הפקולטה להנדסה ע"ש איבי ואלדר פליישמן
אוניברסיטת תל אביב

בי"ס להנדסת חשמל

פרויקט מס' 21-2-1-2424

דו"ח סיכום

שם הפרויקט: מערכת הקלטות מסונכרנת לרכב אוטונומי

מבצעים:

שם: אבי צחי ת.ז. 315901231

שם: שקד נתן ת.ז. 208709816

מנחה:

רועי רייך אוניברסיטת תל אביב

מקום ביצוע הפרויקט: אוניברסיטת תל אביב - מעבדת רכב אוטונומי

תוכן עניינים

רשימת איורים

רשימת טבלאות

תקציר:

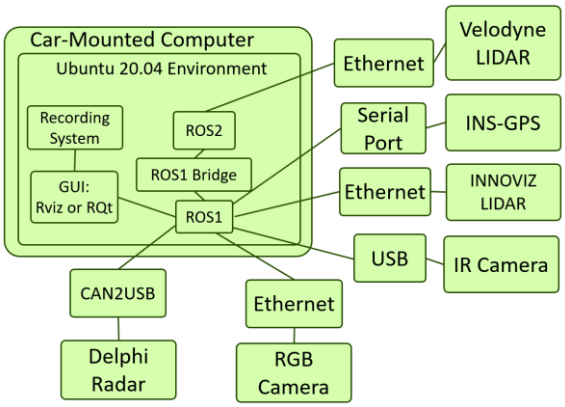
תחום הרכב האוטונומי צבר תאוצה משמעותית בשנים האחרונות ויצרניות רכב רבות עוסקות בו, ובנוסף יש בו גם חברות שאינן יצרניות רכב, למשל מובילאיי, המתמקדת במערכת המאפשרת לרכב לנסוע באופן אוטונומי (חיישנים, עיבוד המידע, בקרת הרכב). פרט לתחום התחבורה היבשתית, תחום התחבורה האוטונומית כולל תחבורה נוספת – ימית ואווירית. המשותף לכל המוזכרים לעיל הוא שבכולם יש צורך במערכת האוספת מידע מחיישנים שונים, מאחסנת ומעבדת אותו.

פרויקט זה הינו חלק מפרויקט מערכתי בפקולטה להנדסה להקמת מעבדה בנושא רכבים אוטונומיים. במסגרת המעבדה, אשר הוקמה בשיתוף עם התעשייה האווירית, ישנו רכב נוסעים מסוג KIA Niro בו מותקנים חיישנים שונים: רדאר, מצלמות (מצלמת IR ומצלמת RGB), סורקי לייזר (LIDAR) ו-GPS. החיישנים מחוברים למחשב המותקן ברכב ועליו מתבצעים ההקלטה ועיבוד הנתונים. מטרת הפרויקט היא הקמת מערכת חדשה לתיעוד נסיעה, באופן מסונכרן, באמצעות החיישנים השונים וייצוא ההקלטות בפורמט נוח לעיבוד, כגון: CSV , JPEGs , MP4, כך שניתן יהיה לפתח את האלגוריתמים הנדרשים לצורך נהיגה אוטונומית של רכב המעבדה.

מימוש הפרויקט התבצע בסביבת לינוקס (Ubuntu) ובעזרת ROS2, ROS1 כאשר המושג ROS יורחב בפרק 2 (רקע תיאורטי).

התוצר הסופי של הפרויקט הוא מערכת הקלטת נסיעה מסונכרנת ומלאה, ממומשת ב-ROS, המסוגלת להקליט מידע מכל החיישנים המחוברים אליה. למערכת יש ממשק משתמש גרפי שבאמצעותו אפשר לבצע את כל הפעולות הבסיסיות – צפייה ב-Feed של החיישנים, מעבר בין מצב מסונכרן ולא מסונכרן.

בנוסף להקלטה, ניתן לייצא את המידע בפורמט נוח לעבודה (למשל CSV או JPEG) מכל חיישן בנפרד, וניתן להציג מידע מחיישני הרדאר וה-LIDAR באופן מקבילי בתצוגה תלת ממדית. ניתן לתאר את המערכת באמצעות דיאגרמת הבלוקים הכללית הבאה:



איור 1 - דיאגרמת בלוקים כללית

חלק 1 - הקדמה:

1.1 – מטרות הפרויקט

לפרויקט זה מספר מטרות שהוגדרו על מנת להגשים את חזון הרכב האוטונומי:

- **מערכת הקלטות מסונכרנת לרכב אוטונומי:** הכוונה במערכת מסונכרנת היא שנראה על אותו ציר זמן אירועים שקרו באותו הזמן עד כמה שאפשר (מכיוון שלא נוכל להגיע למצב של רמת דיוק מושלמת). כל חיישן עובד בתדר שונה משלו ואנו רוצים לתאם את צירי הזמן עד כמה שאפשר ולקבל ציר זמן אחיד. כלומר, נרצה לקרב את תדרי הדגימה כך שנקבל תמונה כמה שיותר מסונכרנת.
- **מידע ממגוון חיישנים:** המידע נאסף בזמן אמת ממספר רב של חיישנים המותקנים על הרכב, פירוט נוסף על החיישנים השונים יינתן בפרק 3 (מימוש):
  - חיישן LIDAR מסוג Velodyne VLP16 (יצרן: Velodyne)
  - חיישן LIDAR מסוג InnovizeOne (יצרן: Innoviz)
  - מצלמת אור נראה (RGB) מסוג Prosilica GT1930c (יצרן: Allied Vision)
  - מצלמה תרמית מסוג Boson FLIR (יצרן: Teledyne FLIR)
  - מכ"ם פגוש מסוג Delphi ESR 2.5 (יצרן: Delphi)
  - חיישן INS מסוג INS-DL (יצרן: InertialLabs)
- **ייצוא ההקלטות בפורמט נוח לעיבוד:** ישנם מספר סקריפטים שנכתבו על מנת שנוכל לייצא את הקלטות החיישנים השונות בפורמט נוח לעיבוד. הייצוא מתבסס על CSV, פורמט JPEG לתמונות, MP4 לווידאו, PCDs ל-pointcloud.
- **הקמת GUI מאוחד:** ה-GUI כולל את הצגת כל החיישנים השונים בתצוגה ויזואלית נוחה. כמו כן, ניתן לבצע הקלטות מתוך ה-GUI ולהציג את תוצאות ההקלטות בתוך ה-GUI עצמו.
- **התקנת החיישנים במערכת ROS2:** היה צורך להקים מערכת שמכילה, בנוסף ל-ROS1 גם את ROS2, על מנת שהמשתמש יוכל לבחור האם הוא רוצה, בנקודת זמן ספציפית, לעבוד עם ROS1 או לחלופין עם ROS2.
- **יכולת הצגת מידע נוחה ממספר חיישנים:** בפרויקט זה הצלחנו להציג את הרדאר וה-LIDAR באופן מקבילי על אותו frame. בצורה זו, ניתן לקבל תצוגה תלת ממדית של החלל מסביב לרכב.

1.2 - המוטיבציה

לפרויקט זה ישנן מספר מוטיבציות הנדסיות:

- **יכולת הקלטה ממספר חיישנים בו זמנית:** בעבודה עם המערכות שמקבלים מכל יצרני החיישנים, לא ניתן לבצע הקלטות בו זמנית של מספר חיישנים, מכיוון שכל חיישן ממוקם במערכת אחרת. לכן, מערכת ההקלטות המסונכרנת שהוקמה בפרויקט זה על בסיס ROS פותרת בעיה זו ומאפשרת לבצע הקלטה בו זמנית ממספר חיישנים.
- **סנכרון:** צורך במערכת הקלטות מסונכרנת, המאפשרת לבחור נקודת זמן רצויה ולקבל מידע מכל החיישנים בנקודת הזמן הזו. בין היתר למערכת המסונכרנת חשיבות ב-Fusion, שם הסנכרון חשוב במיוחד על מנת לקבל תמונה המתארת את המציאות בצורה האופטימלית, וחשיבות גדולה אף יותר כאשר רוצים ליישם אלגוריתמים מתקדמים המשתמשים במידע מהחיישנים לצורך נהיגה אוטונומית.
- **מודולריות:** בניגוד למערכות שלמות המגיעות מהיצרן עם כל החיישנים ומערכת ההקלטות כיחידה אחת, המערכת העומדת במרכזו של פרויקט זה היא מערכת מודולרית, שבה ניתן להוסיף או להסיר חיישנים לפי הצורך – למשל לצורך שדרוג של אחד החיישנים או לצורך הרחבת יכולות המערכת. במערכת המגיעה כיחידה אחת יש לרוב להחליף את כל המערכת לצורך שדרוג, ולא ניתן להוסיף יכולות כרצוננו – כלומר מערכת כזו יוצרת תלות ביצרן המערכת, שלעתים (למשל ביישומים צבאיים) עדיף להימנע ממנה. לכן, מערכת ההקלטות נבנתה על בסיס ROS, השימוש במערכת ROS ממלא תפקיד חשוב – מערכת זו מאפשרת, כאמור, להתממשק עם חיישנים שונים בעזרת אותה מערכת ובכך מקלה מאוד על יצירת מערכת מודולרית, שכן אם היינו מנסים להתממשק עם כל חיישן בנפרד בעזרת כלים ייעודיים לו המסופקים על ידי היצרן היינו נאלצים לכתוב בעצמנו מערכת הקלטות מאפס.
- **תמיכה בעבודה עם ROS2 במעבדת הרכב האוטונומי:** עבודות קודמות שבוצעו במסגרת פרויקטים המקושרים לרכב האוטונומי היו מבוססים על ROS1 , כגון הפרויקט הקודם שבוצע באובונטו 18 ליצירת מערכת הקלטות ב-ROS. מכיוון שלא ידוע מה יקרה בשנים הבאות במסגרת העבודה עם ROS אז היה צורך לבנות מערכת שמכילה, בנוסף ל-ROS1 גם את ROS2. זה אכן בוצע בפרויקט זה וישנה מערכת אחודה שמפעילה את החיישנים השונים גם ב-ROS1 וגם ב-ROS2. המשתמש יוכל לבחור האם הוא רוצה, בנקודת זמן ספציפית, לעבוד עם ROS1 או לחלופין עם ROS2.

1.3 – גישת העבודה

כפי שנאמר, מערכת ההקלטות המסונכרנת נבנתה על בסיס ROS , עקב היתרונות הנעוצים בה בהתממשקות עם חיישנים שונים שמשתמשים בפרוטוקולי תקשורת שונים. בנוסף לכך, ROS היא מערכת חינמית ו-open-source עם קהילה פעילה. לכן, יש תמיכה למערכת וגם ניתן לתחזק אותה באמצעות מהנדסים מתחומים שונים. מערכת ROS כוללת, בנוסף ליכולת להתממשק עם חיישנים שונים ולקבל מהם מידע בעזרת כלי אחד, גם פורמט תיעוד הנקרא Rosbag. פורמט זה שימש כבסיס למערכת ההקלטות המסונכרנת שמימשנו בפרויקט זה. לאחר שהבנו איך עובדים עם מערכת ROS1, ביצענו את ההתאמות הנדרשות על מנת שנוכל להתקין את החיישנים השונים גם ב-ROS2. לבסוף, הוקם GUI בכל אחת מהמערכות (ROS1 ו-ROS2) שמאפשר לצפות בחיישנים השונים. מכיוון שה-GUI במערכת ROS1 מאפשר התקנת תוספים (Plugins) שמאפשרים לבצע הקלטות בצורה נוחה, מימשנו את מערכת ההקלטות דרך ה-GUI של מערכת ROS1.

