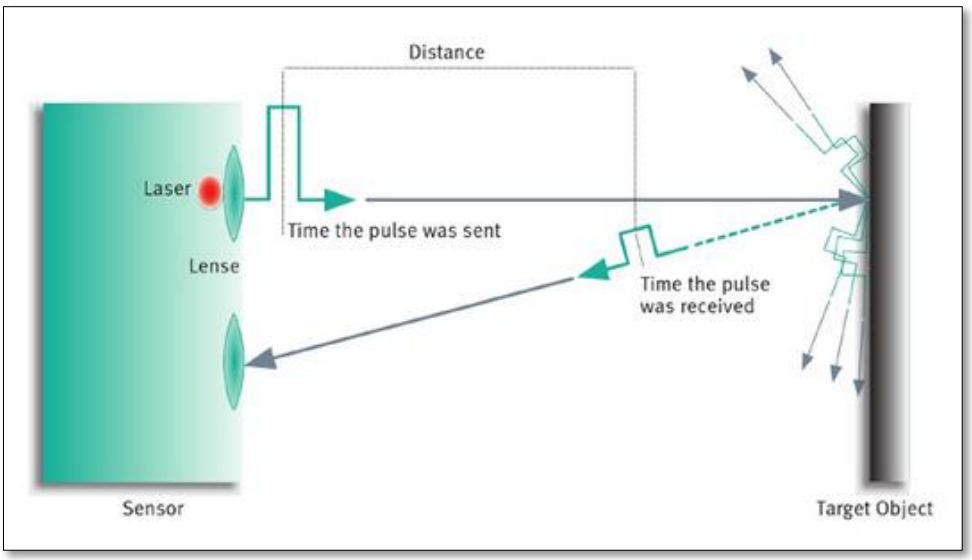
לסיכום, כפי שניתן להתרשם מבחינת איכות המידע וכמות הדגימות לשנייה, הבחירה האופטימאלית לצרכי זיהוי אובייקטים במהירות גבוהה הינה חיישן מרחק VLP-16 של חברת Velodyne הפועל על פי טכנולוגיית ה-LiDAR. מגבלת המחיר הגבוה של חיישן זה, הינה המשמעותית ביותר מבחינת המשאבים המצומצמים שעמדו לרשותנו בהגדרת פרויקט זה, אך העובדה כי חיישן זה נרכש בעבר עבור מעבדת VISL בפקולטה להנדסת חשמל בטכניון, למעשה אפשרה לנו מראש להתייחס אל האפשרות להשתמש בו כשרירה וקיימת.

# Pulse Ranging Technology - PRT

ניתן להיווכח כי כלל חיישני המרחק שנבדקו בהשוואה במהלך פרויקט זה, מסתמכים על אופן חישוב המרחק מהאובייקט באמצעות חישוב זמן החזרת קרן האור/גל הקול בשיטה שנקראת Time of Flight.

מרחק בין 2 נקודות מחושב על פי הנוסחה- $D = \frac{ct}{2}$ , כאשר $c$ הינו מהירות האור, ו- $t$ הינו כמות הזמן הכולל שלקח לקרן האור/גל הקול לעבור בתווך מרגע השידור ועד רגע החזרה למקלט. ניתן לחשב זמן זה כך- $t = \frac{\varphi}{\omega}$ , כאשר $\varphi$ הינו פאזת הגל, ו- $\omega$ הינו המהירות הזוויתית של הגל.

דיוק המדידה בשיטה זו נקבע על פי זמן העלייה/ירידה של פולס אות הלייזר ועל פי מהירות תגובת המקלט. חיישן שמשתמש בקרן לייזר בעלת אלומה חדה ובמקלט בעל זמן תגובה מהיר יכול להגיע לדיוק מדידה של מילימטרים בודדים. באיור הבא, ניתן להתבונן באופן בו מדידת המרחק מתבצעת:

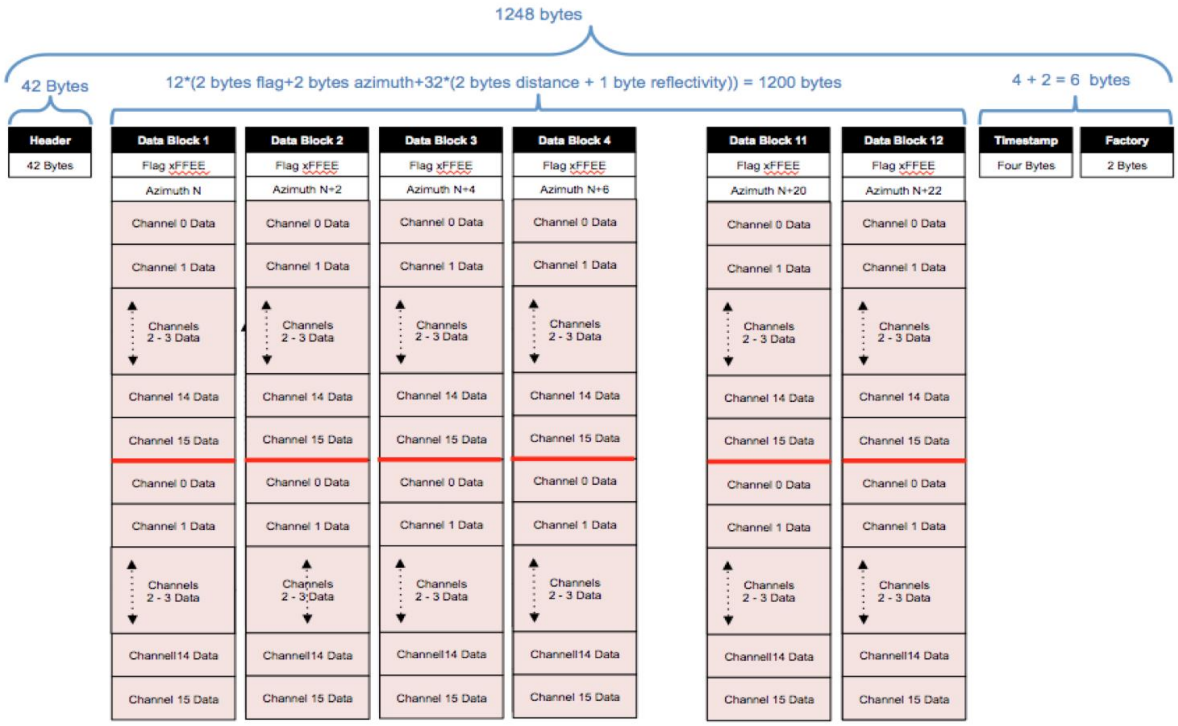



*איור 4 - תיאור Time of Flight*

# פלט LiDAR VLP-16

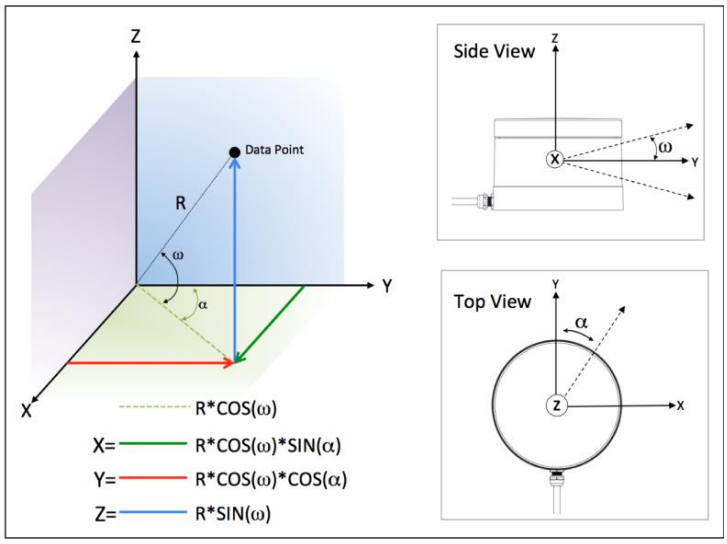
יצירת התקשורת עם חיישן ה- LiDAR נעשית באמצעות פרוטוקול UDP דרך פורט מספר 2368 ובכתובת IP מספר 192.168.1.201 , כאשר כל פקטת מידע המתקבלת מהחיישן הינה בגודל 1248 Byte (מתוכן 1206 Byte מידע).

כל פקטת מידע המגיעה מהחיישן [1] מורכבת מ- 24 מחזורי יריות של 16 לייזרים כפי שניתן לראות בבירור באיור 5. כל מחזור ירייה מורכב מ-16 ערכי מדידה של סדרת הלייזרים בחיישן (פרושים בקו אנכי, כאשר ההבדל בין כל לייזר הינו 2 מעלות). המידע שנותן כל לייזר הוא מרחק מהנקודה בה פגע + רמת ההחזר/רפלקטיביות של הנקודה. בנוסף בכל מחזור ירייה זוגי, כלומר רק ב-12 מחזורים, אנו גם מקבלים מידע אודות האזימוט של המחזור כולו (זווית מרחבית), כך שבעזרת אינטרפולציה פשוטה ניתן למצוא גם את הזווית המרחבית של מחזורי הירייה האי זוגיים.
לכן, בסה"כ אנו מקבלים 384 (24 מחזורי ירייה * 16 לייזרים) נקודות מידע על כל פקטה, כאשר לכל פקטה מוצמדת חותמת זמן ייחודית (פנימי של המכשיר- הזמן מרגע הפעלת המכשיר).



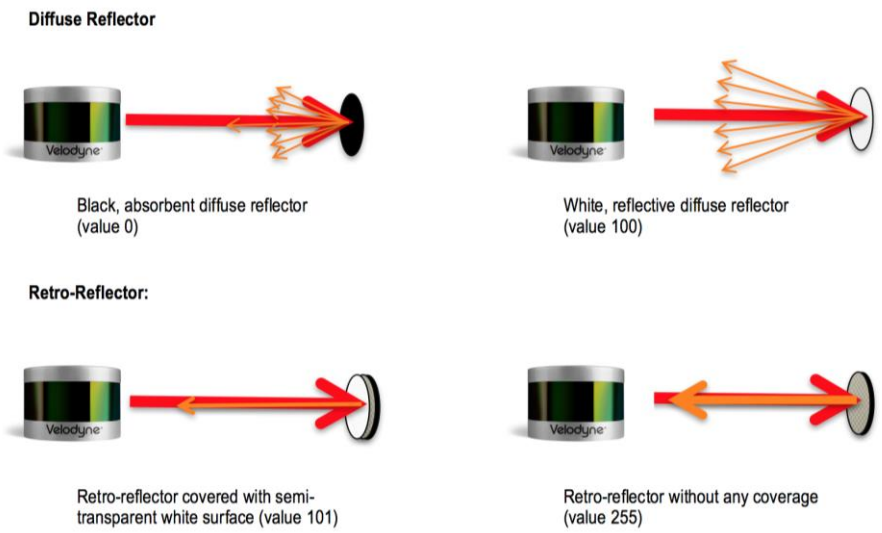

*איור 5 - מבנה פקטת מידע* VLP-16

באמצעות מספר פעולות אלגבריות פשוטות ניתן להמיר את ההצגה הפולארית של הנקודות (מרחק, זווית אנכית וזווית מרחבית) המתקבלת כפלט מהחיישן לקואורדינטות קרטזיות. להלן ההמרה שביצענו כאמור בקוד הפרויקט על מנת להציג כל נקודה במרחב באמצעות תיאור קרטזי (X,Y,Z):




*איור 6 - תיאור המרה מפולרי לקרטזי*

כפי שהוזכר קודם לכן, מלבד ערכי המרחק, הזווית המרחבית, הזווית האנכית וחותמת זמן, אנו גם מקבלים כפלט מהחיישן את רמת ההחזר של העצם בו פגעה קרן הלייזר. על פי מדריך ההפעלה של החיישן ערך ההחזרה הינו מספר עשרוני הנע בטווח 0-255 , כאשר למעשה ערך זה מושפע מצבעו, מבנהו, חומרו של האובייקט. לדוגמא, כפי שניתן לראות באיור 7, אנו מצפים לקבל רמת החזר מאובייקט שחור לגמרי (בולע את קרן הלייזר) ערך 0.




*איור 7 - רפלקטיביות*

# בדיקת היתכנות - חקר ביצועים

בבחינת נתוני היצרן שנעשתה, נלקחו הנתונים הרלוונטיים הבאים על מנת להבין את טיב מדידות המרחק אותן נבצע:

$$\text{Work freq} = 600\ \text{RPM} = 10\ \text{rounds per sec} = 1\ \text{round per } 0.1\ \text{sec}\ (360°)$$

All sixteen lasers are fired and recharged every 55.296μs.

$$\Longrightarrow \frac{0.1\ \text{sec}\ (360°)}{55.296\mu\text{s}} \approx 1808\ \text{lasers sets per round (set} = 16\ \text{lasers fired)}$$

$$\Longrightarrow \frac{1808 * 16}{360°} \approx 80\ \text{data points per degree}$$

מאחר וקיים בידנו מידע אודות המרחקים הצפויים להיות בין קונוסים על גבי המסלול, סביר מאוד להניח כי בעת פגיעה של קרן לייזר בקונוס ספציפי, אנו למעשה נקבל נתונים אך ורק מלייזר אחד (מתוך 16 אופציונליים) וזאת לאור העובדה כי בין כל לייזר ללייזר ישנן 2 מעלות הפרש מבחינת הזווית האנכית בו הם ממוקמים פיזית בחיישן. לאור הנחה מחמירה זו, נחלק את כמות המידע למעלה בכמות הלייזרים שיש ברשותנו:

$$\Longrightarrow \frac{80\ \text{data points per degree}}{16} = 5\ \text{data points per degree}$$

יש לציין כי על מנת לוודא כי החישוב התיאורטי אכן מדויק, ביצענו מדידת פלטי הנקודות של החיישן בפועל, וקיבלנו שהתוצאה מאוד קרובה. בפועל ב- $360°$ החיישן מספק 28,960 נקודות, ולאחר חלוקה ב- 16 לייזרים וב- $360°$ , קיבלנו 5.02 נקודות למעלה (החלוקה בוצעה על מנת לקבל את כמות הנקודות ללייזר אחד למעלה אחת).

כעת נתבונן באובייקט הרלוונטי לפרויקט זה, עליו למעשה אנו מתכוונים לבצע תהליך זיהוי באמצעות חיישן ה- VLP-16 . מידות הקונוס ידועות ולקוחות מספר חוקי התחרות (תיאור הקונוסים שישמשו אותנו בתחרות ומידותיהם מופיעים באיור 8). נבצע שלוש מדידות של "אורך ויזואלי" בשלוש מקומות שונים לאורך הקונוס- תחתית, אמצע, פיה עליונה.

"אורך ויזואלי" הוגדר על ידנו כמחצית ההיקף פחות 70% מערך הרדיוס, כאשר ההיקף נמדד 3 פעמים בהתאם למיקום הרלוונטי על גבי הקונוס.

הכוונה בחישוב זה הינה לבצע חישוב מחמיר, בו אנו מדמים את המציאות בה קרני הלייזר למעשה אינם יכולים לפגוע לכל אורך מחצית ההיקף, ולכן דרוש תיקון נוסף שבמקרה שלנו הוחלט על 0.7 מערך הרדיוס.

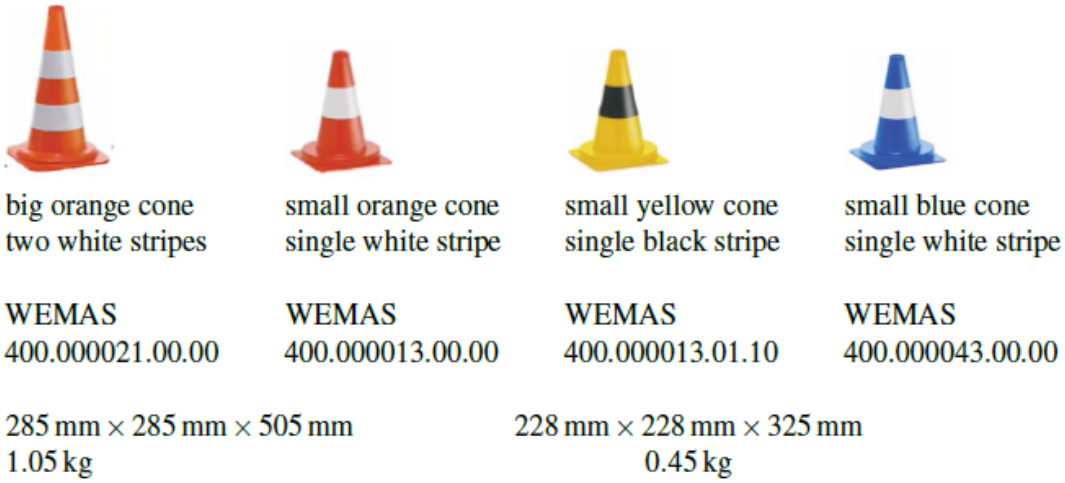
להלן ההיקפים שנמדדו ידנית ובהתאם שאר חישובי ה-"אורכים הוויזואליים" המתאימים לכל אזור בקונוס:

היקף תחתית- 46cm , היקף אמצע- 33cm, היקף פיה עליונה- 14.5cm

$$\text{Length}_{bottom} = r_{cone\_bottom} * \pi - (r_{cone_{bottom}} * 0.7) = 23 - 5.12 \approx 17.5\ cm$$

$$\text{Length}_{middle} = r_{cone\_middle} * \pi - (r_{cone_{middle}} * 0.7) = 16.5 - 3.66 \approx 12.5\ cm$$

$$\text{Length}_{up} = r_{cone\_up} * \pi - (r_{cone_{up}} * 0.7) = 7.25 - 1.61 \approx 5.5\ cm$$

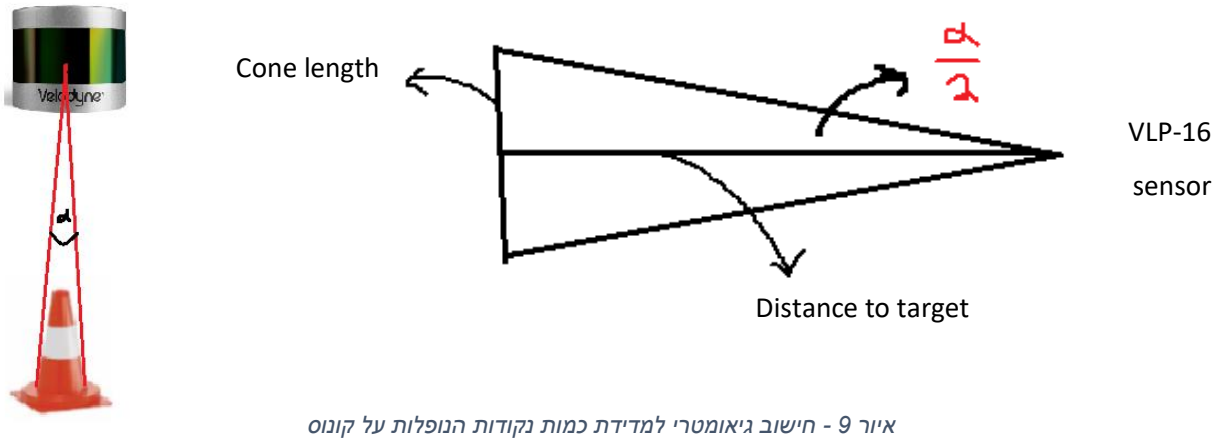



איור 8 - מידות וצבעי קונוסים בתחרות

<u>חישוב כמות הנקודות כפונקציה של מרחק האובייקט מהחיישן:</u>

בשלב זה, כאשר יש בידנו את הנתונים של אורכי מטרת הפגיעה של קרני הלייזר ב- 3 מיקומים שונים על הקונוס, אנו יכולים לבצע חישוב גיאומטרי שיבהיר את כמות נקודות המידע שאנו מקבלים עבור כל מיקום שונה בקונוס בתלות במרחק הקונוס מהחיישן.

חישוב גיאומטרי זה מתבסס על האיור הבא:



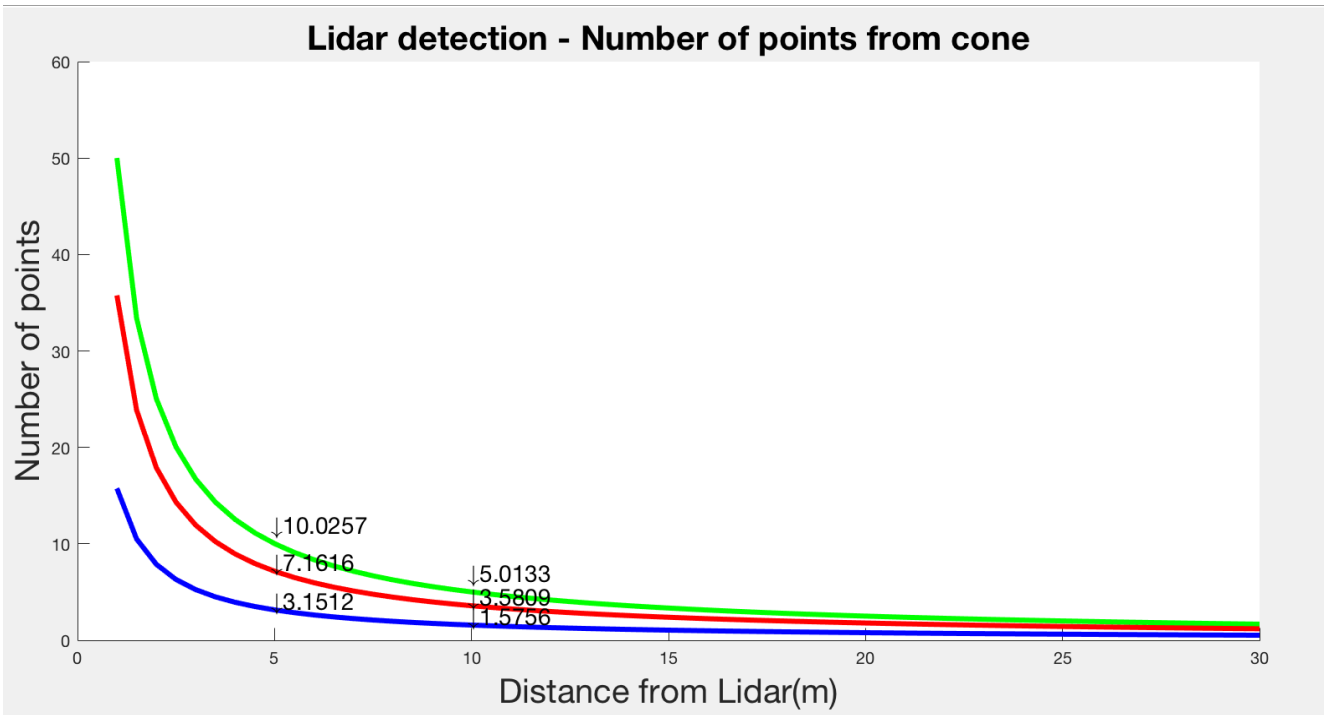

*איור 9 - חישוב גיאומטרי למדידת כמות נקודות הנופלות על קונוס*

כלומר החישוב המתמטי הבא מבטא בדיוק את כמות הנקודות הקיימת בכל אזור של הקונוס כתלות במרחק הקונוס מהחיישן:

$$Points\ on\ cone = 2 * (5 \text{ data points per degree} * degree\ size)$$

$$\Longrightarrow Points\ on\ cone\ =\ 10 *\ \tan^{-1}\left(\frac{\frac{Cone\ length}{2}}{Distance\ to\ target}\right)$$

כעת נציג את התוצאות בצורה גרפית:




איור 10 - גרף כמות נקודות שמתקבלות על קונוס כפונקציה של מרחק מהחיישן

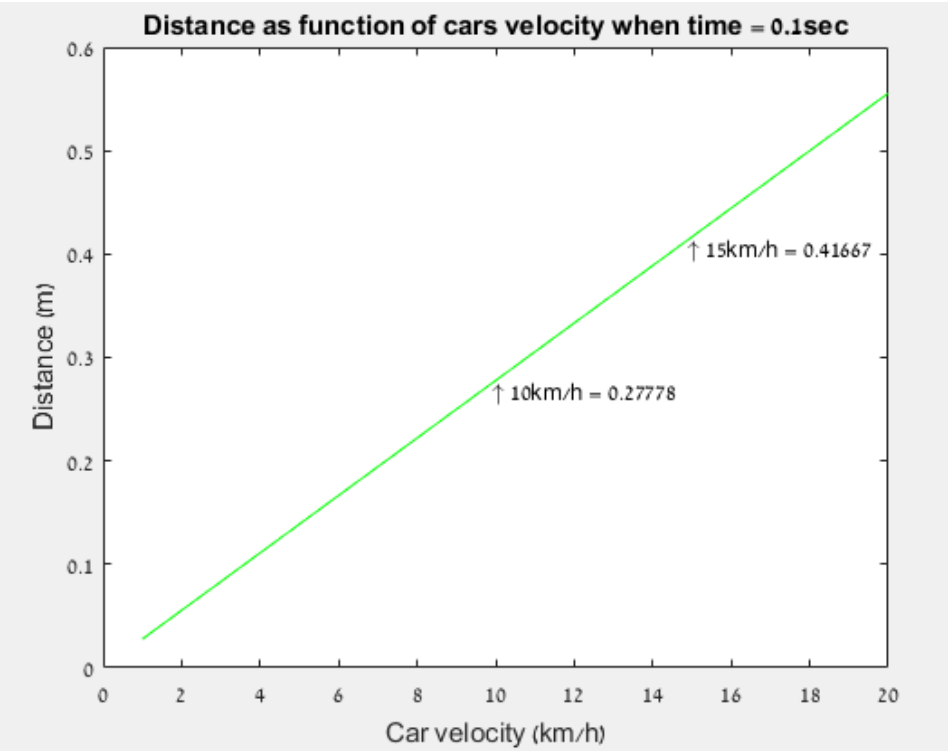
כפי שניתן ללמוד מהגרף שהתקבל, כאשר החיישן יזהה קונוס במרחק של כ- 5 מטרים ממנו אנו יכולים לצפות כי לכל הפחות יהיו ברשותנו כ- 20 נקודות של מידע, וזאת כאמור בתלות במיקום פגיעת הלייזר ביחס למבנהו של הקונוס (במרווח דגימה מספיק גדול של כמה עשיריות השנייה, בהן הרכב מתקדם, למעשה אותו הלייזר יסרוק את הקונוס לאורך כל מבנהו).

חישוב מרחק שעובר הרכב בהתאם למהירותו:

בהתחשב בנתונים אליהם הגענו עד כה, שכוללים את העובדה שהחיישן מבצע דגימת $360°$ כל 0.1sec, נוכל גם למצוא כעת כמה מרחק מתקדם הרכב במשך זמן זה, כפונקציה של מהירות הרכב. מציאת נתון זה, חשוב לנו כי למעשה בזמן בו החיישן מבצע סיבוב דגימה שלם הרכב ממשיך להתקדם ללא דגימה נוספת, כך שנוצר לנו "עיוורון" מסוים של קבלת נתוני סביבה חדשים.

במילים אחרות, מציאת המרחק כתלות מהירות הרכב תאפשר לנו להבין יותר לעומק בסופו של תהליך מהי המהירות הקריטית בה הרכב יכול לנסוע במסלול בהתחשב בעובדה כי הוא איננו מקבל נתוני מרחק מהחיישן.

להלן התוצאות אותן קיבלנו בחישוב מרחק ההתקדמות של הרכב כתלות במהירותו:




איור 11 - מרחק התקדמות הרכב כתלות במהירותו

כפי שניתן להבחין בגרף, במהירות של 10 קמ"ש הרכב יעבור מרחק של 0.28 מטר ללא קבלת נתוני מידע חדשים על הדרך, ובמהירות 15 קמ"ש הוא יעבור מרחק של 0.42 מטר.

# תקשורת עם חיישן ה- LiDAR

לאור כל הנתונים שהוצגו בפרקים הקודמים, הוחלט כאמור להשתמש בחיישן ה- VLP-16 ולכן השלב הבא בתהליך היה יצירת תקשורת עם החיישן, קבלת מידע רציף בזמן אמת והצגתו על גבי מסך על מנת שנוכל ללמוד על יכולות החיישן ביתר פירוט. בשלב זה, נוצלה העובדה כי במהלך סמסטר קודם זוג סטודנטים[1] כבר יצר תקשורת עם החיישן ולכן תחילה הוחלט להשתמש בקוד שלהם הנוגע ליצירת תקשורת וקבלת מידע מהחיישן. החלקים הרלוונטיים בהם השתמשנו היו קבצי הקוד הבאים שנכתבו לסביבות MATLAB, ו- Python 2.7:

- get_LiDAR_data.py
- decryptor.py
- text_reader_off_line.m

בקבצי קוד אלו מצאנו דרך גישה ראשונית ליצירת תקשורת בסיסית עם החיישן על מנת לקבל מידע מפוענח, כאשר כתיבת הפקטות המפוענחות נעשתה לקובץ טקסט שאותו לאחר מכן היה ניתן להציג כענן נקודות במרחב ע"י שימוש בקטע קוד שנכתב עבור סביבת MATLAB.

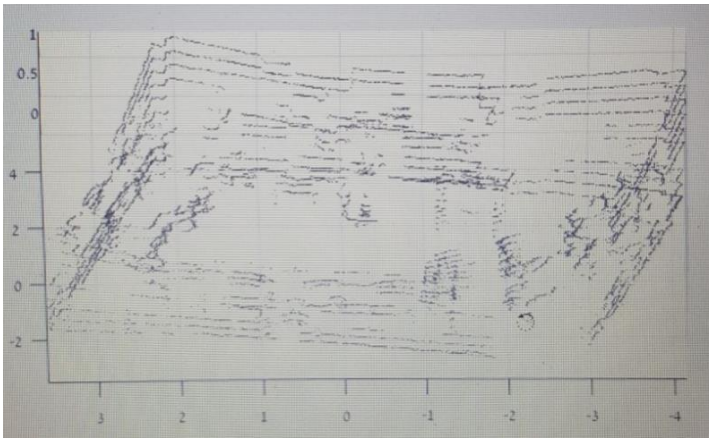
פענוח פקטות המידע בקבצי קוד אלו, נעשה מתוך כלל הלייזרים של החיישן והציג בצורה קרטזית (X,Y,Z) כל נקודה שהתקבלה מהחיישן. מכיוון שאנו רצינו שליטה על הנתונים שמפוענחים בטרם הוצאתם כפלט למערכת הבאה בשרשרת, אז היינו צריכים לבצע שינויים בקבצי קוד הללו אשר אפשרו לנו בין השאר-

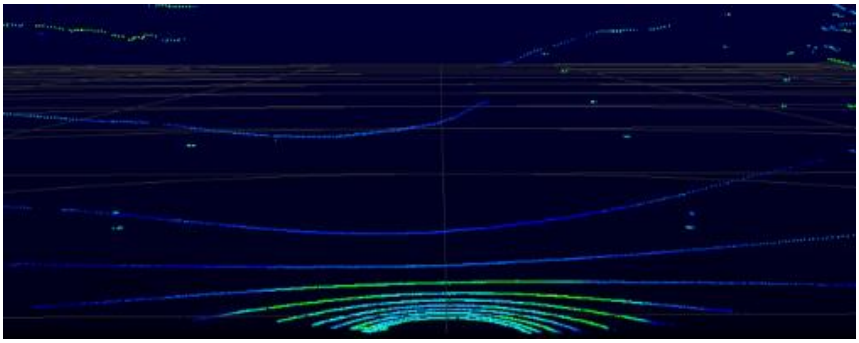
- בחירת הלייזרים הספציפיים אותם נבחר לייצא כפלט, כלומר בחירת הזווית האנכית (elevation degree) שבה נרצה לסרוק את המרחב.
- בחירת מבנה הפלט המיוצא למערכת הבאה- כלומר, מתן אפשרות למשתמש לבחור אילו עמודות יופיעו בפלט כאשר האופציות שניתנו הן- מיקום X, מיקום Y, מיקום Z, רדיוס, זווית מרחבית (אזימוט), רמת החזר (רפלקטיביות), חותמת זמן.
- הוספת מידע אודות רמת ההחזר של כל נקודה במרחב (בהנחה כי נוכל להשתמש במידע זה על מנת לבחון שימוש עבור מתודת זיהוי הקונוסים בהמשך), ובנוסף הוספת חותמת זמן ששימשה ככלי לסנכרון עם חיישנים אחרים שייקחו חלק בתהליך האינטגרציה עם חיישן המרחק.

[1] רובוט ממפה מנהרות אוטונומי LiTANK, אשר נכתב ע"י אביב רובינשטיין ותום ארז בסמסטר אביב תשע"ה

כפי שהוסבר לעיל, בשלב הראשוני בו יצרנו תקשורת עם החיישן המטרה העיקרית הייתה לקבל ענן נקודות ויזואלי באחת מהתצורות האפשריות- תצוגת MATLAB (בעזרת הקוד שהוזכר), תצוגת Python (יוסבר בהמשך), או תצוגת VeloView (ממשק משתמש שניתן על ידי חברת Velodyne- ספק החיישן). המטרה של קבלת הפלט הוויזואלי הייתה בראש ובראשונה אימות של מערכת הייחוס בה הגדרנו את פלט החיישן, וכמו כן אימות הדגימה של המרחק אשר מתקבלת כפלט על כל נקודה בענן.

כלומר, לאחר קבלת ענן הנקודות, בחרנו אובייקט בחדר שנסרק ונמדד פיזית המרחק בינו לבין החיישן באמצעות מד מדידה. השגיאה שהתקבלה הייתה סנטימטר בודד לכל ציר.

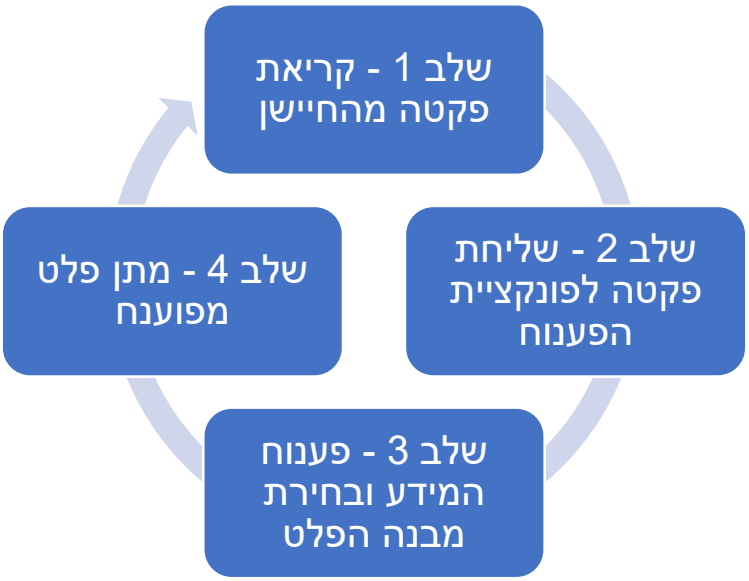


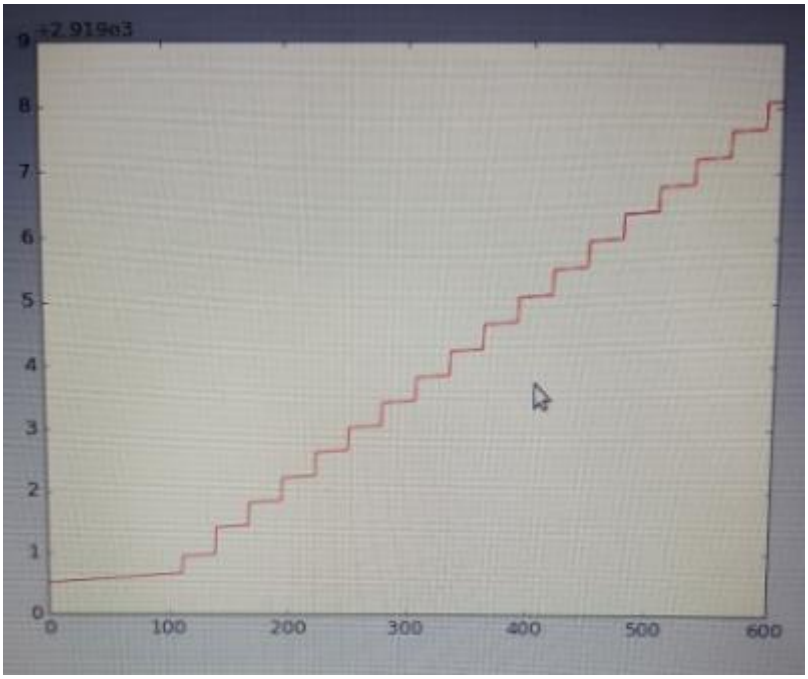
איור 12 - דוגמא לתצוגת ענן נקודות ב-2 תצורות: veloview (מימין) ותצוגת matlab (משמאל)

לאחר תקופה בה בחנו את המערכת לעומק, הגענו למסקנה כי ישנה בעיה בתהליך הפקת הנתונים מהחיישן כאשר השימוש נעשה בקבצי הקוד שהוזכרו בתחילת הפרק, ואשר הוסבו על ידנו על מנת לקבל שליטה על מבנה הנתונים שיופיעו בפלט. הבעיה העיקרית שנתקלנו בה הייתה צוואר הבקבוק בו נתקלנו בשלב פענוח המידע (packet decryption), אשר הוביל למצב בו אנו לא מצליחים להגיע לקריאה ופענוח של 754 פקטות בשנייה (שזהו קצב שידור הנתונים מהחיישן כפי שמופיע בנתוני היצרן).