1.4 – השוואה לעבודה קודמת

עבודה קודמת שבוצעה בנושא הייתה פרויקט עבר במסגרתו פותחה מערכת הקלטות לחיישנים שהיו קיימים דאז. במהלך העבודה על הפרויקט, נעזרנו במערכת הקודמת שפותחה, אולם היו מספר בעיות שבגללן לא היינו יכולים להסתמך על מערכת זו, אלא לעבוד על התוצרים שלנו מהיסוד. קודם כל, המערכת בפרויקט הקודם נמצאת במערכת הפעלה Ubuntu 18.04 והתבססה על ROS1 ולעומת זאת, מערכת ההפעלה שהתקנו בפרויקט הנוכחי היא Ubuntu 20.04 על מנת שנוכל לעבוד עם מערכת ROS2 שהיא שדרוג ממערכת ROS1. כמו כן, ישנם חיישנים שהתווספו בפרויקט הנוכחי כמו ה-LIDAR של Innoviz ומצלמת ה-IR. יתר על כן, בעבודה עם ROS1 במערכת ההפעלה Ubuntu 20.04 השתמשנו בהפצת ROS שנקראת Noetic, ולעומת זאת במערכת ההפעלה Ubuntu 18.04 משתמשים בהפצת ROS שנקראת Melodic. ישנם הבדלים בקוד כאשר עוברים מהפצה להפצה על אף שמדובר באותה גרסה של ROS (ROS1). למשל, כאשר ניסינו להתקין את הדרייבר של הרדאר ב-ROS1 במערכת שלנו, לא היינו יכולים להיעזר במה שנעשה בפרויקט הקודם מכיוון שהקוד לא פעל כהלכה, לכן היינו צריכים להתקין דרייבר חלופי לרדאר, בהתבסס על חבילת ROS1 לרדאר ב-GitHub, על מנת שנוכל לצפות בנתוני המכ"ם במערכת שלנו.

חלק 2 – רקע תיאורטי:

בחלק זה אנו נסביר בעיקר על העבודה עם מערכת ROS ומושגים בסיסיים במערכת זו מכיוון שזו המערכת העיקרית בה השתמשנו בפרויקט זה. כמו כן, נסביר על ההבדלים בין ROS1 ל-ROS2 . יתר על כן, יפורטו מספר פרוטוקולים שעבדנו איתם בפרויקט זה.

2.1 – מה זה ROS?

איור 2 - הסמלים של ROS ו - ROS2

ROS, ראשי תיבות של Robot Operating System, היא מערכת קוד פתוח ומספקת עבור החיישן/הרובוט את השירותים שמצפים ממערכת הפעלה, כגון: הפשטת חומרה (hardware abstraction) כך שהמפתחים דואגים רק לתוכנה, בקרת מכשירים ברמה נמוכה, הטמעת פונקציונליות נפוצה, העברת הודעות בין תהליכים, ניהול ותחזוקת חבילות. ROS מספקת כלים וספריות להשגה, בנייה, כתיבה והרצה של קוד על פני מספר מחשבים. ROS דומה במובנים מסוימים ל'robot frameworks', כמו Player, YARP, Orocos, CARMEN, Orca, MOOS ו-Microsoft Robotics Studio. ROS מיישמת מספר סגנונות שונים של תקשורת, כולל תקשורת סינכרונית בסגנון RPC על שירותים, הזרמת נתונים אסינכרונית על פני נושאים (topics) ואחסון נתונים על שרת פרמטרים. בפרויקט שלנו, מערכת ההקלטות המסונכרנת נבנתה על בסיס ROS, עקב האפשרות להתממשק עם חיישנים שונים שמשתמשים בפרוטוקולי תקשורת שונים דרך אותה פלטפורמה (ROS). הספריות בROS מיושמות ב-C++ וב-Python.

2.2 – מטרות ROS

המטרה העיקרית של ROS היא לתמוך בשימוש חוזר בקוד, במחקר ופיתוח רובוטיקה. ROS היא מסגרת מבוזרת של תהליכים (המכונים גם Nodes) המאפשרת לתכנן קובצי הפעלה באופן אינדיבידואלי ולצמוד את התהליכים בזמן ריצה. ניתן לקבץ תהליכים אלה לחבילות (packages) ולערימות, אותן ניתן לשתף ולהפיץ בקלות. ROS תומכת גם במערכת מאוחדת של מאגרי קוד המאפשרים להפיץ גם שיתופי פעולה שבוצעו. עיצוב זה, מרמת מערכת הקבצים ועד לרמת הקהילה, מאפשר קבלת החלטות עצמאיות לגבי פיתוח והטמעה. מטרה נוספת של ROS היא התאמה למערכות זמן ריצה גדולות ולתהליכי פיתוח גדולים.

2.3 – מערכת הפעלה

ROS מוכוונת למערכות דמויות UNIX, כמו לינוקס (ברוב ההפצות, כאשר התמיכה הטובה ביותר היא ב-Ubuntu), אך ניתן להשתמש בה גם במערכות הפעלה אחרות כמו Microsoft Windows (ב-ROS2).

בפרויקט שלנו העבודה מבוצעת בסביבת Ubuntu 20.04 שזו הסביבה ה"טבעית" לROS2, וגם ל-ROS1 בהפצת Noetic.

2.4 – יישומי ROS

ROS היא בפיתוח מתמיד, ובכל פעם אפשר להוסיף אליה יישומים נוספים בתחום הבינה המלאכותית והרובוטיקה, כך שבכל פעם היא תעשה את עבודתה טוב יותר.

יישומים נפוצים שבשבילם משתמשים ב-ROS:

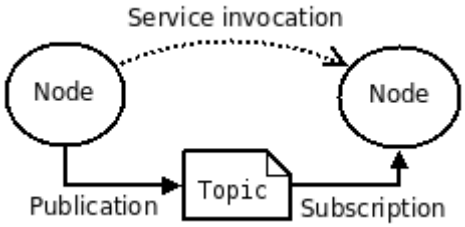
- הצגת הודעות שמתקבלות מסנסורים.
- מערכות תפיסה מלאכותית.
- זיהוי חפצים וראייה מלאכותית (יישום נפוץ בהקשרי רכב אוטונומי).
- זיהוי פנים, זיהוי מחוות וכו'.
- מעקב אחר אובייקטים.
- בדיקת מרחק חזותית.
- הבנה של תנועות.
- ראיית סטריאו.

2.5 – מושגים בסיסיים ב-ROS [1]

- **חבילות (Packages)** - חבילות הן היחידה העיקרית לארגון תוכנה ב-ROS. חבילה עשויה להכיל תהליכי ריצה של ROS (Nodes), ספרייה תלוית ROS, מערכי נתונים, קבצי תצורה או כל דבר אחר שמאורגן בצורה שימושית יחד על מנת לשרת מטרה יחידה, כמו תקשורת עם חיישן. חבילות הן פריט הבנייה היסודי ביותר ב-ROS. כלומר, הדבר הכי פרטני שאפשר לבנות ולשחרר הוא חבילה.
- **הודעות (messages)** - תיאורי הודעות, המאוחסנים ב-my_package/msg/MyMessageType.msg, מגדירים את מבני הנתונים עבור הודעות שנשלחות ב-ROS. זו הדרך העיקרית בה הnode מחליף מידע עם nodes אחרים. ניתן לומר שהודעה היא פשוט מבנה נתונים, הכולל שדות טקסט.
- **שירותים (services)** - תיאורי שירות, המאוחסנים ב-my_package/srv/MyServiceType.srv, מגדירים את מבנה נתוני הבקשות והתגובה עבור שירותים ב-ROS. מדובר בדרך נוספת בה הnode מחליף מידע עם nodes אחרים. דרך תקשורת זאת מורכבת מ- 2 הודעות: הודעת "בקשה" והודעת "תשובה". נשלחת בקשה ואז ממתינים לתשובה מהnode שנשלחה אליו הבקשה.

- **צמתים (nodes)** - nodes הם תהליכים שמבצעים חישוב. ROS תוכננה להיות מודולרית. מערכת בקרה של רובוט כוללת בדרך כלל nodes רבים: node אחד שולט על מד טווח לייזר, node אחד שולט במנועי הגלגל, node אחד מבצע לוקליזציה, node אחד מבצע תכנון נתיב, node אחד מספק תצוגה גרפית של המערכת, וכן הלאה. תהליך ROS node נכתב עם שימוש בספריית לקוח של ROS, כגון roscpp או rospy. בפרויקט שלנו, אנו השתמשנו בnodes שמגיעים עם package של כל חיישן.
- **מאסטר (master)** - ה-ROS Master מספק רישום של nodes פועלים במערכת. ללא המאסטר, nodes לא יוכלו למצוא זה את זה, להחליף הודעות או להפעיל שירותים.
- **נושאים (topics)** - הודעות מנותבות באמצעות מערכת תחבורה עם סמנטיקה של publish/subscribe. Node שולח מידע באמצעות ביצוע publish ל-topic נתון. topic הוא שם המשמש לזיהוי תוכן ההודעה. Node שמתעניין בסוג מסוים של נתונים יבצע subscribe ל-topic המתאים.
- **Bag –** מדובר בפורמט לשמירה והפעלה של נתוני הודעות ROS. Bags הם מנגנון חשוב לאחסון נתונים, כגון נתוני חיישנים, שיכולים להיות קשים לאיסוף אך הם הכרחיים לפיתוח ובדיקת אלגוריתמים. בפרויקט זה ביצענו שימוש רחב היקף ב-rosbag על מנת להקליט הודעות שמתקבלות מחיישנים שונים ולאחר מכן להפעיל הודעות אלו בזמן שמתאים לנו.

נצרף איור שממחיש את המושגים עליהם פירטנו בסעיף זה:



איור 3 - שליחת הודעה בין שני nodes

כפי שניתן לראות מאיור 4, ה-node הימני רוצה להאזין ל-topic מסוים של ה-node השמאלי. לכן, הnode השמאלי צריך לשלוח את ההודעה באמצעות ביצוע publish ל-topic המבוקש. כמו כן, הnode הימני צריך לבצע subscribe לאותו ה-topic. כל התהליך מתחיל להתבצע לאחר שמתקבלת הודעת "בקשה" מהnode הימני.

2.6 – ROS2 וההבדלים בינה לבין ROS1

מאז שROS1 נוצרה והופצה בשנת 2007 היא צברה פופולריות רבה בקהילת הרובוטיקה והפכה לפלטפורמה לביצוע מחקר וניסויים. עם זאת, ROS1 לא נבנתה מתוך מחשבה של

שימוש מסחרי. לכן, דברים חשובים כמו אבטחה, טופולוגיית רשת וזמן פעילות של המערכת לא קיבלו עדיפות. מכיוון שהרבה חברות החלו להשתמש ב-ROS, הפגמים העיקריים שלה הפכו ברורים יותר ויותר. לפיכך, היה צורך לבנות מחדש את ROS מהיסוד מתוך מחשבה על שימוש מסחרי, כלומר ROS2.

ROS2 נבנתה מהיסוד עם הדרישות המרכזיות הבאות על מנת לטפל בבעיות שהועלו תוך שימוש ב-ROS1:

- **אבטחה** – השימוש במערכת צריך להיות מאובטח עם הצפנה מתאימה כשנדרש.
- **מערכות משובצות מחשב** - ROS2 צריכה להיות מסוגלת לפעול על מערכות משובצות מחשב.
- **חישובי real time** – יש צורך להיות מסוגלים לבצע חישוב בזמן אמת בצורה מהימנה שכן יעילות זמן הריצה היא חיונית ברובוטיקה.

כדי לעמוד בדרישות הנ"ל, מפתחי ROS2 החליטו לעבור לשימוש בDDS כפרוטוקול הרשת לכל התקשורת המתרחשת פנימית ב-ROS2. DDS מציין data distribution service ונמצא בשימוש נרחב בתשתיות קריטיות כגון: ספינות קרב, מערכות חלל וכו'. פרוטוקול זה מספק את ערבויות האבטחה, כמו גם את האמינות הדרושה לשמירה על תקשורת טובה באזורים עם חיבורים חלשים (Wi-Fi או קישוריות לווין). זאת, בניגוד לפרוטוקול המותאם אישית TCP/UDP שפותח בROS1 ופספס הרבה מהאמינות והאבטחה שמסופקים באמצעות DDS.

ההבדלים העיקריים בין ROS2 ל-ROS1 [2]:

- **ROS2 ו-DDS מספקות ערבויות אבטחה, ROS1 לא –** בהרבה תרחישים, מערכות רובוטיות צריכות לתקשר ברשתות לא מאובטחות בין אם זה באמצעות topics, services או פעולות של ROS. מדובר בדאגה עצומה ליישומים מסחריים קריטיים שROS1 לא מטפל בה. לכן, ההחלטה לעבור לDDS ב – ROS2 גרמה לכל חששות האבטחה להיעלם.
- **ROS1 צריכה ROS master בשביל תקשורת בין nodes שונים, ROS2 לא –** ב-ROS1 ה-ROS Master מספק רישום של nodes פועלים במערכת. ללא המאסטר, nodes לא יוכלו למצוא זה את זה, להחליף הודעות או להפעיל שירותים. לעומת זאת, ב-ROS2 לא צריך master מכיוון ש-ROS2 משתמשת ב-DDS, שמאפשר ל-nodes לתקשר זה עם זה ללא צורך במתווך. ה-DDS מאפשר לכל node שעולה ברשת למצוא node אחר מבלי להזדקק למתווך.
- **ROS2 מתפקדת טוב יותר מROS1 במצבי רשת חלשה ותקיעות רשת –** ROS1 בנויה באמצעות פרוטוקול TCP ולכן צריכה מצבי רשת אמינים בשביל שידורים

חוזרים של נתונים. מכיוון ש-ROS2 משתמשת ב-DDS, אין צורך בשידורים חוזרים של נתונים ולכן מתפקדת טוב יותר במצבי רשת לא אמינים.

- **ל-ROS2 קיימת תמיכה מרובת פלטפורמות –** ROS1 נתמכת רק בלינוקס, לעומת זאת ROS2 נתמכת בלינוקס, Windows ו-MacOS. כמו כן, קל יותר לשלב את ROS2 עם משאבי ענן כמו AWS.
- **ל-ROS2 חישובי זמן אמת מהימנים יותר מ-ROS1 –** בשונה מ-ROS1, מפתחי ROS2 שמו דגש על חישובי זמן אמת בצורה מהימנה מכיוון שיעילות זמן הריצה היא חיונית ברובוטיקה. ROS2 יוצרה מתוך מחשבה על שימוש מסחרי ולכן היה צורך לשפר את חישובי זמן האמת ממערכת ROS1.
- **תמיכה בגרסאות השונות –** התמיכה ב-ROS1 עתידה להסתיים בחודש מאי של שנת 2025 ולכן ישנו הצורך בתחילת העבודה עם ROS2 שעתידה להישאר לאחר סיום התמיכה ב-ROS1.

בנוסף לכל היתרונות הנ"ל, יש ב-ROS2 אפשרות של תאימות לאחור עם ROS1. אם כבר נבנתה אפליקציה כלשהי ב-ROS1 וזה לא אפשרי להחליף את כל האפליקציה מ-ROS1 ל-ROS2 אז מפתחי ROS2 פתרו בעיה זו באמצעות node שנקרא ROS1 bridge. לכן, ניתן להעביר את הקוד ופעולת החיישנים מ-ROS1 ל-ROS2 (וגם להפך) בלי לשבור את התקשורת בין החלקים השונים של המערכת.

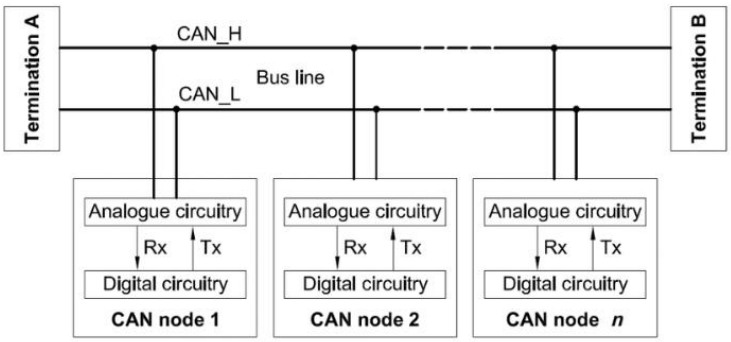
בפרויקט שלנו ביצענו שימוש רב ב-ROS bridge על מנת להפעיל את החיישנים השונים ב-ROS2. (יפורט בהרחבה בפרק הבא – מימוש)

2.7 – פרוטוקולים שונים

בפרויקט ישנם מספר פרוטוקולים שנאלצנו לעבוד איתם בשביל לתקשר עם החיישנים השונים:

- **Can bus** [3] **–** can זה ראשי תיבות של controller area network, מכונה לעיתים גם CAN network. זהו פרוטוקול תקשורת מתוקנן שתוכנן לאפשר למיקרו בקרים והתקנים אחרים לתקשר האחד עם השני. במקורו, הוא יועד לשמש במערכות רכב אך משמש כיום גם בתחומים אחרים. מדובר ב-bus מסוג שידור, המשמעות היא שכל ה-nodes יכולים "להקשיב" לכל השידורים מכיוון שאין דרך לשלוח הודעה רק ל-node ספציפי. כל ה-nodes תמיד יאספו את כל תעבורת השידור. לעומת זאת, החומרה של ה-can bus מספקת סינון מקומי כך שכל צומת עשוי להגיב רק להודעות שמעניינות אותו. כפי שניתן להבין מכאן, התקשורת על ה-can bus מתבצעת בפורמט של שליחת הודעות. במערכת שלנו יש שימוש ב- Kvaser USB to CAN על מנת לחבר את המחשב לרשת ה-Can bus . כמו כן, ה-Kvaser מתפקד כ-controller בכך שהוא בודק לאיזו הודעה מגיע לעלות על ה-Bus . נציין

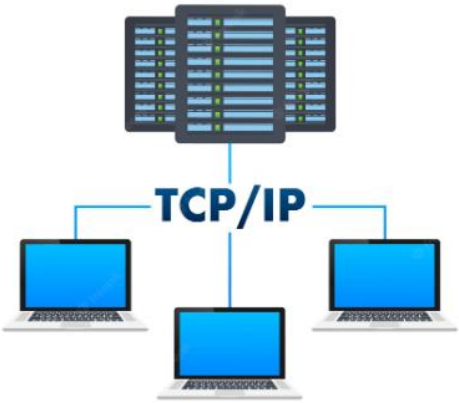
שהרדאר ברכב מחובר ל-Kvaser בסטנדרט ISO 11898-2 שזו תצורת high speed .



איור 4 - can bus structure ISO 11898-2 [4]

כפי שניתן לראות מאיור 5, הפרוטוקול מורכב מ-bus מרכזי שמכיל שני חוטים הנקראים CAN_H ו-CAN_L שמאפשרים מעבר של הודעות למספר nodes הממוקמים על ה-Bus. הסיבה להעברת המידע בתצורה זו היא לצורך הקטנת רעשים ושגיאות.

- **TCP/IP** – [5] מדובר בחבילת פרוטוקולי תקשורת עליה מושתתת רשת האינטרנט. פרוטוקול זה הוא שם למודל שכבתי המתאר תקשורת ברשתות מחשבים. הפרוטוקול מאפשר תקשורת במודל של client/server כך שמחשב המשתמש (client) מבקש ומקבל שירות (כמו שליחת דפי Web) ממחשב אחר (server) ברשת. TCP/IP היא תוכנית בעלת 2 שכבות. השכבה העליונה, פרוטוקול בקרת שליחה (TCP), מנהלת את חלוקת הקובץ שהוא ההודעה ל-packets קטנים שנשלחים ברשת האינטרנט ואוספת מחדש את ה-packets עד לקבלת ההודעה המקורית. השכבה התחתונה, פרוטוקול אינטרנט (IP), מטפלת בכתובת אליה נשלח כל packet כדי שיגיע ליעד הנכון. כל מחשב gateway ברשת בודק כתובת זו כדי לדעת לאן לשלוח את ההודעה. אפילו שחלק מה-packets של אותה הודעה מנותבים באופן שונה, הם יקובצו מחדש ביעד. לכל רשת שאינה סגורה יש כתובת gateway שדרכה ניתן להתחבר לרשת חיצונית ואפילו לאינטרנט. אך כדי לבצע ניתובים אלו בהצלחה יש לדעת להגדיר טבלת ניתובים אשר מגדירה מה הן כתובת תת- רשת וכיצד להגיע ל-gateway אם רוצים כתובת שלא תואמת את התת רשת. בפרויקט שלנו, המצלמה ושני ה-LIDARs מתקשרים עם המחשב דרך נתבים בפרוטוקול זה.



איור 5 - תקשורת TCP/IP

רואים מאיור 5 שישנם מספר מחשבים שמתקשרים ביניהם באמצעות פרוטוקול TCP/IP עליו הסברנו.

- **RS-232** [6] **–** הוא תקן תקשורת להעברה טורית של מידע בינארי בין מכשיר המשמש כמקור המידע (Data Terminal Equipment, DTE) ומכשיר המשמש כקולט המידע (Data Circuit Equipment, DCE). קצב העברת הנתונים קובע כמה מהר לוקח למידע לעבור בקו ונמדד בביט לשנייה. חיישן הINS ברכב מחובר בחיבור המשתמש בתקן זה. קצב העברת המידע בו הינו 115,200 ביטים לשנייה ויש 8 ביטי מידע.

איור 6 - כבל להעברת תקשורת בפרוטוקול RS-232

באיור 7 רואים כבל להעברת תקשורת בפרוטוקול זה.

חלק 3 – מימוש

3.1 – התשתית ותיאור החומרה

לצורך מימוש הפרויקט השתמשנו ברכב נוסעים עליו מותקנים החיישנים/סנסורים אותם אנו מכניסים למערכת.

איור 7 - תמונה של רכב הפרויקט

המערכת כוללת את החיישנים הבאים:

- **חיישן LIDAR מסוג Velodyne VLP16** (יצרן: Velodyne) [7]
  החיישן מתחבר למחשב באמצעות Ethernet ובתקשורת TCP/IP. לחיישן יש דרייבר ל-ROS2 שמסופק על ידי היצרן, ובו השתמשנו על מנת לקבל ממנו נתונים. [8]
- **חיישן LIDAR מסוג InnovizeOne** (יצרן: Innoviz) [9]
  גם חיישן זה מתחבר למחשב באמצעות Ethernet ובתקשורת TCP/IP. היצרן מספק דרייבר וחבילת Nodes עבור ROS1, ובהם השתמשנו כדי לקבל ממנו נתונים. [10]
- **מצלמת אור נראה (RGB) מסוג Prosilica GT1930c** (יצרן: Allied Vision) [11]
  המצלמה מתחברת למחשב באמצעות Ethernet ובתקשורת TCP/IP. על מנת

לחבר את המצלמה למערכת היה עלינו להתקין דרייבר מאתר היצרן [12] ולהשתמש בחבילות ה-ROS1 שהיצרן מספק: SDK [13] וחבילת ROS [14].