בבחינה יותר מדוקדקת של הנתונים שקיבלנו, הגענו למסקנה כי מבנה הקוד שנכתב מוביל לאיבוד מידע קריטי בסריקה וזאת מכיוון שמבנה הקוד עבד על פי התרשים הבא-




כלומר, במילים אחרות, מכיוון שקראנו פקטת מידע, הזנו אותה לפונקציית הפענוח, והמתנו לקבל מידע מפוענח, אז למעשה יצרנו "קפיצה/חור" במידע הרציף אותו היינו אמורים לקרוא מהחיישן. התופעה הבלתי רצויה הזו קיבלה אישוש כאשר החלטנו להציג גרפית את מספר הפקטה המפוענחת שהתקבלה כפונקציה של חותמת זמן (שכזכור הצמדנו לכל פקטה בקוד). כפי שניתן לראות באיור הבא, קיבלנו גרף "מדרגות" שמעיד על כך שבכל קפיצה שמתוארת בגרף אנו מאבדים מידע שאמור היה להיות רציף, וזאת מכיוון שמבנה הקוד איננו תקין.




*איור 13* - גרף של מספר פקטה מפוענחת כפונקציה של זמן- מבנה קוד לא תקין

ציר ה-X של הגרף לעיל מייצג את מספר הפריים וציר ה-Y מייצג את חותמת הזמן של הפריים שנקלט בתהליך הפענוח.

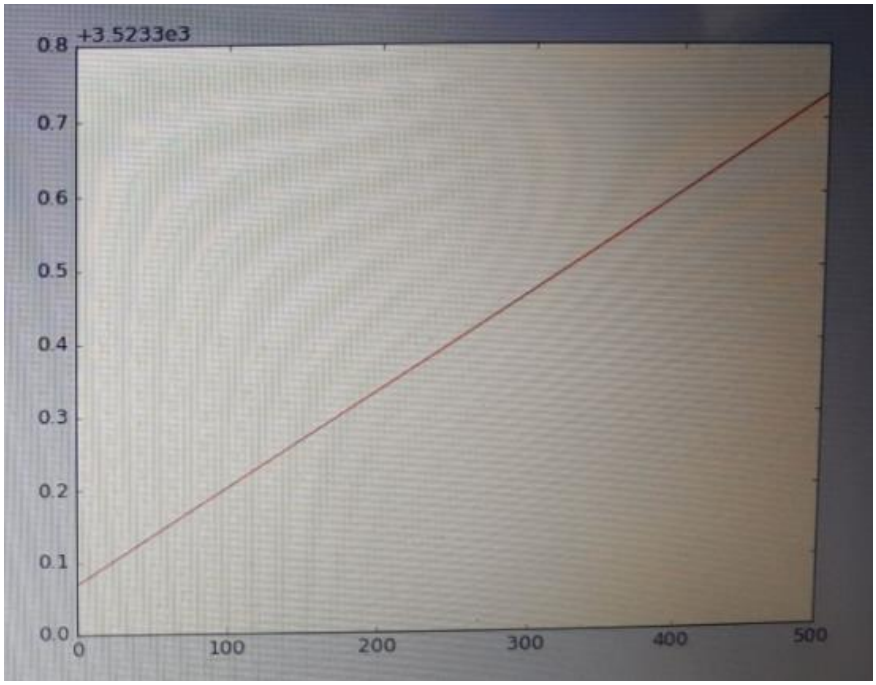
**לאור הבעייתיות במבנה הקוד הקיים, הוחלט לא להשתמש בקוד הקיים וזאת על מנת לכתוב תוכנה מלאה, במבנה תקין יותר, באמצעות שימוש ב- Python 3.6, אשר תאפשר לא לאבד מידע קריטי הנחוץ לפעולה תקינה של המיפוי.**

מבנה התוכנה החדשה שנכתבה על ידנו מורכבת משני תהליכים שרצים במקביל:

- **תהליך 1-** אחראי על קריאת פקטות מהחיישן בקצב המצופה לפי נתוני יצרן, כאשר כל פקטת UDP מקורית שמגיעה מהחיישן נכנסת לתור אינסופי (queue) של פקטות שהוכן בטרם תחילת התקשורת עם החיישן.
- **תהליך 2-** תהליך ראשי. תהליך זה מודע לקיום תור הפקטות ובהתאם לבקשת המשתמש (סריקת פריים $360°$ , או סריקה על פי כמות פקטות ספציפית) הוא מושך פריים המונה 76 פקטות מידע מסוף התור (על מנת לשמור על פעולת "זמן אמת" תקינה) ומתחיל בתהליך הפענוח שלו.

בתצורה כזו, למעשה אנו יכולים כעת להבטיח כי לא נקבל איבוד פקטות, כך שהסריקה לא תצא רציפה – דבר אשר קריטי למטרות הפרויקט בו אנו דורשים להגיע לתוצאה אשר תהווה בסיס לזיהוי אובייקטים ספציפיים בסביבת הרכב האוטונומי, אשר לפיהם יוחלט תהליך הבקרה וניווט הרכב בכל חלק מהמסלול.

לאחר תיקון מבנה הקוד וכתיבתו מחדש בצורה שתוארה, ביצענו הצגה גרפית נוספת שמאששת את הפתרון שהוצע, על מנת להיות בטוחים שכעת הפלט המתקבל מהפקטות המפוענחות מהווה למעשה סריקה רציפה. גם בגרף זה ציר ה-X מייצג את מספר הפריים וציר ה-Y מייצג את חותמת הזמן של הפריים שנקלט בתהליך הפענוח. להלן התוצאות-




איור 14 - גרף של מספר פקטה מפוענחת כפונקציה של זמן- מבנה קוד תקין

# בחירת סביבת עבודה

חלק משמעותי מהתנעת הפרויקט הצריך בחירת סביבת עבודה נוחה לקריאת הדגימות מהחיישן, אשר בסופו של תהליך תאפשר עיבוד נוח והצגתו בצורה ויזואלית מתאימה.

בשלב הראשון התמקדנו בשפת התכנות עמה נעדיף לעבוד:

**MATLAB-** העדפה הראשונה שלנו הייתה להשתמש בשפה שכבר יש לנו עמה ניסיון רב ומאפשרת עיבוד תמונה ברמה גבוהה ובקלות יחסית. במהרה גילנו שעל אף יתרונותיה - נוחות ומספר רב של כלים מוכנים, החיסרון של שפה זו היה בעיקר המהירות איטית של עיבוד נתונים. חיסרון זה הפך אותה לבחירה פחות טובה עקב דרישת הפרויקט ליכולת עיבוד נתונים מהירה, בזמן אמת על מחשב הבקרה של הרכב.

**Python-** בבחינת האפשרות להשתמש בשפה זו, נלקחה בחשבון העובדה כי פרויקטים קודמים בתחום העבודה עם חיישן ה-LiDAR שנעשו בפקולטה, נכתבו כבר בשפה זו.

היתרונות - מהירות גבוהה של עיבוד נתונים ומספר רב של כלים מוכנים, וזאת אל מול החסרונות - חוסר ניסיון משמעותי שלנו בשפה אשר מהווה בפועל מכשול התחלתי בהתנעת הפרויקט. שיקול נוסף שנלקח בחשבון הינה העובדה כי השימוש בשפה זו נהיה מאוד שכיח בתעשייה ולכן רצוי שנשקיע מאמצים המהלך הפרויקט כדי לנצל את האפשרות ללמוד שפה חדשה כחלק ממטרות הפרויקט.

חשוב לציין כי לאחר מספר בדיקות של זמני הרצה שעשינו בפועל עם החיישן היה ברור שהבחירה לעבוד עם Python היה אכן כדאית יותר.

השלב השני היה לבחור מערכת הפעלה עליה נעבוד, Windows או Linux.

מכיוון שפרויקט זה מהווה חלק אינטגרלי מפרויקט הפורמולה האוטונומית, היה לנו חשוב שנוכל לבצע אינטגרציה בקלות יחסית למחשב בקרת הרכב (Nvidia PX2). מערכת ההפעלה של מחשב זה הינה Linux, כך שהחלטנו גם כן לפתח בסביבה זו, וזאת על אף הקושי הקיים באי הכרת מערכת ההפעלה והצורך הטריוויאלי בהקמת סביבת עבודה חדשה בה.

חשוב לציין שהחיסרון המשמעותי בבחירות אלו שביצענו, הן בבחירת שפת התכנות והן בבחירת מערכת ההפעלה, היווה בפועל עקומת הלמידה חדה אשר נמשכה מספר חודשים.

# בחירת שיטה לזיהוי ומיפוי

קיימות כיום שתי שיטות עיקריות ליצירת תמונה תלת ממדית מענן נקודות:

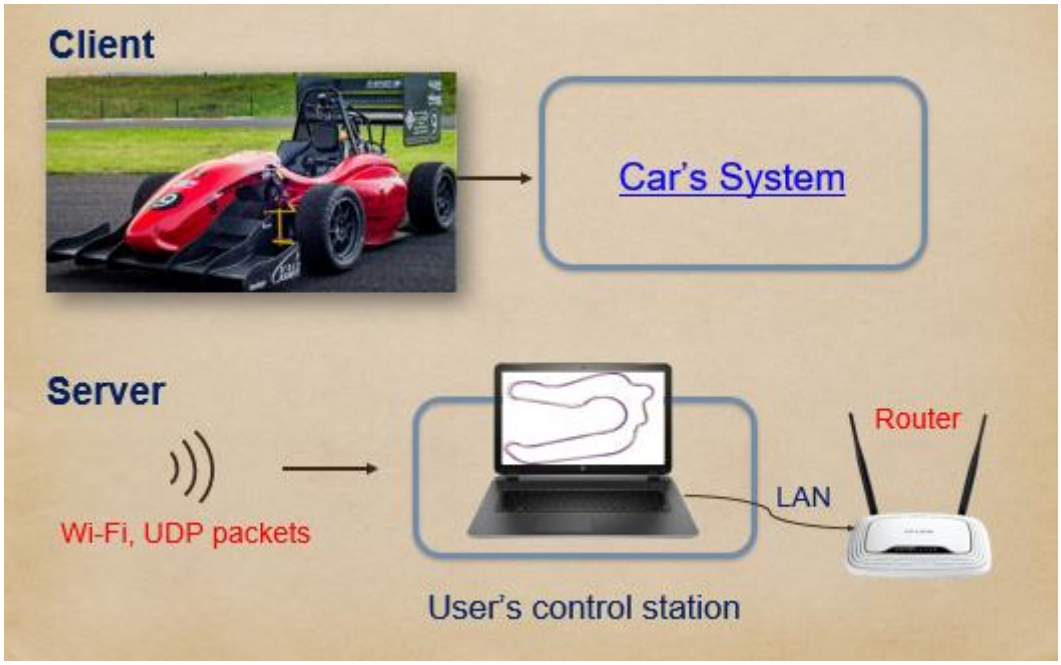
1. בחירת אלגוריתם [2] כלשהו לזיהוי קווים ישרים ועצמים בולטים בכל פריים של מידע והלבשה שלהם אחד על השני כדי לקבל את תמונה כוללת. יתרונות- לא מצריך חומרה נוספת. חסרונות- מצריך פריימים של מידע עם המון פרטים בהם (קווים ישרים, אובייקטים וכדומה) ועיבוד כמות גדולה של נתונים אשר לוקח יותר זמן הרצה.

איור 15 - דוגמא לפלט של אלגוריתם זיהוי קווים ישרים

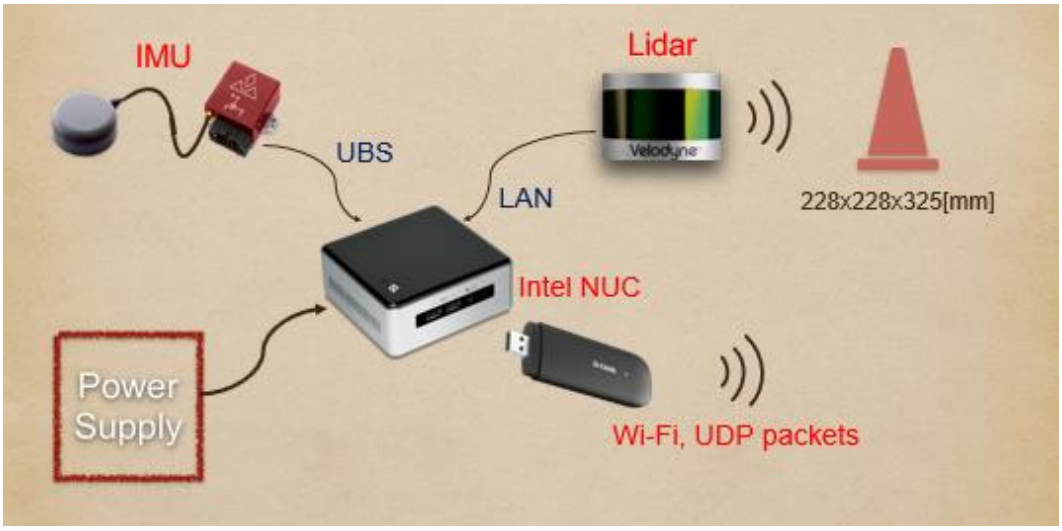
2. שימוש בהתקן סנכרון חיצוני IMU - שימוש ביחידה לזיהוי מיקום עולמית כדי להעביר את המידע בכל פריים לייצוג חד-חד ערכי וכך לקבל תמונה כוללת. יתרונות - עיבוד מאוד מהיר של המידע, לא מצריך הרבה אובייקטים בכל פריים של מידע. חסרונות - ניתן לשימוש רק במקומות בהם יש קליטת GPS (כלומר מחוץ למבנה), בנוסף מצריך איזושהי אינטגרציה בין פלט חיישן ה- LiDAR וה-IMU.

לסיכום, מכיוון שהשימוש באלגוריתם מיפוי על ידי מציאת קווים ישרים מתאים ברוב המקרים לשימוש בעיקר בתוך מבנים סגורים (המאופיינים בכך שכמעט ואין שינויים במרחב הדגימה), אז בפרויקט זה בחרנו לבצע את תהליך המיפוי על ידי שימוש בהתקן חיצוני IMU. כפי שהוזכר, בחירה זו הצריכה למידת התהליך הדרוש לביצוע אינטגרציה בין פלטי שני חיישני ה- LiDAR וה-IMU.

**תרשים מבנה המערכת הצפוי:**




איור 16 - מבנה כללי של המערכת




איור 17 - מבנה מערכת הרכב

בפרויקט זה השתמשנו בחיישן IMU מדגם VectorNav VN-200, אשר היה קיים לרשותנו באדיבות מעבדת CRML בפקולטה להנדסת חשמל. חברת VectorNav מספקת עם החיישן גם API בסביבת Python מה שאפשר עבודה מאוד ידידותית ומהירה להגדרת הפלט הספציפי שהיינו צריכים מהחיישן [3].

# תכן תוכנת המערכת

תוכנת המערכת בנויה למעשה מ- 6 קבצים עיקריים כאשר כל אחד אחראי על חלק אחר בתצורת תוכנת המערכת. להלן פירוט תכן תוכנת המערכת:

1. **Main_t.py-** תהליך עיקרי של כלל המערכת האחראי על יצירת התקשורת עם חיישני ה-LiDAR וה-IMU. תהליך זה אחראי על הפעלת כל המתודות הרלוונטיות לביצוע מטרות הפרויקט, וכולל למעשה בתוכו את הפעולות הבאות-
   - יצירה וניתוק תקשורת עם חיישנים
   - בחירת תצורת עבודה- offline/online
   - בחירת הפעלת תצוגה גרפית למטרות debugging
   - משיכת פריים מחיישן ה-LiDAR וביצוע פענוח למידע
   - הפעלת פעולת זיהוי אובייקטים במרחב וביצוע סינון למידע לא רלוונטי בפריים
   - הפעלת מתודת המרת מערכות צירים (אינטגרציה בין פלט ה-LiDAR לפלט ה-IMU)
   - שליחת פלט האינטגרציה במערכת ייחוס גלובלית אל השרת (תחנת עבודה העומדת בסמוך לרכב) באמצעות פקטות UDP

2. **Lidar.py-** תהליך עיקרי של חיישן ה-LiDAR. מורכב משתי מחלקות אשר אחראיות על קבלת מידע רציף מהחיישן (feeders) כאשר האחת עבור תקשורת Online עם הרכיב, והשנייה עבור תקשורת Offline אשר מבצעת קריאת פקטות מידע מתוך קובץ pickle (אליו נשמר כל המידע בזמן עבודה בתצורת online).
   בנוסף, קיימת מחלקה נוספת בשם vlp16_decoder אשר אחראית על משיכת המידע מאובייקט ה-feeder ופענוחו. ההפרדה בין האובייקטים, נועדה בראש ובראשונה לבצע הפרדה בין תחומי אחריות כך שלמעשה לחלק האחראי על הפענוח כלל לא משנה האם הוא מקבל את המידע מתוך קובץ חיצוני (pickle במקרה שלנו) בתצורת עבודה offline או בזמן אמת מתוך החיישן.
   פענוח פקטות המידע המגיעות מהחיישן בחלק זה, נעשות על סמך פירוט מבנה המידע שהוסבר קודם לכן בספר זה. על מנת לשמור על תצורת עבודה "זמן אמת" חשוב לציין כי כאשר אנו מבצעים קריאת פריים הבא של החיישן, אנו לוקחים למעשה אך ורק את ה-76 פקטות האחרונות אשר הוזנו לתור (queue) המשמש אותנו לתקשורת עם ה-process שקורא נתונים מהחיישן.

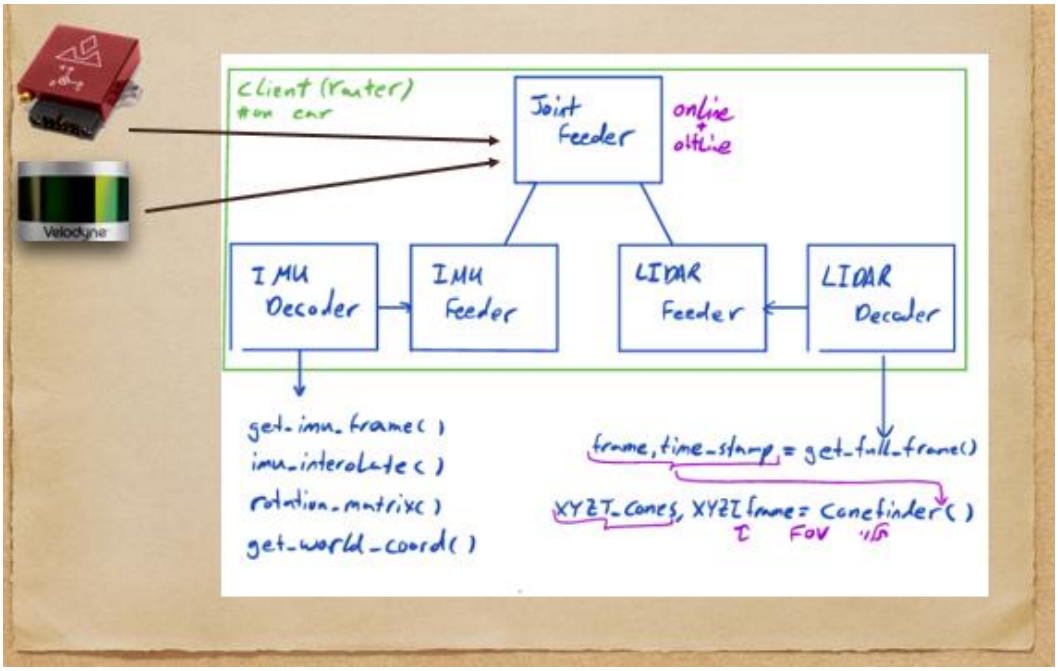
3. **IMU.py-** תהליך עיקרי של חיישן ה-IMU. מורכב משתי מחלקות אשר אחראיות על קבלת מידע רציף מהחיישן (feeders) כאשר האחת עבור תקשורת Online עם הרכיב, והשנייה עבור תקשורת Offline אשר מבצעת קריאת פקטות מידע מתוך קובץ pickle (אליו נשמר כל המידע בזמן עבודה בתצורת online).

בנוסף, קיימת מחלקה נוספת בשם vn200_decoder אשר אחראית על משיכת המידע מאובייקט ה-feeder ופענוחו.

כמו בסקריפט האחראי על תקשורת ה-LiDAR גם כאן ההפרדה בין האובייקטים, נועדה לבצע הפרדה בין תחומי אחריות כך שלמעשה לחלק האחראי על הפענוח כלל לא משנה האם הוא מקבל את המידע מתוך קובץ חיצוני (pickle במקרה שלנו) בתצורת עבודה offline או בזמן אמת מתוך החיישן.

תחומי האחריות העיקריים של אובייקט מסוג vn200_decoder הינם:

- משיכת מידע מ-feeder
- ביצוע אינטרפולציה בין 2 נקודות מידע של החיישן אשר קרובות ביותר מבחינת חותמת זמן לפלט חיישן ה-LiDAR. האינטרפולציה מתבצעת באמצעות רגרסיה לינארית כאשר ההנחה הינה שהשינויים הינם לינאריים בקבועי זמן מאוד קטנים.
- יצירת מטריצות סיבוב בין מערכת ייחוס ה-LiDAR לבין מערכת ייחוס ה-IMU.
- המרת נקודות פלט ה-LiDAR לנקודות במערכת ייוחס גלובלית ע"י התחשבות בנתוני 6dof (yaw, pitch, roll, latitude, longitude, altitude) המתקבלים מחיישן ה-IMU.




איור 18- תיאור ויזואלי לתכן תוכנה

4. **Track_calculations.py-** חלק שאחראי על מציאת אובייקטים חשודים כקונוסים במרחב מנתוני חיישן ה-LiDAR . למעשה בחלק זה קיימת מתודה בודדת אשר נקראת מתוך קובץ Main_t.py. מתודה זו אחראית בין השאר על ביצוע הפעולות הבאות:
   - בחירת לייזרים בודדים (מתוך 16 אפשריים)
   - סינון מרחבי- בחירת זוויות מרחבית רלוונטית לבדיקה (במקרה שלנו 170 מעלות מקדמת הרכב)
   - מציאת "קפיצות" בסריקת הרדיוס המרחבי של כל לייזר – תיוג המידע שנמצא מעל סף threshold מסוים כנקודות חשודות כקונוסים
   - סינון נקודות המתויגות כאדמה (X=0 & Y=0 & Z=0)
   - סינון מרחבי (תצורת מלבן) של החלקים הרלוונטיים לנו בקדמת הרכב (abs(X) <= 2.1 & Y <= 10 & Z <= 0.2 & Z >=-0.4)

5. **Server_Client.py-** חלק שאחראי על יצירת אובייקטי תקשורת בין הרכב (אובייקט מסוג Client) לבין תחנת העבודה של המשתמש שנמצאת בקרבת הרכב (אובייקט מסוג Server), כאשר התקשורת שמועברת בין חלקי המערכת השונים הינה בצורת פקטות מידע (bytes array). למעשה תחנת העבודה מחוברת לנתב ייעודי (router) ויוצרת רשת WIFI ייעודית סגורה אך ורק למטרות המיפוי של מסלול הרכב.
   - **ביצוע clustering:**

בעקבות העובדה שבצד ה- server, כלומר תחנת העבודה, מופעל main של סקריפט זה בלבד אז לחלק זה בתוכנה נוספו גם 2 המחלקות שנבדקו במסגרת פרויקט זה עבור ביצוע קיבוץ הנקודות שמגיעות במערכת ייחוס גלובלית.

1. מחלקת PDF- מיפוי הסתברותי בה אנו למעשה הופכים כל נקודה החשודה כקונוס (אשר התקבלה במערכת הייחוס הגלובלית) לגאוסיאן. לאחר קבלת מספר פריימים ניתן לקבוע סף threshold מסוים על פיו נוכל לקבץ את קבוצות הנקודות על פי הסבירות שלהן להיות אכן אובייקט מרחבי שהינו קונוס במפה.
2. מחלקת kmeans_cluster- מיפוי באמצעות מציאת K מרכזי מסה של כלל הנקודות המופיעות בפריים ספציפי. כלומר על סמך ידע מוקדם (לדוגמא מהירות ומרחק שרכב עובר במס' פריימים מסוימים של חיישן ה- LiDAR) ניתן לקבוע את הכמות הצפויה של מס' קונוסים בפריים – כלומר לקבוע את K.

חלק נוסף שמתבצע תחת סקריפט זה הינו בניית תצוגת המשתמש על מנת שבתחנת העבודה יהיה ניתן לראות בזמן אמת את המיפוי הנבנה לאור התקדמות של הרכב במסלול.

6. **utilities.py-** סקריפט זה אחראי על מתודות עזר המתלוות לכלל הפרויקט הזה, על מנת לאפשר פעולות debugging בצורה ויזואלית נוחה יותר. הסקריפט כולל את המתודות הבאות:

- print_progress – מאפשרת להביט בצורה ויזואלית בסרגל התקדמות של קריאה/כתיבה.
- set_aspect_equal_3d – מבצעת יישור ייחוס של גרפים תלת ממדיים, כך שכל ציר יהיה בייחוס מתאים לצירים האחרים.
- plot_2d ו- plot_3d – מאפשרת תצוגה גרפית דו-ממדית ותלת ממדית של גרף מסוג scatter.
- createMovie – אפשרות להפקת סרטון קצר מכלל התמונות שנשמרו בתיקייה כלשהי קודם לכן על ידי המפעיל. אנו השתמשנו במתודה זו בעיקר עבור צרכי debugging לנוחות צפייה רציפה בניסויים שביצענו.
- plot_telemetry - תצוגת המשתמש על מנת שבתחנת העבודה יהיה ניתן לראות בזמן אמת את המיפוי הנבנה לאור התקדמות של הרכב במסלול. במסך טלמטריה זה ניתן לראות בצדו הימין של המסך את המיפוי במתבצע סביב הרכב בזמן אמת, כאשר המסלול הכחול האמצעי הוא למעשה מסלול הרכב שנלקח באמצעות נתוני מיקום מה-IMU. בצידו השמאלי של המסך ניתן לראות את נתוני ה-Yaw, Pitch, Roll שגם כן נלקחים בהתאמה מאותו פריים מתאים של ה-IMU.

- נציין כי הסיבה לביצוע הקלטת המידע הגולמי המתקבל מהחיישנים השונים במצב עבודה Online נועד על מנת לשמור את המידע הגולמי (במהלך נסיעה על רכב הפורמולה / עגלת ניסיון) וזאת בכדי שנוכל לאחר מכן לעבד את הנתונים המתקבלים מהחיישן ללא כל תלות בהפעלתו בשנית.

7. **steering.py-** קובץ נוסף שנוסה במהלך עבודה משותפת עם צוות הפורמולה האוטונומי של שנת 2018. בקובץ זה ניתן למצוא את האפשרות לקבל את זווית ההגה הרציפה שלמעשה ניתן לשלוח לבקר החשמלי שאחראי על שליחת פקודות להיגוי הרכב.

למעשה אנו השתמשנו בקובץ זה על מנת לבקר את נסיעת הרכב בניסויים שבוצעו בפרויקט. כלומר, הרכב קיבל פקודות היגוי ישירות מהתוכנה שלנו על מנת לבצע נסיעה חלקה בין הקונוסים.

השיטה על סמך אנו מוצאים בכך פריים את זוויות ההיגוי הבאה, הינה למעשה על סמך מיפוי ה-PDF שהוזכר קודם לכן, כאשר אנו מחפשים בכל פריים את הווקטור שנמצא ב-margin מקסימלי מכל הרי הגאוסיאנים (שכל אחד מהם מתואר על ידי קונוס שזוהה במרחב). כך למעשה נוצר קו בזווית מסוימת אל עבר האופק, שנבחר להיות כמה מטרים מקדמת הרכב. אותה הזויות ניתנת לחישוב פשוט וממנה מיוצרת פקודת ההגה הבאה לבקר החשמלי.

במסגרת פרויקט זה חשוב להדגיש כי לא שמנו דגש על אופטימיזציה של קוד זה, אך למרות זאת ראינו לנכון לצרפו לטובת המשך פעילות עתידית בתחום.

# מבנה קוד ה-LiDAR

להלן תיאור המחלקות והמתודות השונות אשר אחראיות על פעולות התקשורת וביצוע פענוח המידע המגיע מחיישן ה- LiDAR:

- **vlp16_online_feeder-** מחלקה האחראית על יצירת תקשורת עם החיישן בזמן אמת. בשלב האתחול של אובייקט מסוג זה, אנו יוצרים תור אינסופי (queue) שאליו מוזנות פקטות המידע הגולמיות, ובנוסף מאתחלים תהליך מקבילי אשר מתחבר לחיישן ובלולאה אינסופית מקבל ממנו פקטות ושם אותן בתור התור המיועד. מידע המוכנס לתוך התור הוא למעשה tuple המורכב משני משתנים- פקטת המידע הגולמית וחותמת זמן.

  <u>מתודות במחלקה זו:</u>

  - get_frame: מאפשרת בהמשך לאובייקט הפענוח (decoder) למשוך מידע מתוך תור הפריימים הגולמיים.
  - run: מתודת הפעלת התהליך המקבילי שמקבל בלולאה אינסופית מידע מהחיישן. מתודה זו מתחילה את פעולתה כבר בשלב אתחול האובייקט. המתודה מחזיקה מנעול (Mutex) לשמירה על פעולה תקינה של כתיבה/קריאה מקבילית מאותו משאב משותף (queue).
  - recv_worker: מתודת התהליך המקבילי אשר מופעלת מתוך מתודת run. אחראית על יצירת תקשורת עם החיישן והכנסת המידע הגולמי לתוך התור. בעת זיהוי בקשה לניתוק התקשורת, בעזרת דריסת ניהול אות הבקרה SIGTERM, המתודה מבצעת כתיבה של כלל המידע שהוכנס לתור אל קובץ Pickle.
  - close_socket: ניתוק התהליך המקבילי שלמעשה שולח אליו את האות בקרה SIGTERM.

- **vlp16_offline_feeder-** מחלקה האחראית על קבלת פלט החיישן מתוך קובץ Pickle שהוקלט קודם לכן. בשלב האתחול של אובייקט מסוג זה, אנו למעשה קוראים את כל הקובץ ומחלקים אותו לפריימים של 76 פקטות (שהינו ערך של סיבוב 360 מעלות).

  <u>מתודות במחלקה זו:</u>

  - get_frame: מאפשרת בהמשך לאובייקט הפענוח (decoder) למשוך מידע מתוך תור הפריימים הגולמיים.

- **vlp16_decoder-** מחלקה האחראית על תהליך הפענוח של המידע הגולמי שמתקבל ב-feeder. בשלב האתחול של אובייקט מסוג זה אנו למעשה יוצרים את הפורמט (מבחינה מבנית) של פקטה בודדת שנקבל מה-feeder, וזאת בהתאם לכל מה שהוסבר קודם לכן בספר זה על מבנה פקטת LiDAR.

  <u>מתודות במחלקה זו:</u>

  - interpolate_az: ביצוע אינטרפולציה של זוויות מרחביות החסרות בכל פקטה. מכיוון שישנם 24 בלוקים של מידע בכל פקטה, אך רק 12 נתוני זווית המתקבלים מהחיישן (עבור כל זווית אי-זוגית) אנו צריכים לבצע "השלמה" לכל הזוויות הזוגיות, ע"י ביצוע חישוב ממוצע של כל 2 זוויות אי-זוגיות קרובות.
  - decode_packet: מתודה מרכזית בה למעשה נעשה כל שלב הפענוח של מידע גולמי מתצורה של בתים ל-string. לאחר מכן ביצוע המרה קרטזית בעזרת כלל הנתונים הפולריים שהתקבלו במידע הגולמי.
  - get_full_frame: משיכת פריים מלא (360 מעלות) של החיישן באמצעות קריאה למתודה מתאימה של אובייקט ה-feeder אשר הוזן בעת בניית אובייקט ה-decoder. נשים לב כי אנו למעשה לוקחים את ה-76 פקטות האחרונות בכל עת על מנת לשמור על רציפות עבודה של תצורת real-time.
  - decode_N_packets: אפשרות לעבודה בתצורה של פענוח מספר ספציפי של פקטות מהחיישן. כרגע לא נמצא בשימוש בפרויקט זה.
  - plot_lidar_cones: אפשרות לתצוגה ויזואלית של הענן הנקודות המקורי שהתקבל מהחיישן כאשר על תצוגה זו נוספות נקודות בצבע אדום החשודות כקונוסים במרחב. כמו כן, במתודה זו ישנה האפשרות לבצע שמירה של כל תמונת פריים עם קונוסים לתיקייה נפרדת.

# מבנה קוד ה- IMU

להלן תיאור המחלקות והמתודות השונות אשר אחראיות על פעולות התקשורת וביצוע פענוח המידע המגיע מחיישן ה- IMU:

- **vn200_online_feeder -** מחלקה האחראית על יצירת תקשורת עם החיישן בזמן אמת. בשלב האתחול של אובייקט מסוג זה, אנו יוצרים תור אינסופי (queue) שאליו מוזנות פקטות המידע הגולמיות, ובנוסף מאתחלים תהליך מקבילי אשר מתחבר לחיישן ובלולאה אינסופית מקבל ממנו פקטות ושם אותן בתוך התור. בסיום הריצה כל התור נשמר לקובץ מסוג pickle, כך שכל המידע מהריצה הנוכחית נשמר.

  <u>מתודות במחלקה זו:</u>

  - **receiver:** מתודת התהליך המקבילי אשר מופעלת מתוך מתודת run. אחראית על יצירת תקשורת עם החיישן והכנסת המידע הגולמי לתוך התור. בעת זיהוי בקשה לניתוק התקשורת, בעזרת דריסת ניהול אות הבקרה SIGTERM, המתודה מבצעת כתיבה של כלל המידע שהוכנס לתור אל קובץ Pickle.
  - get_packet: מאפשרת בהמשך לאובייקט הפענוח (decoder) למשוך מידע מתוך תור הפריימים הגולמיים.
  - close_imu: ניתוק התהליך המקבילי שלמעשה שולח אליו את האות בקרה SIGTERM.