- **מצלמה תרמית מסוג Boson FLIR** (יצרן: FLIR) [15]
המצלמה מתחברת למחשב באמצעות USB, ועל מנת להציג את המידע השתמשנו בחבילת ROS1 שנוצרה על ידי AutonomusStuff. החבילה מבוססת על חבילת דוגמה שסופקה על ידי יצרן המצלמה. [16]
- **מכ"ם פגוש מסוג Delphi ESR 2.5** (יצרן: AutomomousStuff) [17]
המכ"ם מתחבר למחשב דרך מתאם Kvaser USB, הממיר בין פרוטוקול CAN (CANBUS, שבו הרדאר משדר) ל-USB (שבו המחשב קולט). על מנת לקבל את נתוני הרדאר השתמשנו בתוכנת can-utils ובחבילת ROS1 ייעודית עבורו. [18]
- **חיישן INS מסוג INS-DL** (יצרן: InertialLabs) [19]
ה-INS מחובר למחשב דרך serial port-RS232. על מנת לקבל את הנתונים ב-ROS השתמשנו בחבילת ROS1 ייעודית של היצרן. [20]

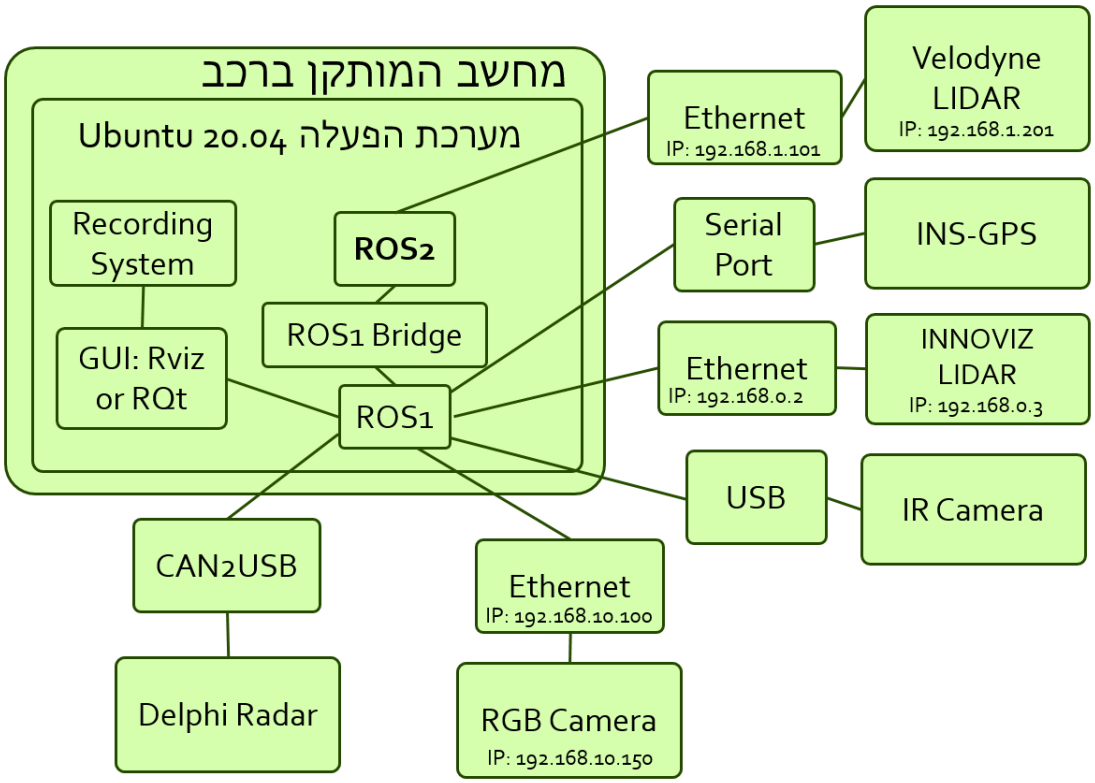
איור 8 - תמונות החיישנים השונים המותקנים ברכב

המערכת כולה רצה על מחשב PC (x86) המותקן ברכב. על המחשב הותקנה מערכת ההפעלה Ubuntu 20.04, ועליה הותקנו שתי גרסאות של ROS:

- ROS1 Noetic – גרסה זו משמשת לתקשורת עם רוב החיישנים.
- ROS2 Foxy – גרסה זו משמשת לתקשורת עם ה-LIDAR של Velodyne.

הרדאר של חברת Delphi ממוקם בפגוש הקדמי של הרכב. על גג הרכב מותקנות המצלמות וגם LIDARs של חברת Velodyne ו-Innoviz. התשתית החומרתית של המערכת מותקנת בבגאז' של הרכב, וכוללת מחשב המחובר בבגאז' ומתאמים לחיבור החיישנים השונים. במושב האחורי של הרכב נמצאים מסך, מקלדת ועכבר, ושם ישבנו במהלך העבודה עם המחשב של הרכב. למחשב יש מסך נוסף בבגאז' של הרכב, שלא השתמשנו בו מטעמי נוחות.

כך נראית דיאגרמת הבלוקים של המערכת:



איור 9 - דיאגרמת בלוקים מפורטת של המערכת

טבלת הניתוב עבור החיישנים המחוברים ב-Ethernet

Address	Netmask	Switch	Comments
192.168.1.101	255.255.255.0	Switch 1 – enp0s31f6	Gateway for Velodyne LIDAR
192.168.0.30	255.255.255.0	Switch 1 – enp0s31f6	
192.168.0.2	255.255.255.0	Switch 1 – enp0s31f6	Gateway for Innoviz LIDAR
192.168.10.100	255.255.255.0	Switch 2 – enp1s0	Gateway for RGB Camera
192.168.1.100	255.255.255.0	Switch 2 – enp1s0	Gateway for switch 1
192.168.1.201	255.255.255.0	Switch 1	Velodyne LIDAR
192.168.10.150	255.255.255.0	Switch 2	RGB Camera
192.168.0.3	255.255.255.0	Switch 1	Innoviz LIDAR

טבלה 1 - ניתוב עבור החיישנים המחוברים ב-Ethernet

3.2 – תיאור התוכנה

השימוש ב-ROS במערכת ההקלטות הוא בכל רכיבי המערכת: לשם תקשורת עם החיישנים, לשם הצגת המידע למשתמש ולצורך ההקלטה.

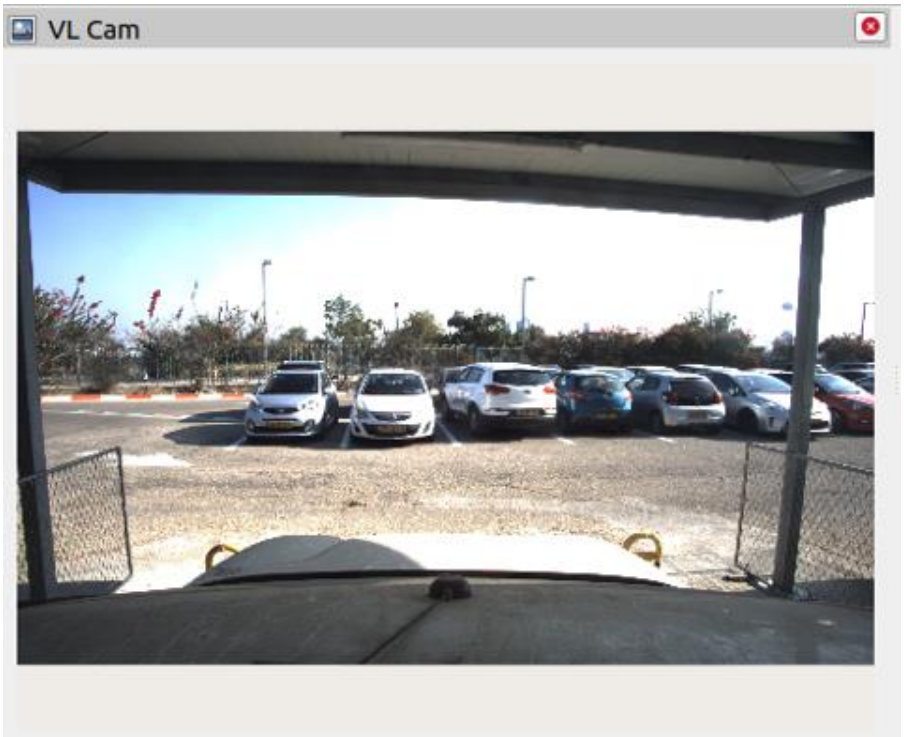
התקשורת עם החיישנים מתבצעת בעזרת חבילות ROS ייעודיות להם, שברוב המקרים סופקו ע"י היצרנים. עבור רוב החיישנים אין תמיכה כעת ב-ROS2, ולכן השתמשנו עבורם בחבילות המיועדות ל-ROS1, ובנוסף ב-ROS1 Bridge על מנת לאפשר צפייה ב-topics של ROS1 ב-ROS2 ולהיפך.

ה-Nodes וה-Topics המשמשים במערכת עבור כל אחד מהחיישנים:

<u>מצלמת RGB:</u>

על מנת לקבל אות מהמצלמה יש להתקין את הדרייבר שלה (Vimba), להוסיף ניתוב מתאים ב-Switch שאליו מחוברת המצלמה ולהתקין את חבילת ה-ROS הקיימת עבורה. ה-Node של המצלמה הוא camera_driver/ והמערכת שלנו משתמשת ב-topic בשם image_raw/ המפורסם על ידיו. Topic זה מכיל מידע גולמי מחיישן המצלמה, לפני תהליכי Demosaic.

את התמונה ניתן לראות באמצעות Rviz או באמצעות RQt Image View:



איור 10 - תמונה ממצלמת הRGB

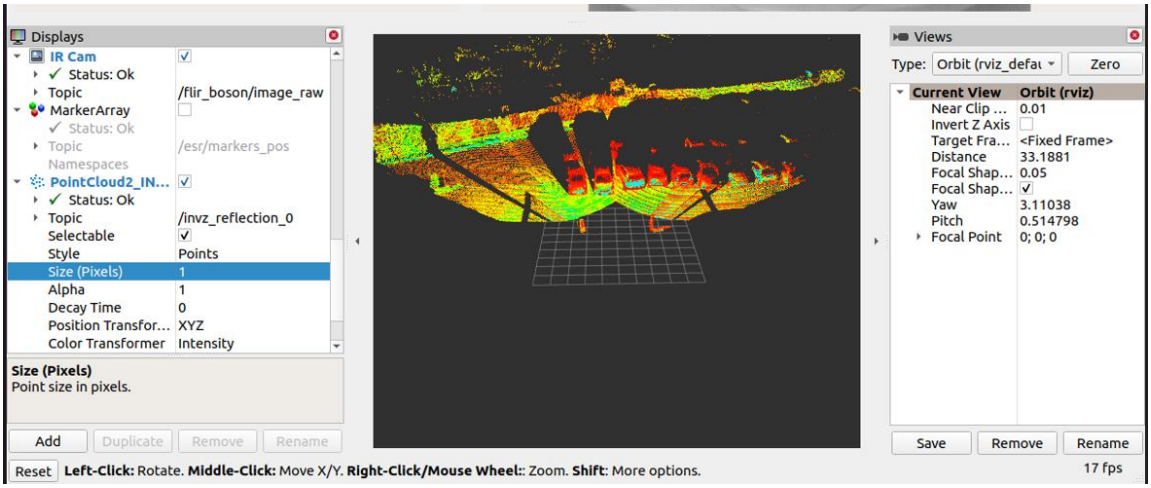
מצלמת IR:

על מנת להשתמש במצלמת ה-IR יש להתקין את חבילת ה-ROS שלה. ה-Node של מצלמת ה-IR הוא /flir_boson/flir_boson_usb_mode והוא משדר על topic שנקרא /flir_boson/image_raw, המכיל מידע גולמי מחיישן המצלמה. את התמונה ניתן לפתוח באמצעות Rviz או RQt Image View:

איור 11 - תמונה ממצלמת ה-IR

Innoviz LIDAR:

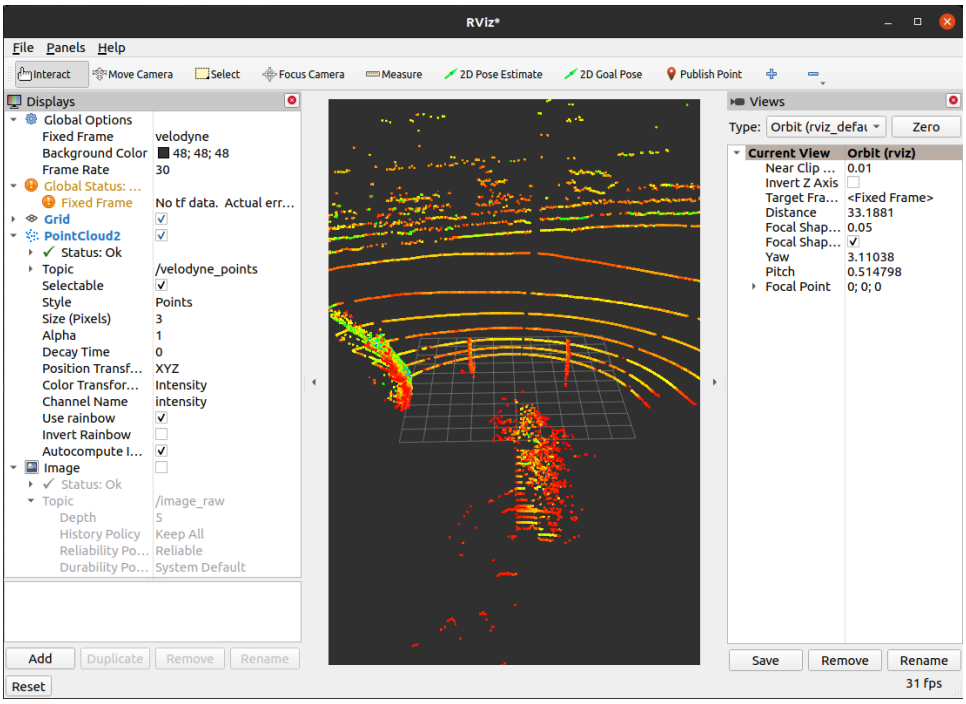
על מנת להשתמש ב-LIDAR של Innoviz הוספנו ניתוב מתאים בטבלת הניתוב והתקנו את הדרייבר שלו ואת חבילת ה-ROS, שסופקו על ידי היצרן. ה-Node של ה-LIDAR של אינוביז הוא /invz_publisher. המערכת שלנו מציגה את המידע המתקבל על topic בשם /invz_reflection_0, המכיל את ענן הנקודות של ה-LIDAR. את ענן הנקודות ניתן לראות באמצעות Rviz:



איור 12 - תצוגת המידע מה-LIDAR של אינוביז

Velodyne VLP-16:

על מנת להשתמש ב-LIDAR של Velodyne יש להתקין את חבילת ה-ROS2 שלו ולהגדיר את מתאם הרשת שאליו הוא מחובר ככתוב בטבלת הניתובים. ה-LIDAR של Velodyne פועל על Node בשם /velodyne ומפרסם את המידע שלו על topic בשם /velodyne_points, המכיל את ענן הנקודות של ה-LIDAR. את ענן הנקודות ניתן להציג דרך Rviz כך:



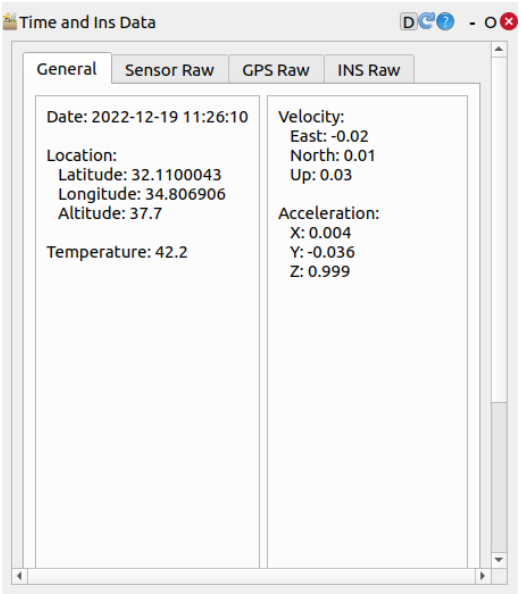
איור 13 - תצוגת המידע מה-LIDAR של Velodyne

Inertiallabs INS:

על מנת להשתמש ב-INS יש להתקין את חבילת ה-ROS המתאימה לו, ולהגדיר אותה כך שתחפש את ה-INS ב-/dev/ttyS3 – כתובת ה-port שאליו מחובר ה-INS. ה-INS פועל דרך Node בשם /INS ומפרסם את המידע שלו על שלושה Topics:

- /Inertial_Labs/ins_data – כל המידע שמתקבל על ידי ה-INS.
- /Inertial_Labs/gps_data – מידע על קואורדינטות GPS ומידע הנגזר מהם בלבד.
- /Inertial_Labs/sensor_data – מידע המתקבל מכל החיישנים ב-INS למעט ה-GPS.

התוסף לממשק הגרפי של המערכת (בתמונה הבאה) מציג מידע המתקבל על כל שלושת ה-topics.



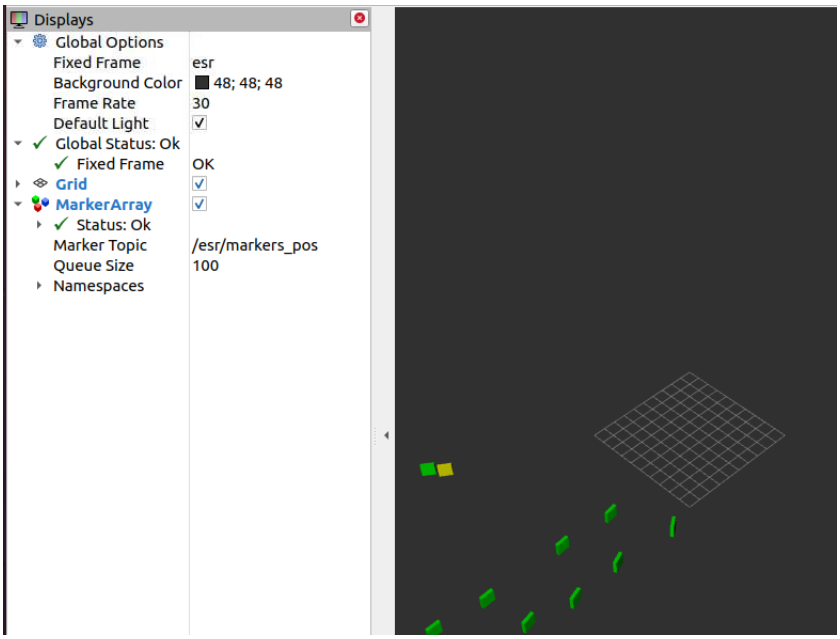
איור 14 - ה-Plugin שמציג את הנתונים מה-INS-GPS

Delphi ESR Radar:

על מנת להשתמש במכ"מ יש להתקין את תוכנת can-utils ואת חבילת ה-ROS הקיימת עבורו. המכ"ם פועל דרך Node שנקרא /esr/esr_tracks_can ומפרסם שלושה Topics עם מידע:

- /esr/markers_pos – קואורדינטות מיקום
- /esr/markers_speed – מידע על מהירויות של עצמים המזוהים על ידי המכ"ם
- /esr/markers_accel – מידע על תאוצות של עצמים שמזוהים על ידי המכ"ם

המידע המשמש אותנו בפרויקט הוא מידע המיקום, ואותו אנחנו מציגים, באמצעות Rviz.



איור 15 - תצוגת המידע מה-RADAR של דלפי

ROS1 Bridge

נוסף על התקנת ROS1 ו ROS2 , ועל מנת שנוכל להציג מידע מהחיישנים בו זמנית, יש צורך להיות מסוגלים לראות את כל זרמי המידע (ה-topics) באותה גרסה של ROS. לשם כך השתמשנו בחבילת ROS2 הנקראת ROS1 Bridge, שתפקידה להעביר topics בשני הכיוונים: גם מ-ROS1 ל-ROS2 וגם בכיוון ההפוך, על פי טבלת מיפויים. כברירת מחדל, ה-Bridge מעביר topics של הודעות מסוגים נפוצים – למשל תמונה, ענן נקודות או סמנים (Marker Array). כל topic שמזוהה על ידי ה-bridge ניתן לראות בשתי הגרסאות של ROS, תחת אותו השם, וה-Node המפרסם אותם (בגרסה שבה אינם מתפרסמים על ידי Node שרץ כ-native) נקרא ros_bridge/. על מנת שה-Bridge יוכל להעביר הודעות שבהן אינו מכיר במצב ברירת המחדל שלו, יש להוסיף לקוד המקור שלו מיפויים מתאימים ולהדר אותו מחדש – תהליך מורכב שהוא מחוץ למסגרת הפרויקט.

ה-INS משתמש בפורמט ייחודי להודעות שלו, שאינו קיים כברירת מחדל בטבלת המיפויים של ה-Bridge, ולכן הוחלט לממש את המערכת רובה ב-ROS1 ולהעביר דרך ה-Bridge את החיישנים שפועלים ב-ROS2. עם זאת, עדיין ניתן להציג את המידע מכל החיישנים האחרים גם ב-ROS2 בעזרת Rviz2, כפי שיפורט בהמשך.

לשם העלאת ה-Bridge בפקודה בודדת, כתבנו סקריפט bash שמאגד בתוכו את כל הפקודות הנדרשות לפתיחת ROS1 Bridge בטרמינל (כאשר כבר יש ROS1 Master פעיל).

הצגת המידע מכל החיישנים יחד

הצגת המידע מתבצעת בשתי דרכים:

- באמצעות Rviz 2 (ROS2): Rviz2 הוא חבילה (ROS2) להצגת מידע מחיישנים שונים בצורה נוחה. ניתן להציג בו קלט וידאו ממצלמה ונקודות המתקבלות מ-Lidar או ממכ"ם, אך לא מה-INS.
- באמצעות RQt GUI: RQt GUI היא חבילת ROS גמישה ביותר המאפשרת לבנות ממשק משתמש גרפי, וניתן להוסיף לה תוספים (Plugins) שונים, שביניהם תוספים שיכולים להציג מידע. זו השיטה העיקרית שבה השתמשנו לבניית מערכת ההקלטות המסונכרנת.

העלאת כל רכיבי המערכת מתבצעת באמצעות Launch File, כמפורט בסעיף הבא.

ההקלטה מתבצעת באמצעות Rosbag, מנגנון ההקלטה המובנה של ROS. המנגנון מקליט את כל ה-topics שמשדרים בעת הפעלתו, ושומר את התוצאה בקובץ עם סיומת bag. ניתן לצפות בהקלטה באמצעות מנגנון מובנה של ROS, אך לא לייצא את המידע בצורה פשוטה. לשם כך, השתמשנו בכלים נוספים המיועדים לכך, שיוסברו בהמשך.

ממשק משתמש גרפי מאוחד

על מנת לבנות את ממשק המשתמש הגרפי השתמשנו ב-RQt – חבילת ROS גמישה ביותר המבוססת על Plugins – חבילות תוכנה בעלות ממשק גרפי, שכל אחד מהם מבצע תפקיד כלשהו. RQt GUI מאפשרת למקם Plugins בצורה גמישה בחלון הראשי של הממשק הגרפי, ומגיעה באופן מובנה עם Plugins שונים: Plugin להצגת קלט וידאו ממצלמה, Plugin המאפשר פתיחה של קבצי Bag וצפייה בהם, Plugin של Rviz להצגת המידע מחיישנים כמו חיישני ה-LIDAR והמכ"ם ועוד.

ניתן לבנות Plugins ל-RQt, על ידי יצירת שני קבצים:

- קובץ תוכנה, המכיל את הפונקציות שעל ה-Plugin להריץ. ניתן לכתוב קבצים כאלה בקלות יחסית בעזרת פייתון.
- קובץ ממשק משתמש – מכיל את תיאור הממשק הגרפי (מיקומי הכפתורים, התמונות השונות והטקסטים, כמו גם קישורים פנימיים בין רכיבים בממשק). קובץ זה ניתן לעריכה, למשל, באמצעות תוכנת Qt Designer.

השליטה בהקלטות מתבצעת באמצעות Plugin המבוסס על Plugin דומה ששימש גם בפרויקט קודם והותאם לפרויקט שלנו: הרחבנו את רשימת החיישנים שה-Plugin מסוגל להקליט, הוספנו כפתורים מתאימים עבורם בממשק הגרפי והמרנו אותו מ-python 2 ל-python 3 (הגרסה הנדרשת על מנת שירוץ ב-ROS1 Noetic).

השתמשנו גם בשני Plugins נוספים:

- גרסה מעט שונה של Plugin ה-Bag playback, ששימשה גם בפרויקט הקודם: בגרסה זו, כאשר צופים בהקלטה, כל חיישן משודר על topic עם קידומת שונה ולא על אותו topic שהוקלט בקובץ.
- Plugin המציג את נתוני ה-INS בצורה גרפית, מתוך ההודעות שמתקבלות על ה-topics של ה-INS.

על מנת לפתוח את כל המערכת בפעולה בודדת, יצרנו Launch File אחוד שהרצה שלו מהטרמינל פותחת את כל רכיבי ה-ROS1 במערכת ואת ה-GUI. לשם כך העתקנו מכל Launch File של כל חיישן את רשימת ה-Nodes הנדרשים להפעלתו, והוספנו רשומה עבור Node של RQt GUI שאליה מועבר קובץ ה-Perspective של RQt, המתאר את תוכן החלון שיצרנו וניתן לשמירה מתוך הממשק הגרפי של RQt, כפרמטר.