- **vn200_offline_feeder -** מחלקה האחראית על קבלת פלט החיישן מתוך קובץ Pickle שהוקלט קודם לכן. כבר בשלב טעינת הקובץ המחלקה מייצרת מערך עם כל המידע, המאפשר בהמשך לקרוא למידע שורה-שורה כמו שהוא היה מתקבל בריצה בזמן אמת.

  <u>מתודות במחלקה זו:</u>

  - get_packet: מאפשרת בהמשך לאובייקט הפענוח (decoder) למשוך מידע מתוך המערך בדומה למשיכה מהטור בריצה בזמן אמת.

- **vn200_decoder -** מחלקה האחראית על העברת הנקודות החשודות כקונוסים במערכת הצירים של ה- LiDAR לייצוג חד-חד ערכי במערכת הצירים של העולם. המעבר בין מערכות הייחוס מתבצע ע״י שימוש במידע הגולמי מה-IMU ( yaw, pitch, roll) וה-GPS (Longitude, Latitude). בעת אתחול אובייקט המחלקה, אנו למעשה שומרים את הנתונים הראשונים שנקראים מה- IMU כדי לחסר אותם מכל שאר הערכים הבאים ובכך ״לאפס״ את המערכת (זוויות ומרחק). בנוסף מחושבים ערכי ההמרה ממעלות למטרים כתלות במיקום שלנו בכדור הארץ.

<u>מתודות במחלקה זו:</u>

  - get_imu_frame: מחזיר 2 פקטות מידע של ה-IMU שהכי קרובות לזמן אמצע ה -frame של נקודות שהגיעו מזיהוי הקונוסים.
  - imu_interpolate: מייצר משוואת ישר לכל פרטמר בפקטת ה-IMU על בסיס 2 הפקטות שהתקבלו מהמתודה get_imu_frame. לאחר מכן, מציב את כל הזמנים של הנקודות בפריים של ה-LiDAR, כך שנקבל ערכי IMU מדויקים יותר(תחת ההנחה שבפרק זמן קצר השינויים לינאריים).
  - rotation_matrix: מחשבת את המטריצות סיבוב הנחוצות לתיקון הנקודות בפריים של ה-LiDAR. מטעמי מהירות ריצה של התוכנה, למעשה בחרנו לקחת את ההנחה כי התיקון לכל פריים יהיה מאד דומה לתיקון מערכי ה-IMU וזאת לפי הזמן של מרכז הפריים.
  - get_world_coords: אחראית על ביצוע המרת מערכות הייחוס כולה. למעשה מתודה זו קוראת לכל שאר המתודות במחלקה בסדר הנכון ומחזירה את הנקודות החשודות כקונוסים (שהתקבלו תחילה במערכת ייחוס של ה-LiDAR) במערכת צירים של העולם.

# מעבר מערכות ייחוס LiDAR - גלובלי

<u>ביצוע הסנכרון בין ה- LiDAR וה- IMU:</u>

לאור ההחלטה לבצע את שיטת המיפוי על פי הסנכרון בין שני החיישנים, עלינו לבצע איזושהי טרנספורמציה שמבצעת מעבר ממערכת הייחוס של חיישן ה- LiDAR אל מערכת הייחוס הגלובלית, כך שלמעשה נצליח למנוע היווצרותן של רפליקות של אותה נקודה על גבי תצוגת המיפוי. במילים אחרות, המטרה העיקרית של שימוש בשני חיישנים אלו על מנת ליצור מיפוי מדויק של המרחב, הינה ליצור זהות חד-חד ערכית במרחב לכל נקודה אשר תדגם על ידי חיישן המרחק.

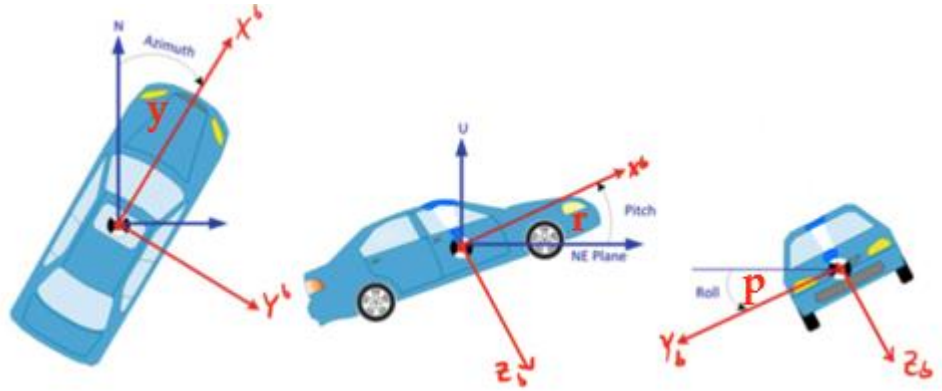
אופי החד-חד ערכיות של כל נקודה במרחב, מונעת מצב בו אחד הלייזרים דוגם קונוס במרחב ומוציא פלט כלשהו, בעוד שהרכב ממשיך להתקדם כך שאותו הלייזר דוגם שוב את אותה נקודת הקונוס ומוציא פלט אחר – כך שלא ניתן לדעת שזוהי למעשה אותה נקודה במרחב.

בפועל, הדרך לבצע המרה בין מערכות ייחוס מתבצעת על ידי חישוב מטריצות הסיבוב הרלוונטיות לכל המרה, כאשר אנו כעת נציג א המעבר שביצענו ממערכת יחוס של ה-LiDAR למערכת ייחוס של מרכז כדור הארץ המאפשרת את אותו ייצוג חד-חד ערכי של כל נקודה במרחב [4].

<u>חישוב מטריצות הסיבוב:</u>

א. בשלב ראשון בצענו המרה ממערכת ייחוס של חיישן ה- LiDAR אל מערכת ייחוס של חיישן ה- IMU וזאת מכיוון שהצירים שלהם לא מסונכרנים. בנוסף ישנו מרחק קבוע שיהיה בין שני מיקומי החיישנים על הרכב (Position offset) ולכן גם הוא יילקח בעתיד בחשבון. מטריצה הסיבוב בה השתמשנו בכדי לתקן את ייחוס הצירים הינה המעבר הבא: [x,y,z] ⟸ [y,x,-z]

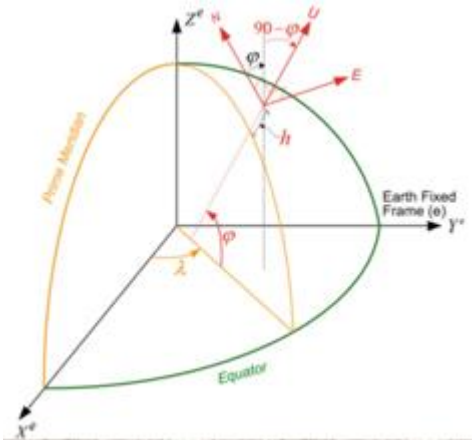
ב. בשלב הבא, ביצענו מעבר ממערכת הייחוס של הרכב (body frame) למערכת הייחוס של הלוקלית של כדור הארץ ( LLF - Local Level Frame), וזאת על ידי שימוש במטריצת ההמרה הבאה:




$$R_b^l = \begin{bmatrix} \cos y & -\sin y & 0 \\ \sin y & \cos y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos p & -\sin p \\ 0 & \sin p & \cos p \end{bmatrix} \begin{bmatrix} \cos r & 0 & \sin r \\ 0 & 1 & 0 \\ -\sin r & 0 & \cos r \end{bmatrix}$$

איור 19 - מטריצת מעבר- Body to LLF

ג. בשלב האחרון ביצענו מעבר ממערכת ייחוס הלוקלית (LLF) אל מערכת הייחוס הגלובלית שהינה ביחס למרכז כדור הארץ ( ECEF- Earth Centered Earth Fixed) , כאשר מעבר זה בוצע על פי החישוב הבא:

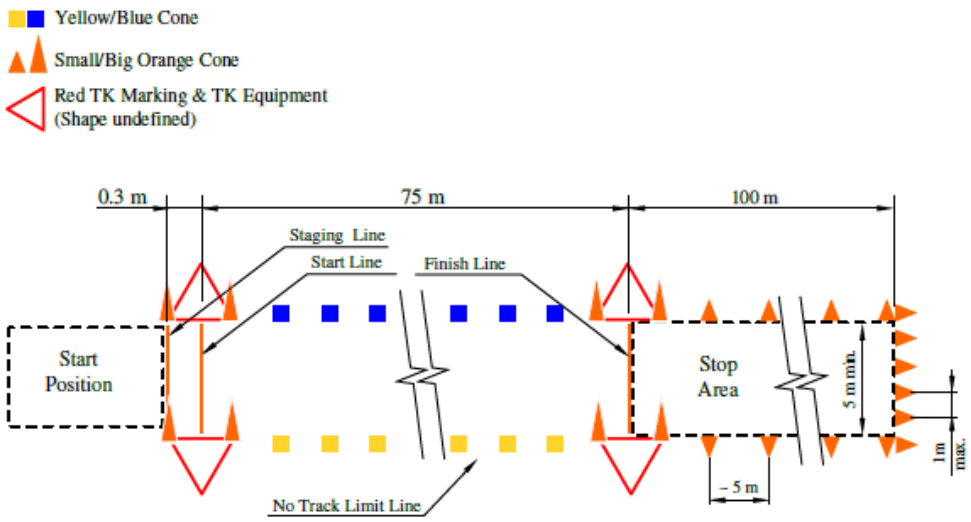



$$R_l^e = \begin{bmatrix} \cos(-\lambda - 90) & \sin(-\lambda - 90) & 0 \\ -\sin(-\lambda - 90) & \cos(-\lambda - 90) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi - 90) & \sin(\varphi - 90) \\ 0 & -\sin(\varphi - 90) & \cos(\varphi - 90) \end{bmatrix}$$

איור 20 - מטריצת מעבר - LLF to ECEF

# תהליך זיהוי קונוסים

תהליך הזיהוי של אובייקטים ספציפיים בסביבה הצריך בראש ובראשונה להבין לעומק מהם המאפיינים של סביבת התחרות. הנתונים הידועים לנו מראש הינם נתוני גדלי הקונוסים אשר אוזכרו קודם לכן, וכמו כן ישנם קווים מנחים אשר כתובים בחוקי התחרות לגבי אופן הנחתם של הקונוסים על המסלול ביחס אחד לשני. מנתונים אלו ניתן ללמוד כי לכל היותר יפרידו בין קונוס לקונוס 5 מטרים (מצוין שלקראת סיבוב ייתכן שמרחק זה אף יתקצר כדי להקל על חיישני המערכת) ובנוסף כי רוחב המסלול יהיה בערך כ-5 מטרים. בנוסף, צידו הימני של המסלול יסומן בקונוסים צהובים וצדו השמאלי יסומן בכחולים (קונוסים מיוחדים של תחילת וסוף מסלול הינם בגדלים מעט שונים ובצבע כתום).




איור 21 - תיאור של מקצה דינמי להבנת אופן הנחת הקונוסים במסלול

## בחירת לייזרים

כפי שהוסבר בתחילה, שמנו דגש על תכונת הרזולוציה בציר האזימוט בבחירת החיישן המתאים לביצוע אתגר זיהוי האובייקטים. בחישוב שביצענו הגענו למסקנה שלאור אופי המסלול המוטורי הצפוי ומרחקי הקונוסים, אנו נאלץ להשתמש במידע חלקי בלבד מתוך כלל המידע אותו מציע חיישן ה- VLP-16.

למעשה, לאור חישובי קצב התקדמות הרכב שבוצעו קודם לכן וכמות הנקודות שנופלות על כל קונוס בהתאם למרחק שלו מהחיישן, הגענו למסקנה כי הנתונים העיקריים שיהיו משמעותיים עבור תהליך הזיהוי יגיעו מהלייזרים הממוקמים בזוויות האנכיות (Elevation degrees) הבאות: 3 ,1 , -1 , -3 .

ארבעת הלייזרים הללו ייקחו חלק בזיהוי קונוסים, כאשר מכיוון שקצב הדגימות לשנייה של לייזר בודד הינו $\frac{300,000}{16} = 18,750$ , אנו מצפים לקבל כמות של כ- 75,000 נקודות דגימה בשנייה (אשר רלוונטיות לצרכינו).

<u>יתרון עיקרי-</u> כמות מידע יותר גדולה לשנייה אשר תאפשר לאשרר באופן ודאי יותר את זיהוי הקונוס. בנוסף, בלקיחת ארבעת הלייזרים שהוזכרו אנו למעשה מבטיחים שבאינטרוול זמן דגימה קצר מאוד (בו הרכב מתקדם במהירות גבוהה ועובר מרחק קצר כלשהו) כי ידגמו לפחות 3 קונוסים בקדמת הרכב מכל צד (כלומר רזולוציית דגימה של עד כ- 15 מטר).

<u>חסרון עיקרי-</u> כמות המידע גדולה ולכן תהליך הפענוח והזיהוי של הקונוסים שצריך להתבצע בזמן אמת, צריך להיות יעיל מאוד. כלומר, דרוש קוד שיעמוד בזמן חישוב יעיל ככל הניתן לאור מורכבות המטרה.

## <u>תהליך זיהוי אובייקטים</u>

לאחר הפרקים הקודמים בהם הבנו במדויק מהם הפלטים העומדים לרשותנו מחיישן המרחק, ובנוסף כיצד נראה האובייקט (קונוס) אותו אנו רוצים לזהות כענן נקודות במרחב הדגימה של החיישן (בדיקת ההיתכנות שנעשתה), התכנסנו למעשה לחשיבה כיצד נוכל לשלב את כל הנתונים הללו למציאת פתרון לזיהוי האובייקט.

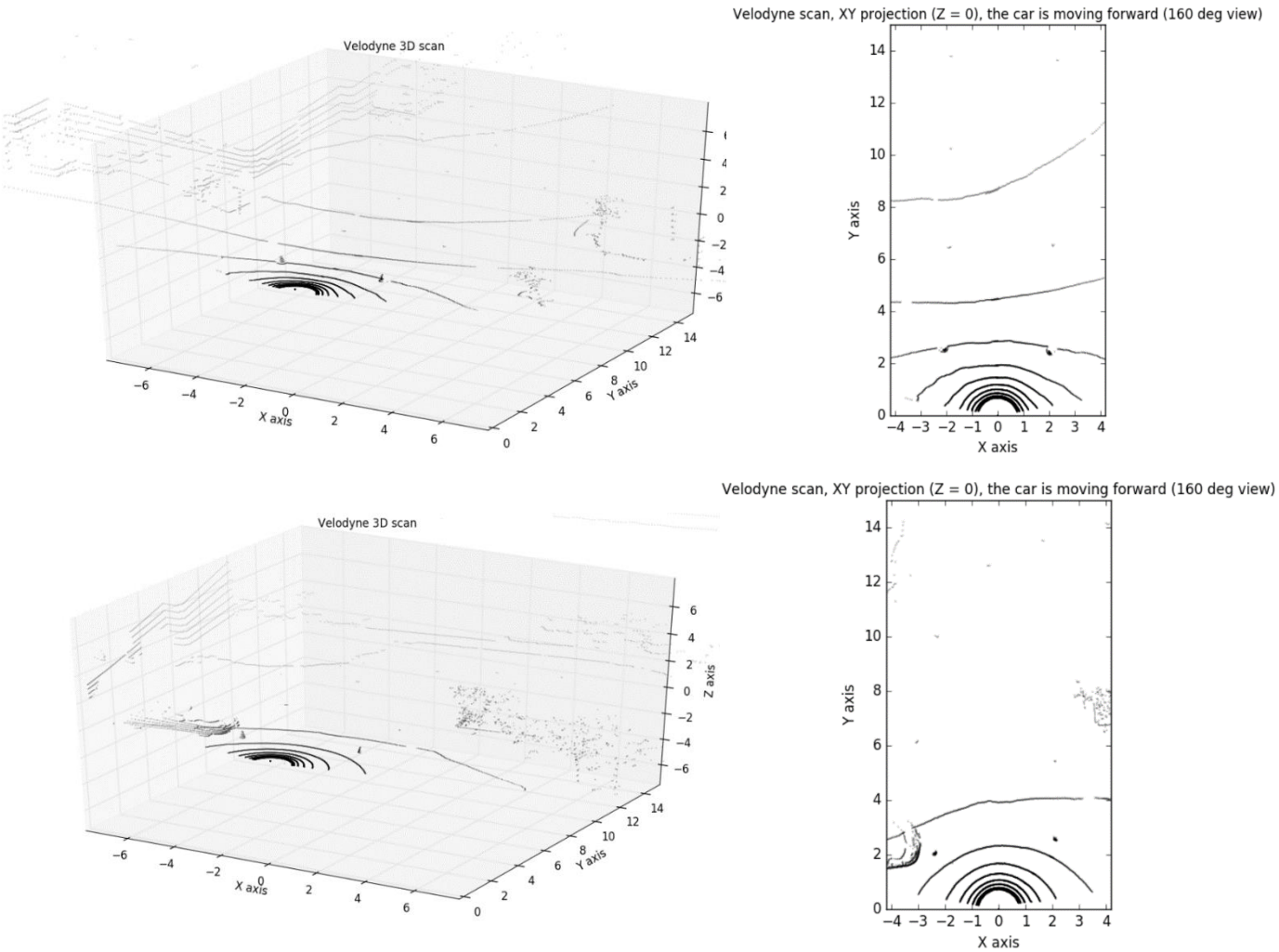
שלב זה כלל הרבה ניסויים בהם חיברנו את החיישן לקדמת רכב הפורמולה בגובה של כ- 15 ס"מ מהכביש, או לעגלת ניסויים המדמה את הרכב, והקלטנו דגימת LiDAR של "מסלול" מדומה שכלל את הקונוסים הספציפיים שיהיו בתחרות בחניית הרכבים של הפקולטה למכונות. הקלטות אלו, תרמו רבות לניתוח הבעיה של הזיהוי ולהבנה יותר ברורה מה מייחד קונוס מהסביבה שלו, וזאת על מנת שנוכל לתייג אותו כנקודה בעלת עניין. מסקנה שהגענו אליה הייתה שבמהלך סריקה מרחבית $360°$ של <u>לייזר ספציפי</u> (כלומר בזווית אנכית ספציפית), ענן הנקודות שמתקבל מהלייזר מורכב למעשה מדגימה רציפה של הכביש עד לפגיעה בקונוס (במידה ואכן הקונוס נמצא בטווח "הראייה" של הלייזר) ולאחר מכן ישנה חזרה לדגימה רציפה של הכביש.