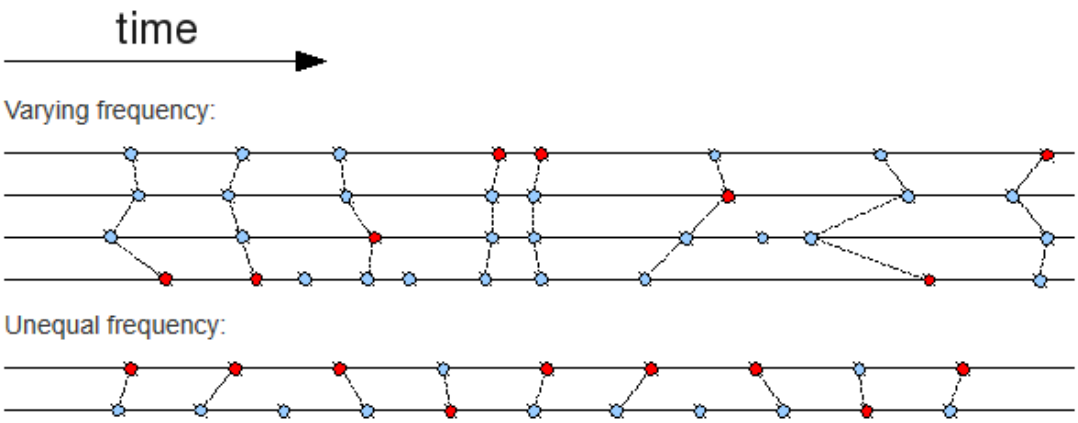
הקלטה מסונכרנת

על מנת לאפשר הקלטה מסונכרנת, השתמשנו ב-Node מתאים ה"מקשיב" ל-topics של החיישנים השונים ומשכפל הודעות נבחרות שמגיעות בהם ל-topics שהוא יוצר. בחירת ההודעות המשוכפלות היא על פי זמן ההגעה, כך שה-topics שהמסנכרן יוצר מכילים

הודעות מסונכרנות. הסנכרון ממומש על ידי יכולת סינון ההודעות של ROS (ספריית message filters), בעזרת מסנן Approximate Time Synchronizer.

האלגוריתם שבו משתמש המסנן מתואר בהרחבה בתיעוד של ROS [21]. בקצרה, כשההודעות מגיעות הן נכנסות לתור לפי סדר הגעתן, וכאשר יש הודעה בכל אחד מה-topics הרלוונטיים נבחרת ההודעה המאוחרת ביותר בתור הציר (pivot), ומכל topic נלקחת ההודעה המאוחרת ביותר שהגיעה לפניה, וכל ההודעות שלפני ההודעה הנבחרת ושייכות לאותו Topic שלה מוסרות מהתור והמערכת מתעלמת מהן.

כך זה נראה גרפית:



איור 16 - תמונה להמחשת אלגוריתם ה-Approximate Time Synchronization

ה-Pivot בכל נקודת זמן מסומן באדום. ההודעות שנבחרות בסופו של דבר מקושרות בקו.

ייצוא המידע מקבצי ה – Bag

יש שלושה סוגים עיקריים של מידע שברצוננו לייצא מתוך קבצי ה-Bag לצורך עיבוד נוסף:

- תמונות – ברצוננו לייצא אותן ל-JPEG, ואולי לאחר מכן לפורמט MP4.
- ענני נקודות – ברצוננו לייצא אותם לפורמט PCD – פורמט סטנדרטי לייצוג ענני נקודות, שניתן לעבד לאחר מכן באמצעות Matlab.
- מידע INS – ברצוננו לייצא אותו לקובץ CSV, על מנת לקרוא אותו בנוחות.
- RADAR – נרצה לייצא את המידע מהרדאר לקובץ CSV.

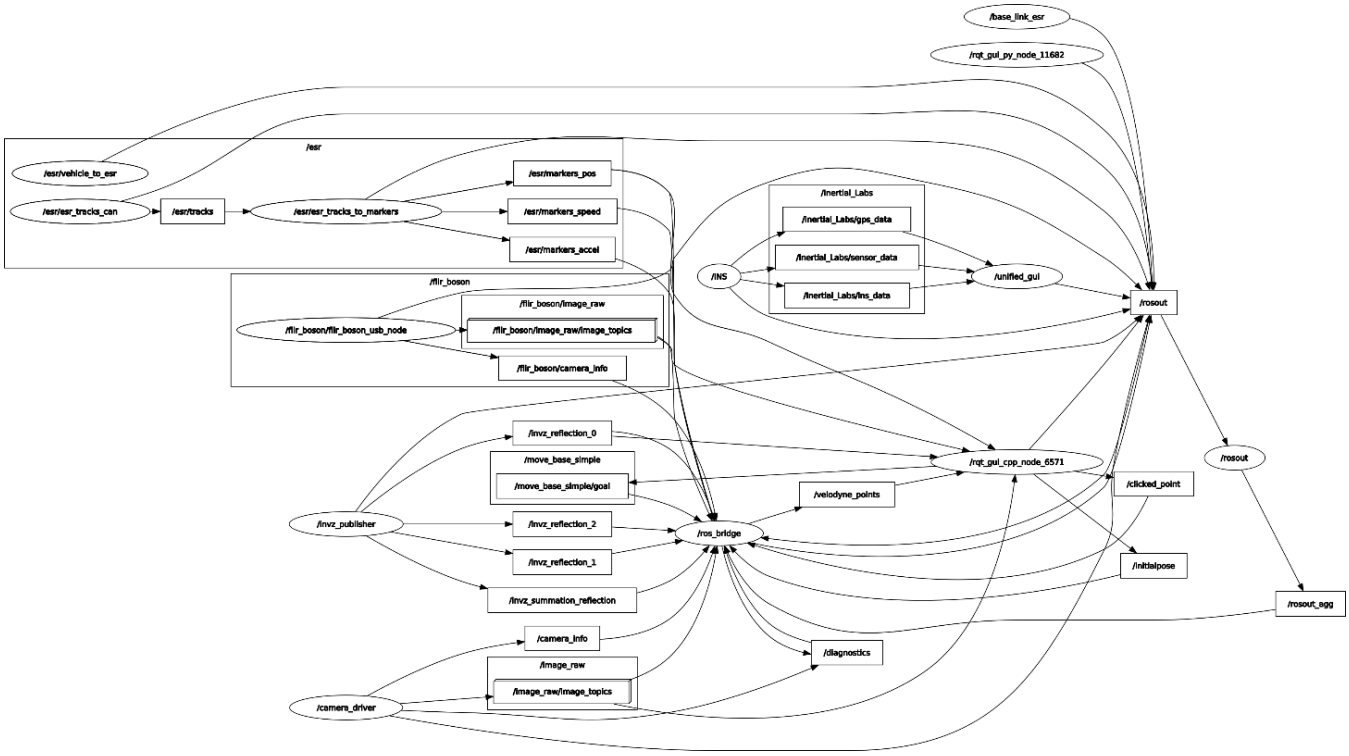
עבור ענן נקודות התשובה פשוטה – קיימת חבילה מובנית ב-ROS שמסוגלת לעשות בדיוק את זה. עבור השניים האחרים היה צורך בפתרון חיצוני:

לשם ייצוא מידע מקובץ Bag לקובץ CSV, השתמשנו בחבילת rosbag_to_csv. החבילה, הכתובה בפייתון, כוללת ממשק משתמש גרפי שבו ניתן לבחור את ה-topics שרוצים לייצא מקובץ bag מסוים, ואז בלחיצה על Convert מופק קובץ CSV המכיל את המידע בטבלה.

לשם ייצוא התמונות, השתמשנו בסקריפט פייתון שפותח את קובץ ה-Bag ומייצא topic נבחר לתיקייה כתמונות JPEG. התאמנו אותו ל-Python3 ושינינו את ההתנהגות שלו כך

שתהיה יותר נוחה לשימוש. משם ניתן להשתמש בממיר וידאו (למשל FFMpeg) על מנת להמיר את התמונות שבתיקייה לקובץ וידאו אחד מסוג mp4.

תרשים ה-Nodes וה-Topics במערכת



איור 17 - גרף ה-Nodes של המערכת

בתרשים ניתן לראות את כל ה-Nodes הרצים במערכת. צמתים בגרף המייצגים מידע שמגיע מהחיישנים השונים (כלומר topics) הם:

- /invz_reflection_0 (ה-LIDAR של Innoviz)
- /esr/markers_pos (מכ"ם פגוש)
- /Inertial_Labs/ins_data (INS)
- /image_raw (מצלמת RGB)
- /velodyne_points (LIDAR של Velodyne)
- /flir_boson/image_raw (מצלמת IR)

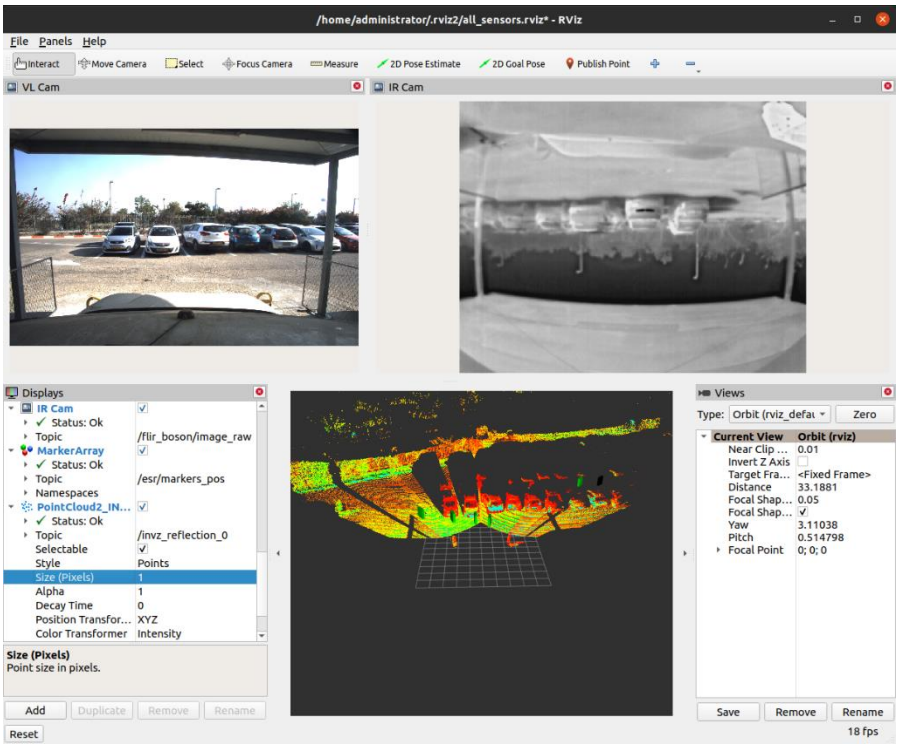
כמו כן ניתן לראות את הצומת /ros_bridge, שנכנסים אליו כל ה-topics הקיימים ויוצא ממנו ה-topic של Velodyne – זה מפני שהמערכת רצה ב-ROS1 וה-topic של Velodyne מגיע דרך ROS2.

צמתים חשובים נוספים הם הצמתים הקשורים ל-RQt GUI (עם rqt_qui בתחילת שמם), המייצגים plugins של RQt GUI.

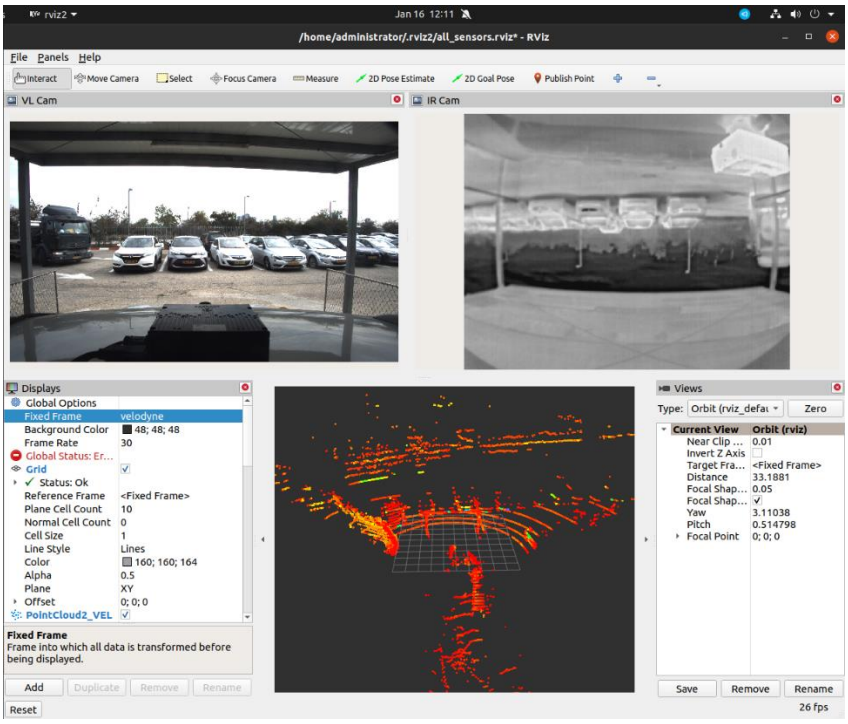
חלק 4 – התוצאות

4.1 – הצגת המידע מהחיישנים ב – ROS2

כך נראה החלון של הצגת החיישנים ב-Rviz2:



איור 18 - תצוגת החיישנים ב-Rviz2: מצלמות, רדאר, Innoviz LIDAR



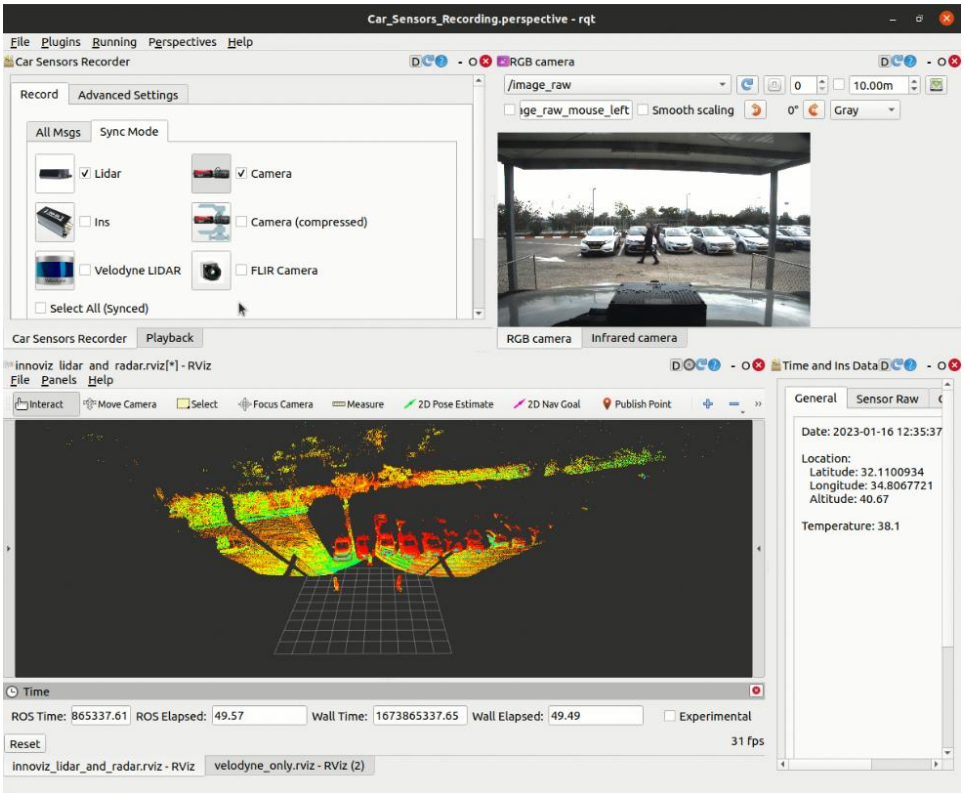
איור 19 - תצוגת החיישנים ב-Rviz2 – מצלמות, Velodyne LIDAR

בתמונות אלה ניתן לראות את התצוגה ב-Rviz2, המחולקת למספר חלוניות:

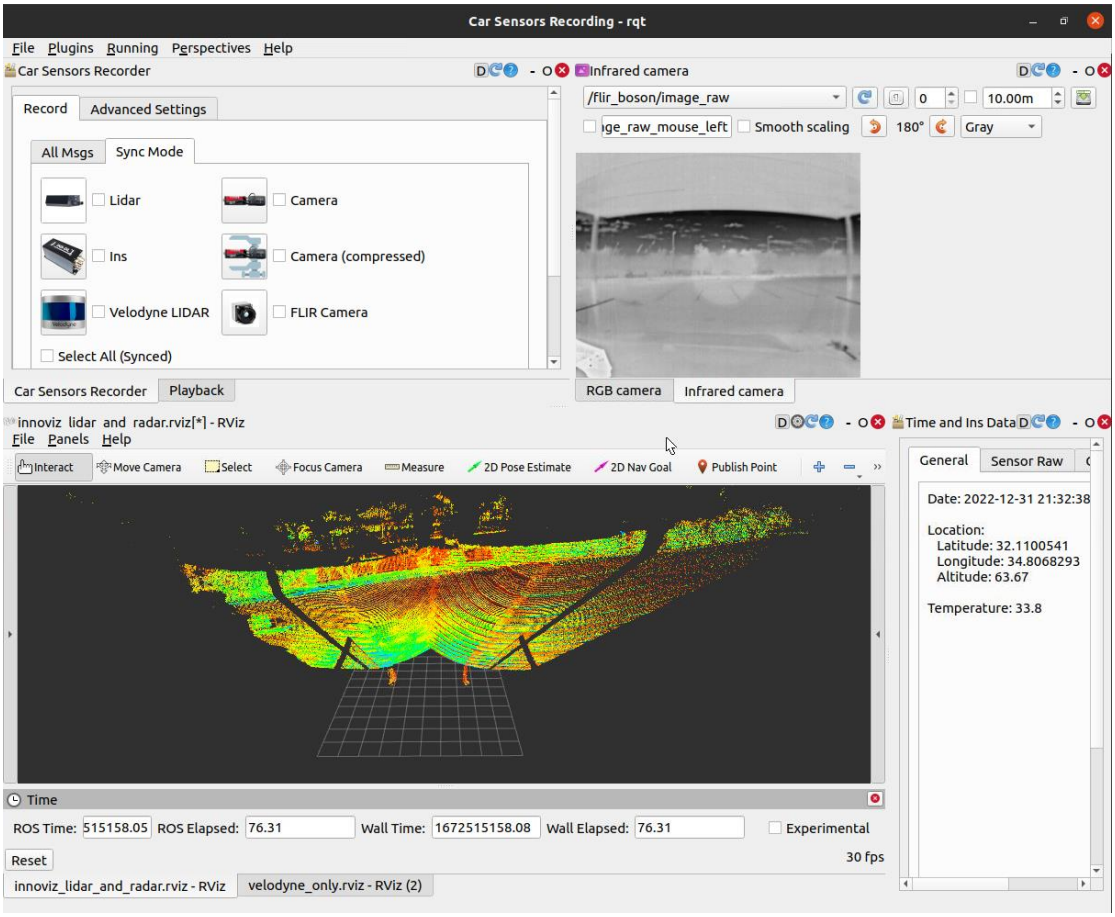
- **<u>חלוניות התמונה מהמצלמות</u>** (שורה עליונה) – בצד ימין מוצגת המצלמה התרמית (שמותקנת במקרה זה הפוך) ובצד שמאל ניתן לראות את מצלמת ה-RGB. במקרה זה התמונה מהמצלמה התרמית מוצגת כתמונה גולמית (raw image), והתמונה ממצלמת ה-RGB מוצגת לאחר מעבר ב-Node שנקרא Image Proc, הממיר אותה מתמונה גולמית לתמונת צבע.
- **<u>חלונית התצוגה של ה-LIDAR ושל מכ"ם הפגוש</u>** (שורה תחתונה באמצע) – בחלונית זו ניתן לראות את המכ"ם וה-LIDAR של Innoviz מוצגים על אותה מערכת צירים. לשם הצגת ה-LIDAR של Velodyne על מערכת הצירים יש לבטל את הצגת ה-LIDAR של Innoviz ומכ"ם הפגוש.
- **<u>חלונית רשימת התצוגות</u>** (שורה תחתונה בצד שמאל) – בחלונית זו ניתן לראות את התצוגות השונות ב-Rviz, להוסיף או להוריד תצוגות ולשנות את הפרמטרים שלהן.
- **<u>חלונית השליטה במערכת הצירים</u>** (שורה תחתונה מצד ימין) – בחלונית זו ניתן לכוון את הפניית התצוגה במערכת הצירים בצורה מדויקת, ולקפוץ בה למספר מצבים מוגדרים מראש של תצוגה (למשל מבט על).

4.2 – מערכת ההקלטות והסנכרון

כך נראה חלון מערכת ההקלטות:



איור 20 - חלון מערכת ההקלטות – מצלמת RGB מוצגת

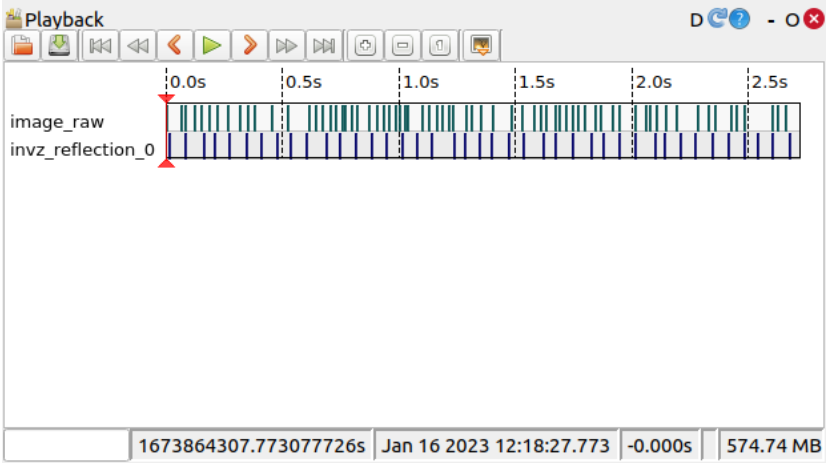


איור 21 - חלון מערכת ההקלטות – מצלמת IR מוצגת

החלון בנוי כך שניתן לראות בו שלושה topics ברגע נתון – קלט וידאו ממצלמה (שורה עליונה מימין), נתוני INS (שורה תחתונה מימין) ואת החיישנים שמפיקים פלט נקודות (מכ"ם ו-LIDARים, שורה תחתונה משמאל). על מנת לעבור בין המצלמות השונות, או בין ה-LIDAR של Innoviz והמכ"ם לבין ה-LIDAR של Velodyne, ניתן להשתמש בכרטיסיות בתחתית הבלוקים המתאימים.

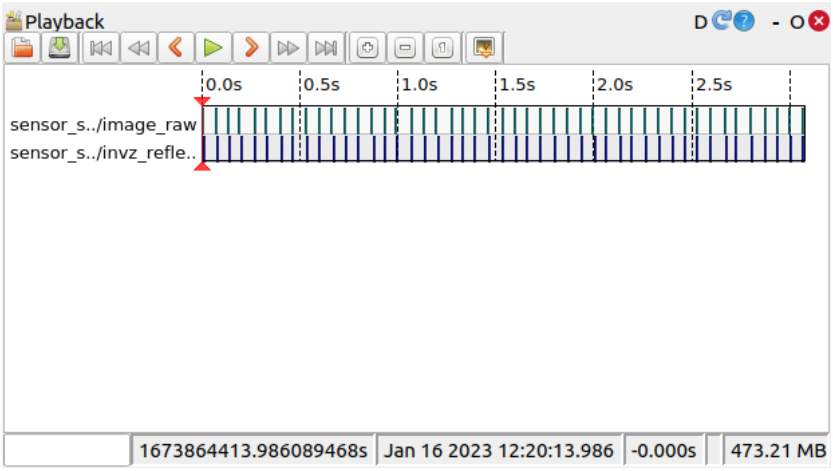
בשורה העליונה מצד שמאל נמצאת חלונית השליטה בהקלטות ובצפייה בהן (גם הם חולקים מקום, וניתן לעבור ביניהם בעזרת הכרטיסיות). בחלונית הצפייה בהקלטות ניתן לבחור האם להקליט עם או ללא סנכרון (בכרטיסיות All Msgs/Sync Mode), ובכל אחד מהמצבים ניתן לבחור את החיישנים המיועדים להקלטה ולהקליט אותם (מפאת חוסר מקום, כפתור ההקלטה מוסתר מעט. ניתן לגלול את החלונית על מנת לראות אותו).