מסקנה זו הובילה להבנה שנוכל להשתמש בעובדה שמצופה קפיצה חדה בערכי הפלט של החיישן- רדיוס ורמת הרפלקטיביות, וזאת בכל הנוגע להבדלים הקיימים בין דגימת הכביש הרציפה לבין דגימת הקונוס. כלומר במילים אחרות, כאשר לייזר ספציפי סורק את המרחב הוא צפוי לתת דגימה רציפה של מרחק קבוע עד להתקלות באובייקט כלשהו, וזאת כמובן בהנחה כי המרחב "נקי" – בדומה לסביבת התחרות, אשר צפויה לספק בענן הנקודות שידגם בה אך ורק קונוסים וכביש.

כפי שהוסבר בפרקים הקודמים, על מנת לייצר סביבת בדיקות נוחה לניסויים הללו, השתמשנו ביכולת התצוגה של Python, כאשר סיפקנו למשתמש לבחור מבין 2 אפשרויות תצוגה-

1. תצוגה תלת ממדית של מרחב הדגימה של החיישן, גזרת זווית מרחבית $80° - 280°$.
2. תצוגה דו ממדית ממבט עילי (רק צירי XY), גזרת זווית מרחבית $80° - 280°$.

בתוצאות הניסוי שיובאו לפניכם בספר זה, למעשה בדקנו את אופי הדגימה של שתי צורות נסיעה שונות- ישרה ופנייה ימינה, כאשר הניסוי כולו התבצע בחניית הפקולטה להנדסת מכונות, והקונוסים הונחו בצורה המתאימה לכללי התחרות (אשר הוסברו קודם לכן).
באיור הבא ניתן להתרשם מפלט רגיל (ענן נקודות מלא ללא סינון) של החיישן כפי שנראות בתצוגות השונות שאנו מאפשרים למשתמש להפעיל על סביבת העבודה ב- Python:




איור 22 - דוגמה לפלט ענן נקודות: נסיעה ישרה (שורה עליונה), פנייה ימינה (שורה תחתונה)

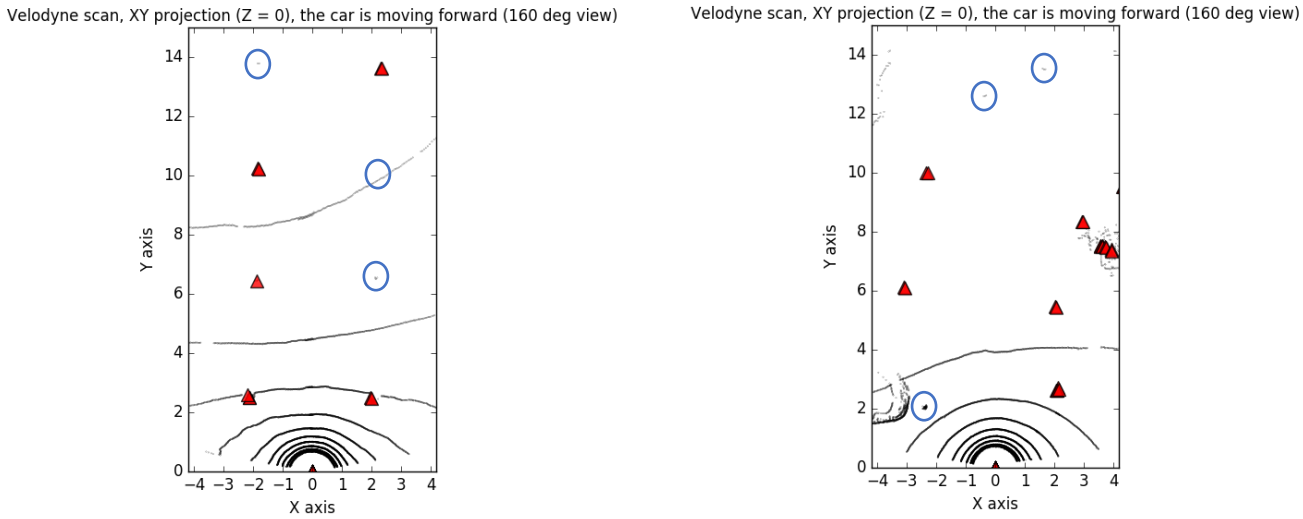
## תוצאות הניסוי

בחינת המידע המפוענח שהגיע מענן הנקודות שסרקנו באמצעות החיישן, הוביל לתוצאה שציפינו לה אשר הוסברה קודם לכן, ואכן קיבלנו קפיצות חדות בנתוני הרדיוס כאשר הדגימה "עברה" מתיאור הסביבה (לרוב כביש) לתיאור קונוס. עם זאת, ההחלטה מהו ערך הקפיצה (האמפליטודה) אשר תהווה חסם מספיק טוב לזיהוי ודאי של קונוס מאובייקטים אחרים בסביבה נותרה בעייתית.

על מנת להגיע למסקנות יותר ברורות לגבי הערכים של ההבדל ברדיוסים וברמת הרפלקטיביות הנחוצים לזיהוי של קונוס, היינו צריכים להפעיל רמות סינון שונות ולבחון את התוצאות. הדרך בה נקטנו היא בחינת רמות הקפיצה (ההבדלים) בסריקה של כל לייזר שנבחר לחוד. במילים אחרות, מבין ארבעת הלייזרים שבחרנו לעבור איתם, עברנו על כל לייזר בנפרד ושמרנו את ההבדל (פונקציית diff) בין כל שני ערכי פלט שקיבלנו מהלייזר בסריקה (עבור וקטור ערכי הרדיוס ועבור וקטור ערכי הרפלקטיביות).

חשוב לציין כי הערכים שנבחרו כחסם נומרי כלשהו, הגיעו למעשה מקריאת הנתונים הנומריים של ערכי הקפיצה כאשר זיהינו בוודאות קונוס, וכך למדו על החסמים הטובים ביותר שיובילו לזיהוי של כלל הקונוסים בתמונת הדגימה. להלן התוצאות והמסקנות אליהן הגענו לאחר כל רמת סינון שונה, כאשר כל נקודה החשודה כקונוס מסומנת במשולש אדום:

א. תיוג קונוס מוגדר ככל קפיצה בוקטור הרדיוס אשר קטנה מ- (-5) מטרים.



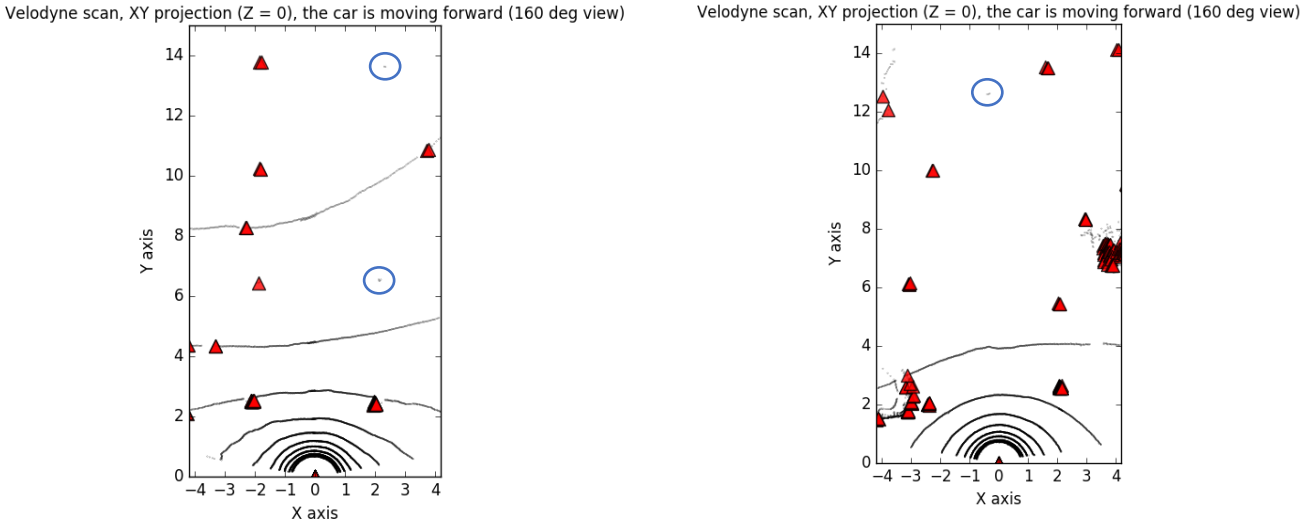

איור 23 - תצוגת זיהוי קונוסים $R < -5m$

כפי שניתן להתרשם מהתוצאות, הגדרת נקודה חשודה קונוס לפי ההבדל ברדיוס, אכן מביאה לתוצאות סבירות בהן ניתן לראות באמת נקודות שתויגו כקונוס שאכן הינן קונוסים במציאות. עם זאת, ישנן נקודות שעדיין לא תויגו והן אכן קונוסים (לנוחות מוקפות בתצוגה המקורית בעיגולים כחולים), לדוגמא בנסיעה הישרה (תמונה שמאלית) ישנם שלושה קונוסים שלא תויגו, ובפנייה ימינה (תמונה ימנית) ישנם שלושה קונוסים שלא תויגו.

לתוצאות אלו, התווסף גם תיוג לא נכון של מדרכה שהייתה בסביבת הקונוסים בצדו הימני של המסלול (בתמונה הימנית של פנייה ימינה)- מתיוג זה אנו מנסים להימנע.

לאור תוצאות אלו הוחלט לבדוק את תוצאות וקטור הרפלקטיביות לבדו, כאשר הוחלט על ערך קפיצה מינימלי של 10 מכיוון שזהו הערך שנצפה בנתונים חוזר על עצמו לפני דגימת קונוס.

ב. תיוג קונוס מוגדר ככל קפיצה בוקטור הרפלקטיביות אשר גדול שווה מ- 10.

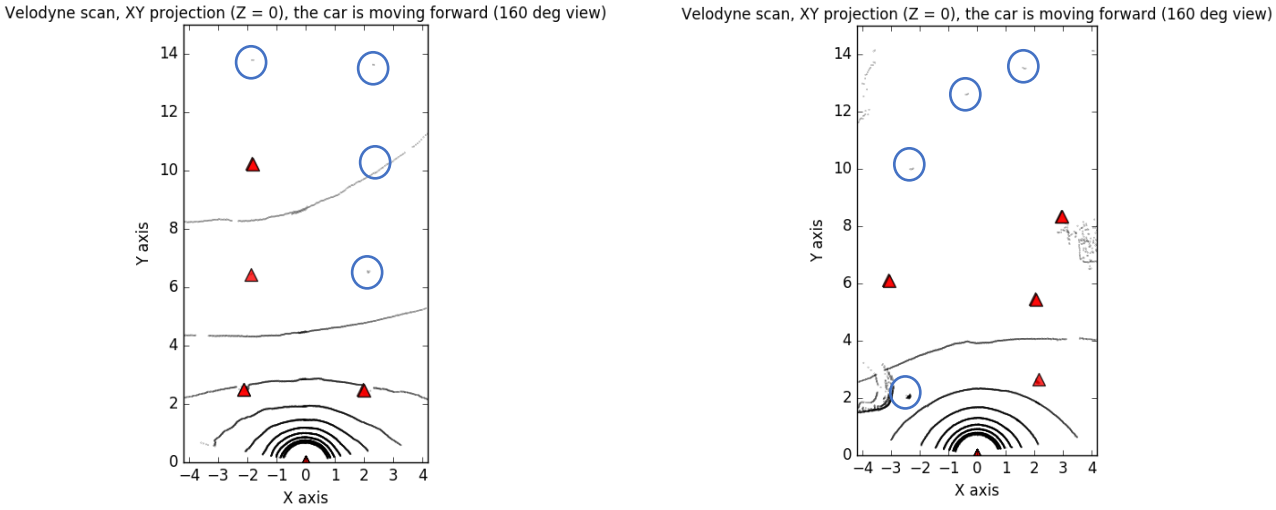



איור 24 - תצוגת זיהוי קונוסים $REF \geq 10$

כפי שניתן להתרשם מהתוצאות, הגדרת נקודה חשודה קונוס לפי ההבדל ברפלקטיביות, נותנת תוצאה הרבה יותר "רועשת", כאשר הרבה גורמים סביבתיים מתויגים כקונוס למרות שאינם כאלו. בפנייה ימינה (תמונה ימנית) ניתן לראות כי שרכב שחנה בחנייה זוהה כקונוס, ושוב המדרכה זוהתה כקונוס. בנסיעה הישרה (תמונה שמאלית) בשני מקרים שונים דגימת הכביש זוהתה כקונוס באופן שגוי. בנוסף, עדיין ישנן נקודות שלא תויגו והן אכן קונוסים (לנוחות מוקפות בתצוגה המקורית בעיגולים כחולים).

לאור תוצאות אלו, הוחלט לנסות לאחד בין שני הפילטרים שהוצגו עד כה ולבדוק האם חיתוך התנאים (יצירת תנאי תיוג חדש שאומר שרק נקודה שעומדת בשני התנאים תתויג) יספק תוצאה יותר טובה.

ג. תיוג קונוס מוגדר ככל קפיצה בוקטור הרפלקטיביות אשר גדול שווה מ- 10 <u>וגם</u> כל קפיצה בוקטור הרדיוס אשר קטנה מ- (-5) מטרים.

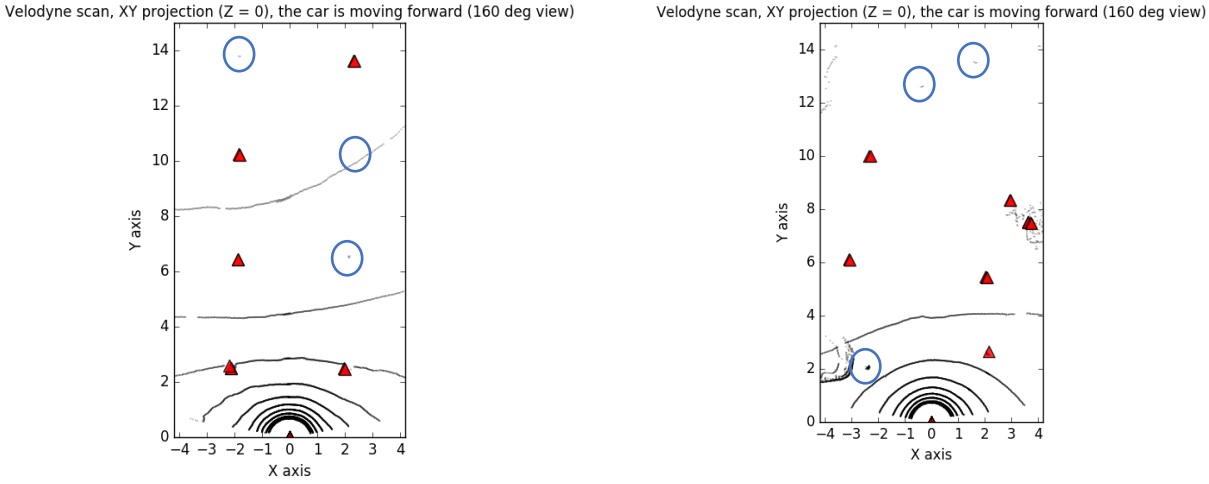



איור 25 - תצוגת זיהוי קונוסים $R < -5m \;\&\&\; REF \geq 10$

כפי שניתן להתרשם מהתוצאות, באמצעות תנאי חדש זה סינון הרעשים נהיה טוב יותר משמעותית, אך עם זאת אנו מאבדים הרבה מידע רלוונטי של קונוסים שתויגו כראוי בתוצאות הקודמות (של כל תנאי לחוד). ניתן לראות שבכל תמונה ישנם ארבעה קונוסים שלא תויגו (לנוחות הנקודות מוקפות בתצוגה המקורית בעיגולים כחולים).

לאור תוצאות אלו, הוחלט לבדוק האם גם קפיצות חיוביות ברדיוס (כי עד רגע זה נבדקו רק קפיצות שליליות בהנחה שדגימת הכביש יותר רחוקה מדגימת הקונוס) מביאות לתוצאה רצויה ותיוג נכון. לכן ברמת הסינון הבאה בחנו את איחוד הפילטרים בשנית, רק שכעת שני התנאים (רפלקטיביות ורדיוס) נבחנו לאור הערך המוחלט שלהם, כלומר הבדיקה הינה עבור קפיצות חיוביות ושליליות בו זמנית.

ד. תיוג קונוס מוגדר ככל קפיצה בערך מוחלט של וקטור הרפלקטיביות אשר גדול שווה מ- 10 וגם כל קפיצה בערך מוחלט של וקטור הרדיוס אשר גדול שווה מ- 5 מטרים.




איור 26 - תצוגת זיהוי קונוסים $REF \geq 10 \;\&\&\; |R| \geq 5m$

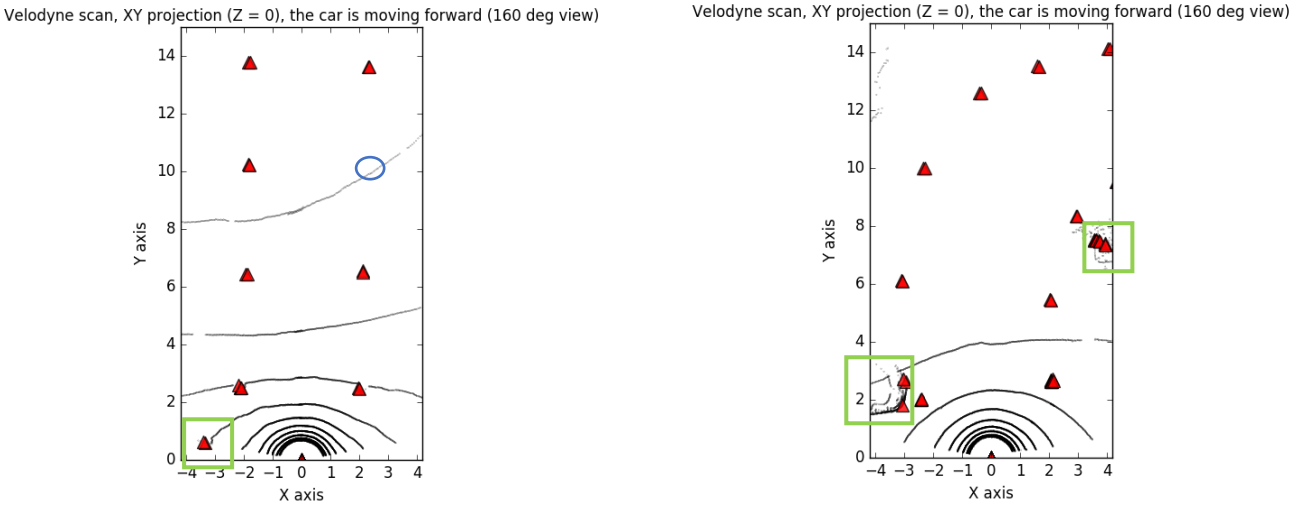
כפי שניתן להתרשם מהתוצאות, התנאי חדש שבודק קפיצות חיוביות ושליליות בערכי הרדיוס והרפלקטיביות מביא לסינון רעשים סביר (עדיין ניתן לראות זיהוי שגוי של מדרכה כקונוס בצדו הימני של המסלול שמתאר פנייה ימינה), אך עדיין הרבה נקודות אינן מתויגות כפי שנאו מצפים שיקרה (לנוחות הנקודות מוקפות בתצוגה המקורית בעיגולים כחולים).

לאחר בחינת תנאי סינון זה חזרנו אל קובץ הפלט שמתאר את הקפיצות ברדיוסים וברפלקטיביות על מנת לדגום בעצמינו כמה נקודות שאנו יודעים בוודאות שהן קונוסים, ובכדי לבדוק מהם המאפיינים הייחודים של קפיצות שמתרחשות בנקודות אלו ביחס לשאר הנקודות במרחב הדגימה.

כאשר עשינו בדיקה זו, גילינו כי רמות הקפיצה בוקטור הרפלקטיביות אינן בעלות תחום מספרים אחיד עבור כלל הנקודות אשר אמורות להיות מתויגות כקונוס. לכן, הוחלט בשלב זה לשלול את האפשרות להשתמש בוקטור הבדלי הקפיצות ברפלקטיביות כמדד כשלהו לזיהוי קונוסים במרחב. כעת, נותר לנו רק פלט וקטור הבדלי הקפיצות ברדיוס.

מבדיקה שערכנו עבור וקטור זה, בנקודות ספציפיות שיודעות כקונוסים, גילינו כי הסף שהגדרנו עד כה למדד מהו קונוס, היה גבוה מדיי וכי ישנן נקודות שהקפיצה בהן ביחס לדגימת הכביש נעשית באזור תחום הערכים של 1.6- מטרים. לכן ברמת הסינון הבאה התנאי הוגדר לוקטור הרדיוס בתחום זה.

ה. תיוג קונוס מוגדר ככל קפיצה בוקטור הרדיוס אשר קטנה מ- -1.6m




איור 27 - תצוגת זיהוי קונוסים R < -1.6m

עבור תנאי זה ניתן להתרשם שקיבלנו את תוצאת התיוג הטובה ביותר עד כה. למעשה, חוץ מקונוס אחד כלל הקונוסים שקיימים במציאות מול החיישן זוהו בפריים. בבדיקת הסיבה העומדת מאחורי אי זיהוי ותיוג של הקונוס "הבעייתי" (מוקף בעיגול כחול בתמונה השמאלית) גילינו שמכיוון שדגימת הכביש התאחדה עם דגימת הקונוס אנו לא יכולים לקוות לקפיצה משמעותית בוקטור הרדיוס לה ציפינו. בעיה זו נפתרת כמובן עם התקדמות הרכב, מכיוון שהלייזר הספציפי שדוגם את הכביש גם כן יתקדם במרחב, ולכן הקפיצה ברדיוס בהכרח תתרחש (הקונוס נותר במקומו בזמן שדגימת הכביש מתקדמת במרחב).

בעיה אחרת שניתן לראות בתוצאות היא העובדה שאנו לא מצליחים לסנן בצורה אבסולוטית את "רעשי הסביבה" שבמקרה זה הינם אובייקטים כלשהם במרחב שאינם כביש או קונוס, לדוגמא במקרה שלנו מדרכה/רכב שהיו בחנייה בה הוקלט הניסוי (מסומנים בריבוע ירוק בתצוגה).

בשלב זה של הפרויקט הוחלט שזוהי רמת הסינון אשר מספקת אותנו, וזאת בהתחשב בידע מקדים שיש לנו על מסלול התחרות, בו אנו מצפים כי לא תהיה סביבה "רועשת" אשר מכילה אובייקטים בלתי רצויים.

## בעיית זיהוי צבע קונוס

כחלק ממטרות הפרויקט הכוונה הראשונית הייתה זיהוי קונוסים על מנת שנוכל למפות מסלול מלא בקדמת החיישן. אולם, על מנת לעשות זאת אנו זקוקים למידע נוסף אודות צבעי הקונוסים מכיוון שצדדי המסלול השונים מזוהים בעיקר בשל צבעי הקונוס השונים- צד ימין מסומן על פי קונוס צהוב עם פס שחור באמצעו, וצד שמאל מסומן על פי קונוס כחול עם פס לבן באמצעו.

כפי שניתן להבין, זיהוי של צבעי הקונוסים הינו חלק קריטי עבור בקרה הניווט של הרכב במסלול וזאת מכיוון שהמסלול איננו ידוע מראש ולכן כל החלטה על שינוי זווית הגה ברכב אוטונומי מתקבלת על סמך מידע מהחיישנים שמתארים למערכת הבקרה את סביבת המסלול.

לאור העובדה כי הקונוסים מכילים פס של צבע לבן וצבע שחור, ולאור העובדה כי פלט חיישן המרחק מספק ערכים קבועים עבור החזרים (רמת רפלקטיביות) מצבע לבן או צבע שחור הוחלט לבחון את האפשרות לזהות את צבעי הקונוסים על סמך מידע זה.

לאחר בדיקת נקודות ספציפיות הידועות לנו מראש כקונוסים, הקיימות בתוך וקטור ערכי הרפלקטיביות הובן לנו כי ערכי רמת ההחזר הינם בתחום רחב מדיי אשר איננו זהה לנתוני היצרן אודות ערכים קבועים ללבן ולשחור.

הסיבה ככל הנראה טמונה בכך שרמת ההחזר תלויה גם בסוג החומר ומבנה הקונוס ולא רק בצבעו, כך שערכי רמת ההחזר שאנו מקבלים עבור צבע שחור בקונוסים צהובים שונים אינם אותם ערכים ומקשים לקבוע חד משמעית כי מדובר בקונוס צהוב ולא כחול (בהתאמה גם בקונוסים הכחולים והצבע הלבן).

לאור תוצאות בדיקה זו, אנו בחרנו להתמודד עם בעיית צבעי הקונוסים באמצעות התחשבות במידע מוקדם על אופי הנחת הקונוסים ביחס לרכב.

ההנחה היא שאם בסיבוב הראשון הרכב מבצע נסיעה איטית יחסית בו בעזרת התוכנה שנכתבה בפרויקט זה, הרכב מקבל בכל רגע נתון זווית היגוי מתאימה וכמו כן מתבצע מיפוי מלא של הקונוסים במסלול ושל המסלול שביצע הרכב בפועל, אז ניתן לאחר סיום הסיבוב הראשון לתייג בצורה ודאית את כלל צבעי הקונוסים – צהוב מצדו הימני של המסלול וכחול מצדו השמאלי.

בנוסף, ניתן תוך כדי תנועת הרכב לבצע זאת על ידי שימוש באותה ההנחה כי קונוסים שזוהו ודאית ונמצאים כרגע בזווית של 160-180 מעלות ביחס לרכב, כבר נכנסו ודאית למיפוי ולכן ניתנים לתיוג צבע ודאי תוך כדי תנועה – צהוב מצדו הימני של המסלול וכחול מצדו השמאלי.

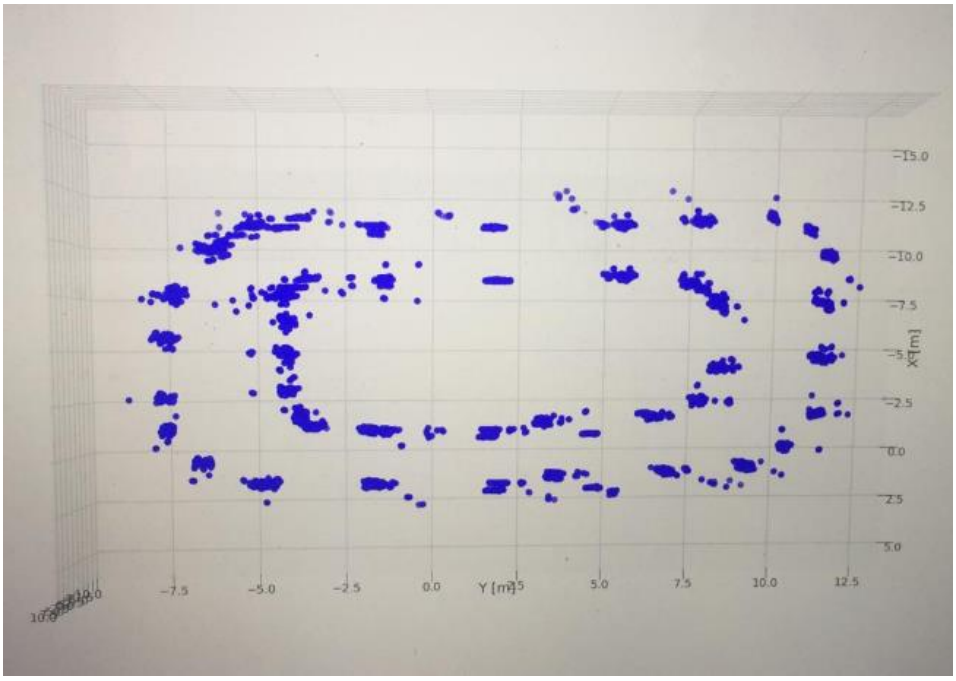
# ביצוע Clustering

על מנת להבין את הצורך בשלב זה נסכם תחילה את כלל השלבים הקודמים-

1. קבלת פלט גולמי מחיישן ה- LiDAR וביצוע פענוח למידע.
2. ביצוע סינון למידע לא רלוונטי וזיהוי נקודות חשודות כקונוסים במערכת ייחוס של ה-LiDAR.
3. המרת הנקודות החשודות ממערכת ייחוס של ה- LiDAR למערכת ייחוס גלובלית באמצעות אינטגרציה עם פלט חיישן ה-IMU.
4. קבלת פלט ויזואלי במערכת ייחוס גלובלית- כלומר, נקודות שנראו בכמה פריימים שונים במערכת ייחוס של ה-LiDAR במקומות שונים על המפה, יימצאו כעת באותו האזור במערכת ייחוס גלובלית.

להלן דוגמא ויזואלית לתוצאת פלט במערכת ייחוס גלובלית בניסוי שביצענו, כאשר בעזרת ההתקן (שני החיישנים ומחשב client) דימינו נסיעה בתוך מסלול מעגלי-


איור 28- ביצוע ניסוי מסלול מעגלי




איור 29 - פלט מסלול מעגלי, מערכת ייחוס גלובלית

לאור תוצאה זו, כל שנותר הוא לבצע פעולת clustering כלשהי על מנת לקבל בזמן אמת נקודה אחת שמייצגת כל קבוצת נקודות במרחב.

בפרויקט זה בחרנו לבדוק שתי שיטות עיקריות לביצוע clustering:

1. **מיפוי הסתברותי באמצעות Probability density function – PDF:**

בשיטת זו אנו למעשה בונים פונקציית צפיפות (Probability density function, בראשי תיבות PDF) גאוסיאנית המתארת את צפיפות המשתנה בכל נקודה במרחב המדגם. ההסתברות שמשתנה מקרי יימצא בקטע מסוים היא האינטגרל של הצפיפות בקטע ולכן המשתנה נוטה יותר לקבל ערכים שבהם הצפיפות גבוהה. לפי תכונה זו, אנו מצפים לקבל גאוסיאן בעל ערך Z יותר גבוה (במפה תלת ממדית) ככל שבאזור מסוים יש יותר נקודות.

כך למעשה על ידי ביצוע פעולת סף – חיתוך ציר Z מעל ערך threshold מסוים, אנו מקבלים קיבוץ של כלל הנקודות באזרו מסוים לנקודה בודדת אותה אנו יכולים לתייג בוודאות גדולה כקונוס במפה שלנו.

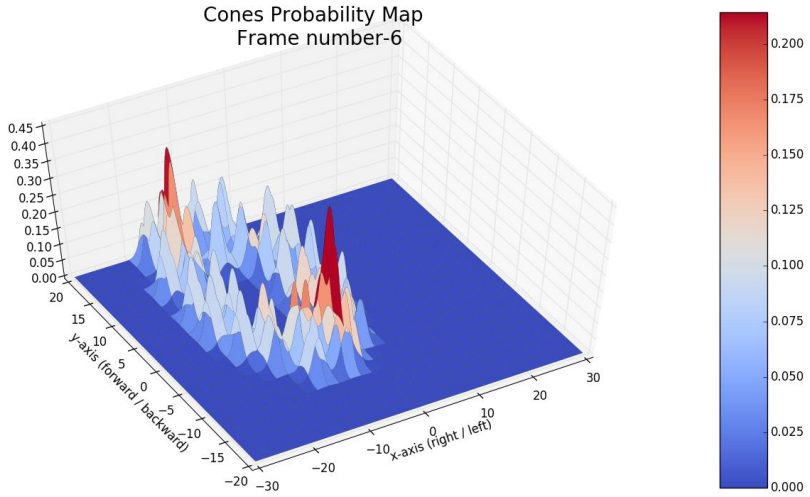
בפועל בעת בדיקות שביצענו עבור שימוש בשיטה זו על מנת לקבל קיבוץ של תמונת הקונוסים במערכת צירים הגלובלית, חיתוך ציר ה-Z לפי ערך סף מסוים לא הפיק תוצאה מיטבית.

על מנת לא לפספס קונוסים שקיבלו ערך Z נמוך יחסית לגאוסיאנים אחרים (כי הקונוס הופיע פחות במהלך הזיהוי), היינו צריכים להתפשר על ערך סף עבור ציר ה-Z שהינו נמוך יחסית. כך שקיבלנו בתוצאה הסופית שוב המון נקודות עבור קונוס שפשוט זוהה הרבה פעמים ונקודות בודדות עבור קונוס שזוהה פחות.

תוצאה זו הזכירה לנו למעשה את התוצאה שקיבלנו כבר באיור 29, רק שכעת הסדר נשמר יותר והרעש נוקה כמעט לחלוטין. החיסרון העיקרי הוא שכעת אנו צריכים לבצע על התוצר גם סוג של תהליך קיבוץ clustering, ולכן שימוש בשיטה זו התברר כלא יעיל כל כך מבחינה מעשית.

חשוב לציין שבידי המפעיל אפשרויות רבות לדינמיות בשינויים שניתן לבצע בפונקציית ה-PDF שצורפה לפרויקט זה (לדוגמא שינויי אמפליטודה, שונות של צירי X ו- Y וכדומה). לדוגמא אנו השתמשנו בפונקציה זו על מנת לחשב את זווית ההיגוי, כפי שהוזכר קודם לכן בספר זה.

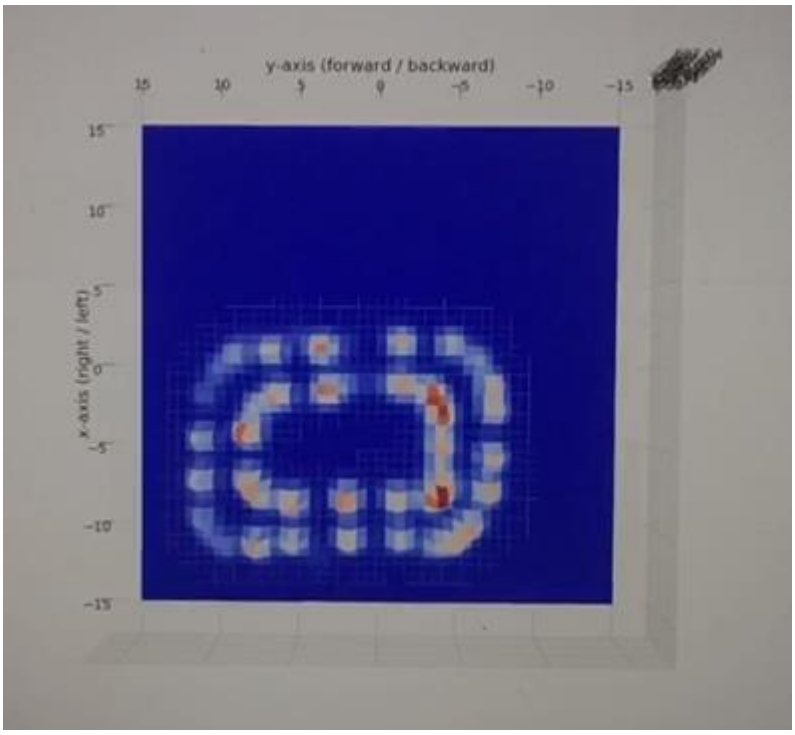
להלן דוגמא ויזואלית לשימוש במתודת מיפוי הסתברותי עבור מסלול מעגלי:



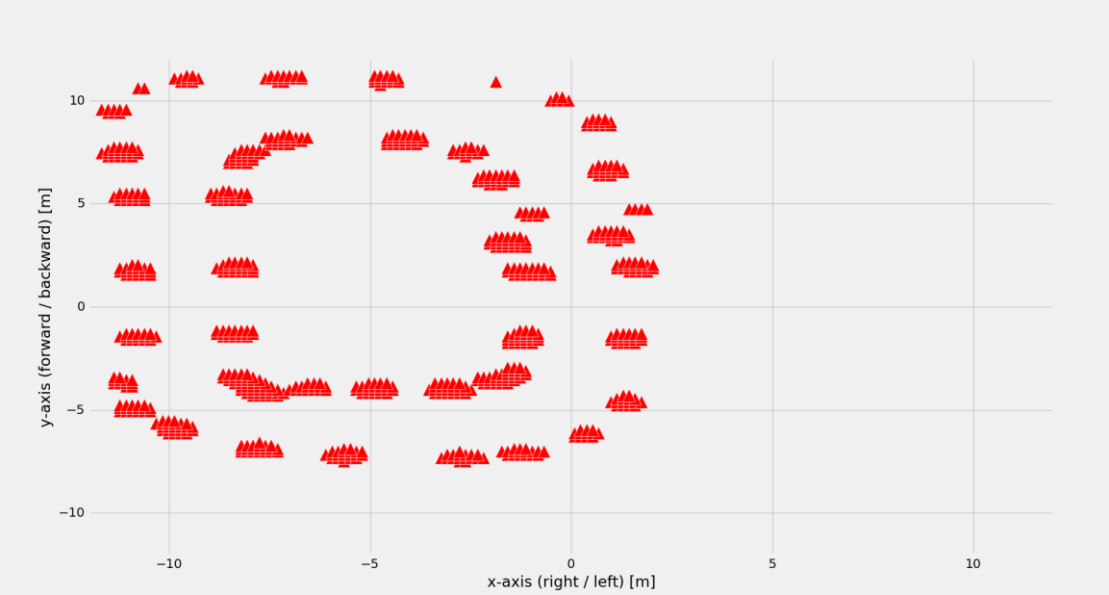

איור 30- מיפוי באמצעות PDF- מבט צד (מסלול מעגלי)

להלן תוצר של מיפוי איור 29 לאחר בחירת ערך סף בגובה $Z > 4000$ ,

כאשר $\sigma_x, \sigma_y = 0.1,0.3$ :




איור 31 - מיפוי באמצעות PDF - מבט עילי לאיור 29




איור 32- תוצר חיתוך בסף $Z > 4000$

## 2. מיפוי באמצעות שימוש באלגוריתם K-means:

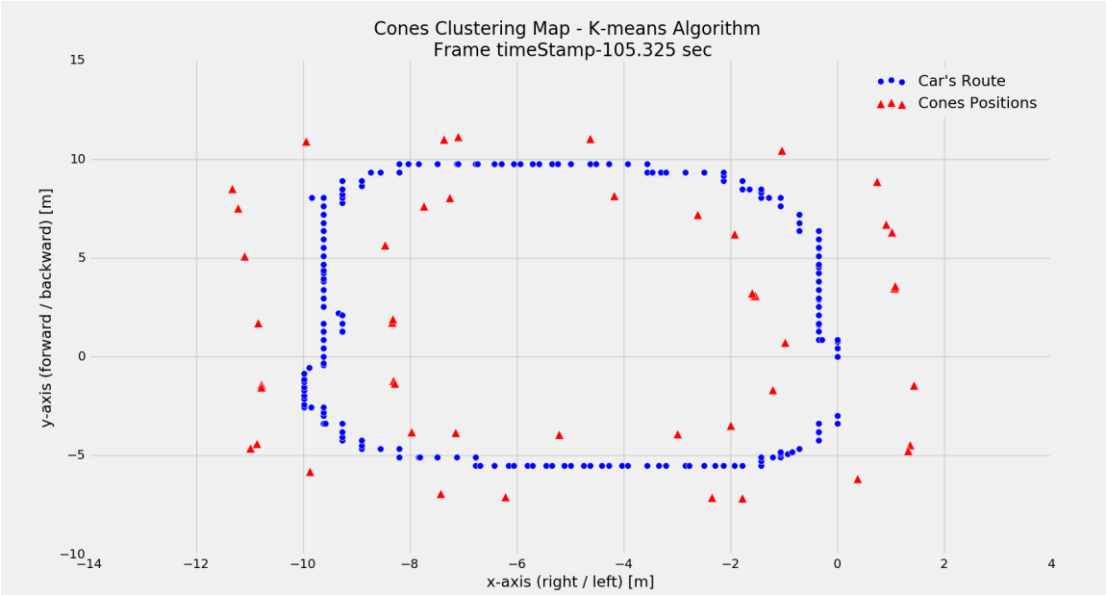
אלגוריתם k-means הינו שיטה פופולרית עבור ביצוע Clustering בכריית נתונים. מטרתו למעשה הינה חלוקה של כלל תצפיות הקונוסים בפריים יחיד (או במקרה שלנו - בנתוני כמה פריימים בודדים שאוחדו יחדיו) ל- k אשכולות לפי מרכזי כובד [5].

זהו מודל סטטיסטי שאינו מתבסס על ידע מוקדם על הנתונים אלא רק על תצפיות בפועל. שיטה זו דומה לאלגוריתם Expectation–maximization המניח התפלגות הדוגמאות מכמה התפלגויות גאוסיאניות.

האלגוריתם עובד בהינתן קבוצה של תצפיות $(X_1, X_2, X_3, \dots, X_n)$, כאשר כל תצפית היא וקטור ממשי היכול להיות בעל מספר ממדים. מטרת המודל היא לחלק את n התצפיות ל-k אשכולות, על מנת למזער את סכום המרחקים בין התצפיות בתוך האשכול.

שיקולים משמעותיים ביישום המודל הם בין השאר מספר המרכזים (K) שאנו רוצים למצוא והכמות מינימלית של נקודות לאשכול. לאור כך, בפרויקט זה בחרנו לאסוף נקודות שהצטברו לאורך כמה פריימים (נתון לשינוי על ידי המפעיל ותלוי מהירות הרכב) ולהגדיר מראש כי כמות הקונוסים K אשר אנו מצפים לקבל באוסף הפריימים הללו הינה 4. כמובן שהכל ניתן לשינוי ולהחלטת המפעיל של התוכנה.

להלן התוצאות שקיבלנו בעת הפעלת האלגוריתם עבור מסלול מעגלי (במערכת גלובלית) שהוצג באיור 29:




איור 33 - מיפוי בשיטת K-means

כפי שניתן לראות באיור, המשולשים האדומים מייצגים את הקונוסים לאחר ביצוע איחוד אשכולות, clustering, בשיטת k-means. על גבי אותו הגרף ניתן גם למצוא בנקודות כחולות את המסלול שביצע הרכב בפועל (על ידי קבלת הנתונים מה-IMU).
בשיטה זו ניתן להיווכח כי קיבלנו תוצאה הרבה יותר מרשימה מאשר בשיטה קודמת שהוצגה, ולכן זוהי גם המלצתנו לביצוע המיפוי הרצוי בפרויקט רכב הפורמולה האוטונומי.

**<u>נקודות חשובות:</u>**

- בתוצר שהוצג לעיל השתמשנו למעשה במעבר על כ- 40 פריימים (כ-4 שניות) בטרם נשלחו כלל הנקודות שהצטברו לביצוע clustering בעזרת השיטה שהוצגה, זאת כאשר ביקשנו למצוא 4 קונוסים מבין כלל הנקודות שהצטברו. ההנחה שעבדנו לפיה היא שבמהירות הניסוי של הרכב/עגלת ניסוי כל 4 שניות בערך משתנה הפריים של החיישנים כמעט לגמרי, ובכל פריים אכן בערך אנו מצפים למצוא 4 קונוסים.
- כל הערכים שבעזרתם קבענו את הניסוי ואת השימוש באלגוריתם ניתנים שלינוי על ידי המפעיל וזאת התאם לנתוני מהירות הרכב שהוגדרו.
- הניסויים בוצעו בחניית הרכבים של הפקולטה למכונות, שנחשב אזור ניסויים "מורעש" יחסית לאור כמות העצים/מדרכות/הפרעות נוספות. לכן, ניתן לראות גם בתוצר שקיבלנו כי ישנה סטייה קלה של הקונוסים מהמקום הטבעי שהיינו מצפים לקבל במיפוי. הנחה סבירה בהחלט שבתנאי תחרות בהם אין כמעט אובייקטים במסלול מלבד קונוסים, תוצאות המיפוי יהיו טובות אף יותר.

# מדידת השגיאה

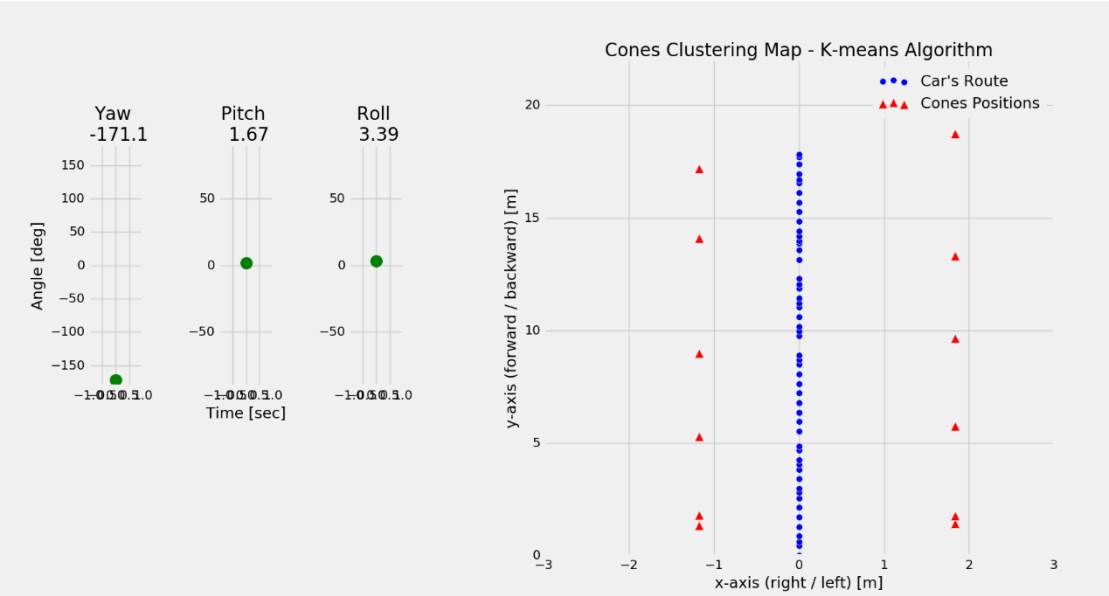
מדד חשוב עבורנו על מנת להבין את טיב המיפוי שאנו מספקים עבור הרכב, הינו שגיאת המרחק האוקלידי הממוצעת בין מיקומי הקונוסים שהתקבלו במיפוי הסופי לבין מיקומי הקונוסים בפועל במציאות.

על מנת לבצע חישוב זה ביצענו ניסוי של מסלול ישר ומסלול מעגלי בהם הנחנו את הקונוסים במרחקים שווים זה מזה, כאשר שמרנו את הנתונים הללו בצד לשם השוואה עם תוצאות המיפוי.

בתמונה הבאה ניתן לראות דוגמא ויזואלית לניסוי שערכנו שבו על מנת לבצע את חישוב השגיאה, הנחנו קונוסים במרחק רוחבי של 3 מטרים ובמרחק אורכי של 4 מטרים:

איור 34 - מדידות שגיאה - מסלול ישר

בהתאם להקלטת ניסוי זה קיבלנו פריים מהמיפוי כולו שנראה כך-




איור 35 - פלט ניסוי חישוב שגיאה- נסיעה ישרה

כאשר התוצאות של מיקומי הקונוסים הינם:

| X | Y |
|---|---|
| 1.833768 | 1.404511 |
| -1.17715 | 1.787582 |
| 1.833768 | 1.744537 |
| -1.17715 | 1.331121 |
| 1.833768 | 5.755747 |
| -1.17715 | 5.270567 |
| 1.833768 | 9.656064 |
| -1.17715 | 8.977213 |
| 1.833768 | 13.29846 |
| -1.17715 | 14.10558 |
| 1.833768 | 18.73515 |
| -1.17715 | 17.18786 |

טבלה 2- תוצאות מיקומי קונוסים במיפוי - נסיעה קו ישר

מחישוב של ממוצעי השגיאות לאורך ולרוחב, קיבלנו כי שגיאת הרוחב עמדה על כ-12 ס"מ, ושגיאת האורך עמדה על כ- 10.5 ס"מ.

# סיכום

פרויקט זה מעמיד פתרון ראשוני לאתגר הזיהוי והמיפוי של אובייקטיבים במרחב, הכולל בתוכו ביצוע עיבוד בזמן אמת ובמהירות גבוהה, באמצעות טכנולוגיית ה- LiDAR.

פתרון זה כולל בין השאר ביצוע תיוג מדויק של נקודות החשודות כקונוסים במרחב, וזאת עד 15 מטר מקדמת הרכב בכל זמן נתון, כלומר מאפשר זיהוי של כ- 3 קונוסים לפחות משני צדדי הרכב. תיוג מהיר ומדויק זה בהחלט מספק "תמונת עולם" למחשב בקרת ניווט הרכב, אשר בעזרתה ניתן יהיה להבטיח מעבר בטוח של הרכב במסלול המיועד.

חשוב להזכיר כי תוצאה מרשימה נוספת היא העובדה כי דיוק המדידה שחושב במהלך עבודה על פרויקט זה בין מצב הקונוסים בפריסה בפועל במסלול לבין המיפוי אשר התקבל בתוכנה, עמד על סטייה של כ- 11 ס"מ בממוצע.

בנוסף, באמצעות ביצוע פעולות איחוד אשכולות, clustering, בעזרת שיטת K-means הצלחנו למעשה ליצור מיפוי מלא בה כל נקודה במרחב ייחוס גלובלי קיבלה למעשה מיפוי של קונוס במסלול ביחס לרכב.

שיטה זו עונה על המטרות שהצבנו בתחילת הפרויקט ובהחלט מהווה בנייה מלאה מאפס של פתרון מיפוי שיכול לשמש את פרויקט הפורמולה בבחינתו פתרונות מיפוי נוספים בעתיד.

# הצעות לעבודה עתידית

לאור ההיכרות הרבה שלנו עם אתגר המיפוי בזמן אמת של רכב הפורמולה ולאור התוצר אליו הגענו, נציג מספר כיווני המשך שמצאנו בהם עניין להמשך ביצוע ומחקר, אך בעקבות מגבלות זמן לא הצלחנו לממש.

אנו מציעים לממש את כיווני התקדמות אלה כפרויקטיי המשך או כתוספות לפרויקטים המתבססים על המערכת שבנינו:

- המשך ביצוע בדיקות מעשיות של מערכת המיפוי שהוצעה בפרויקט זה על רכבים אוטונומיים אשר עתידיים להיבנות כחלק מפרויקט הפורמולה הטכניוני.
- ביצוע בקרה על סחיפת רכיב ה-IMU אשר נצפתה בחלק מהניסויים בפרויקט זה.
- ביצוע אופטימיזציה של מיפוי המסלול באמצעות שיטות clustering שונות ואולי אף יעילות יותר מאלו אשר הוצעו ונבדקו במסגרת פרויקט זה.
- כתיבת פתרון תוכנתי לבעיית זיהוי צבעי קונוסים באמצעות רשת נוירונים (FC /CNN) כאשר הקלט יכול להיות תמונות ענן נקודות / פריים של נקודות חשודות בודדות מול הרכב, בתוספת נתונים הקיימים כבר ברשותנו כגון- רדיוס מכל נקודה ורפלקטיביות הנקודה.

# תודות

פרויקט זה בוצע כחלק מפרויקט רחב יותר, שהינו פרויקט רכב הפורמולה האוטונומי 2018.
כתוצאה ישירה, הפרויקט כלל מעורבות של כמה גורמים אשר אנו מעוניינים בהזדמנות זו לומר להם תודה ענקית על כך שאפשרו לנו למעשה ללמוד תחומים חדשים, מרתקים ומאתגרים עם המון עניין ופרקטיות שהתלוותה לכך.

תודה ענקית בראש ובראשונה מגיעה למנחה שלנו, מר דני וייכרמן, אשר ליווה אותנו מתחילת הפרויקט ואפשר לנו להגיע לתוצר המוצג בספר זה.

תודה למעבדת VISL ובראשה למר יוחנן ארז אשר היוו לנו פלטפורמת עבודה יעילה ומאתגרת, ואפשרה לנו להשקיע מחשבה עמוקה בכל תהליכי הפרויקט במקביל ללימודינו בפקולטה להנדסת חשמל.

תודה למעבדת CRML ובראשה מר קובי כוחיי וגברת אורלי ויגדרזון, אשר סיפקו לנו ייעוץ ושימוש ברכיבי המעבדה על מנת לפתור בעיות שעלו בעת הכנת פרויקט זה.

ולבסוף, תודה לכלל חבריי פרויקט רכב הפורמולה האוטונומי 2018, אשר בהחלט הורגשו בכל החלטה שביצענו במהלך העבודה על פרויקט זה.

# רשימת מקורות


[1] Velodyne, “velodynelidar.com .[מקוון] ”,Available: http://velodynelidar.com/downloads.html.

[2] Q. H. Truong, “Knowledge-based 3D point clouds processing,” Université de Bourgogne, Bourgogne, 2013.

[3] VectorNav, “vectornav.com .[מקוון] ”,Available: https://www.vectornav.com/products/vn-200/documentation.

[4] A .Noureldin, T. B. Karamat ו J. Georgy, Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration, springer, 2013 .

[5] wikipedia, “https://en.wikipedia.org/wiki/K-means_clustering .[מקוון] ”,

[6] Y. Ioannou, “github.com .[מקוון] ”,Available: https://github.com/yanii/kitti-pcl.

[7] N. Pickett, “github.com .[מקוון] ”,Available: https://github.com/vadmium/py-pcap.