בעת צפייה בהקלטה חלונית ה-Playback נראית כך:



איור 22 - חלונית ניהול הצפייה בהקלטות – הקלטה לא מסונכרנת

בעת צפייה בהקלטה מסונכרנת החלונית נראית כך:

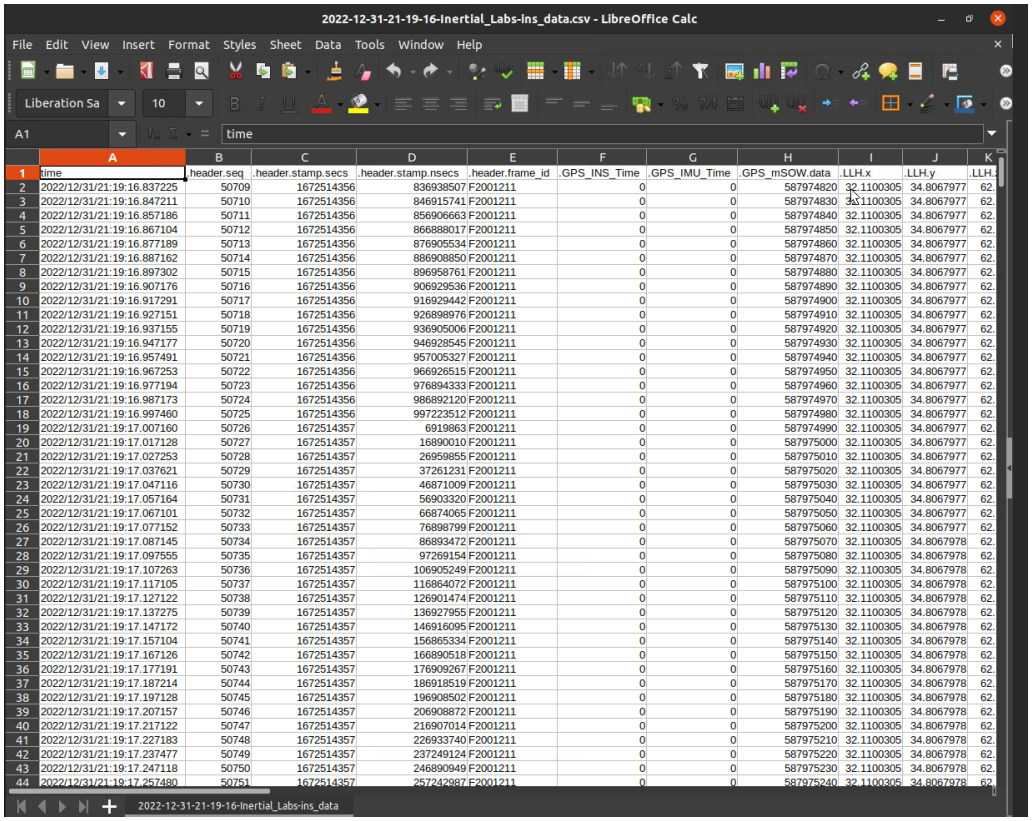


איור 23 - חלונית ניהול הצפייה בהקלטות – הקלטה מסונכרנת

כל שורה בחלונית הצפייה מייצגת Topic אחד, וכל קו אנכי מייצג הודעה ששודרה על ה-Topic הזה והוקלטה. ניתן לראות כי אכן, בהקלטה שאין בה שימוש במסנכרן הקווים האנכיים אינם מיושרים זה לזה – כלומר אין תיאום בין זמני הקבלה של ההודעות. לעומת זאת, בעת שימוש במסנכרן הקווים האנכיים קרובים הרבה יותר להיות מיושרים, והם גם הרבה פחות צפופים – פועל יוצא של העובדה שהמסנכרן לא מעביר את כל ההודעות שמגיעות אליו. כאן למשל החיישן האיטי ביותר הוא החיישן של Innoviz (שורה תחתונה בשתי התמונות), והוא משדר בקצב של 15 Hz ולכן קצב התמונות של המצלמה (שורה עליונה בשתי התמונות) בהקלטה זו הוא גם כן 15 fps.

4.3 – ייצוא קבצים

כאן ניתן לראות דוגמה לקובץ ה-CSV המתקבל מייצוא המידע מתוך הקלטה של ה-INS:

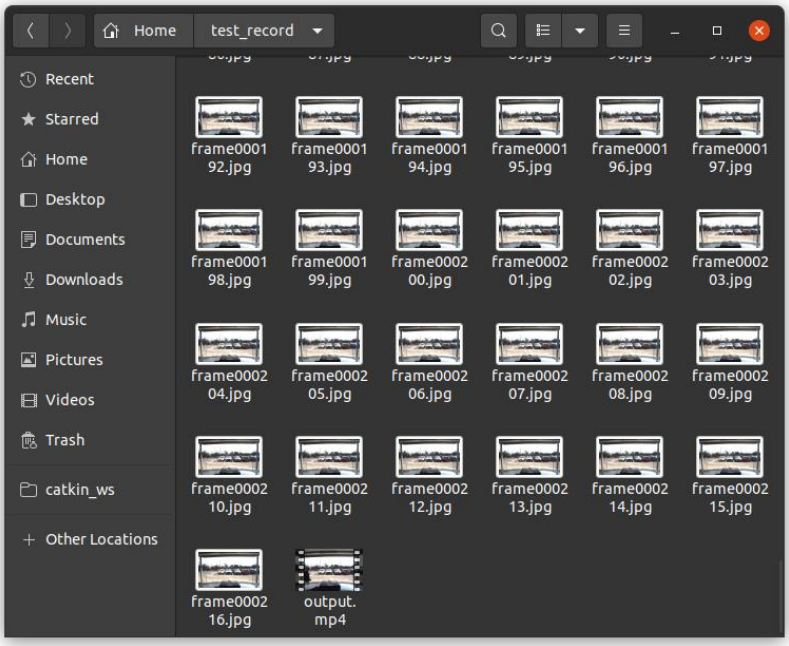


	A	B	C	D	E	F	G	H	I	J	K
1	time	.header.seq	.header.stamp.secs	.header.stamp.nsecs	.header.frame_id	.GPS_INS_Time	.GPS_IMU_Time	.GPS_mSOW.data	.LLH.x	.LLH.y	.LLH.
2	2022/12/31/21:19:16.837225	50709	1672514356	836938507	F2001211	0	0	587974820	32.1100305	34.8067977	62.
3	2022/12/31/21:19:16.847211	50710	1672514356	846915741	F2001211	0	0	587974830	32.1100305	34.8067977	62.
4	2022/12/31/21:19:16.857186	50711	1672514356	856906663	F2001211	0	0	587974840	32.1100305	34.8067977	62.
5	2022/12/31/21:19:16.867104	50712	1672514356	866888017	F2001211	0	0	587974850	32.1100305	34.8067977	62.
6	2022/12/31/21:19:16.877189	50713	1672514356	876905534	F2001211	0	0	587974860	32.1100305	34.8067977	62.
7	2022/12/31/21:19:16.887162	50714	1672514356	886908850	F2001211	0	0	587974870	32.1100305	34.8067977	62.
8	2022/12/31/21:19:16.897302	50715	1672514356	896958761	F2001211	0	0	587974880	32.1100305	34.8067977	62.
9	2022/12/31/21:19:16.907176	50716	1672514356	906929536	F2001211	0	0	587974890	32.1100305	34.8067977	62.
10	2022/12/31/21:19:16.917291	50717	1672514356	916929442	F2001211	0	0	587974900	32.1100305	34.8067977	62.
11	2022/12/31/21:19:16.927151	50718	1672514356	926898976	F2001211	0	0	587974910	32.1100305	34.8067977	62.
12	2022/12/31/21:19:16.937155	50719	1672514356	936905006	F2001211	0	0	587974920	32.1100305	34.8067977	62.
13	2022/12/31/21:19:16.947177	50720	1672514356	946928545	F2001211	0	0	587974930	32.1100305	34.8067977	62.
14	2022/12/31/21:19:16.957491	50721	1672514356	957005327	F2001211	0	0	587974940	32.1100305	34.8067977	62.
15	2022/12/31/21:19:16.967253	50722	1672514356	966926515	F2001211	0	0	587974950	32.1100305	34.8067977	62.
16	2022/12/31/21:19:16.977194	50723	1672514356	976894333	F2001211	0	0	587974960	32.1100305	34.8067977	62.
17	2022/12/31/21:19:16.987173	50724	1672514356	986892120	F2001211	0	0	587974970	32.1100305	34.8067977	62.
18	2022/12/31/21:19:16.997460	50725	1672514356	997223512	F2001211	0	0	587974980	32.1100305	34.8067977	62.
19	2022/12/31/21:19:17.007160	50726	1672514357	6919863	F2001211	0	0	587974990	32.1100305	34.8067977	62.
20	2022/12/31/21:19:17.017128	50727	1672514357	16890010	F2001211	0	0	587975000	32.1100305	34.8067977	62.
21	2022/12/31/21:19:17.027253	50728	1672514357	26959855	F2001211	0	0	587975010	32.1100305	34.8067977	62.
22	2022/12/31/21:19:17.037621	50729	1672514357	37261231	F2001211	0	0	587975020	32.1100305	34.8067977	62.
23	2022/12/31/21:19:17.047116	50730	1672514357	46871009	F2001211	0	0	587975030	32.1100305	34.8067977	62.
24	2022/12/31/21:19:17.057164	50731	1672514357	56903320	F2001211	0	0	587975040	32.1100305	34.8067977	62.
25	2022/12/31/21:19:17.067101	50732	1672514357	66874065	F2001211	0	0	587975050	32.1100305	34.8067977	62.
26	2022/12/31/21:19:17.077152	50733	1672514357	76898799	F2001211	0	0	587975060	32.1100305	34.8067977	62.
27	2022/12/31/21:19:17.087145	50734	1672514357	86893472	F2001211	0	0	587975070	32.1100305	34.8067978	62.
28	2022/12/31/21:19:17.097555	50735	1672514357	97269154	F2001211	0	0	587975080	32.1100305	34.8067978	62.
29	2022/12/31/21:19:17.107263	50736	1672514357	106905249	F2001211	0	0	587975090	32.1100305	34.8067978	62.
30	2022/12/31/21:19:17.117105	50737	1672514357	116864072	F2001211	0	0	587975100	32.1100305	34.8067978	62.
31	2022/12/31/21:19:17.127122	50738	1672514357	126901474	F2001211	0	0	587975110	32.1100305	34.8067978	62.
32	2022/12/31/21:19:17.137275	50739	1672514357	136927955	F2001211	0	0	587975120	32.1100305	34.8067978	62.
33	2022/12/31/21:19:17.147172	50740	1672514357	146916095	F2001211	0	0	587975130	32.1100305	34.8067978	62.
34	2022/12/31/21:19:17.157104	50741	1672514357	156865334	F2001211	0	0	587975140	32.1100305	34.8067978	62.
35	2022/12/31/21:19:17.167126	50742	1672514357	166890518	F2001211	0	0	587975150	32.1100305	34.8067978	62.
36	2022/12/31/21:19:17.177191	50743	1672514357	176909267	F2001211	0	0	587975160	32.1100305	34.8067978	62.
37	2022/12/31/21:19:17.187214	50744	1672514357	186918519	F2001211	0	0	587975170	32.1100305	34.8067978	62.
38	2022/12/31/21:19:17.197128	50745	1672514357	196908502	F2001211	0	0	587975180	32.1100305	34.8067978	62.
39	2022/12/31/21:19:17.207157	50746	1672514357	206908872	F2001211	0	0	587975190	32.1100305	34.8067978	62.
40	2022/12/31/21:19:17.217122	50747	1672514357	216907014	F2001211	0	0	587975200	32.1100305	34.8067978	62.
41	2022/12/31/21:19:17.227183	50748	1672514357	226933740	F2001211	0	0	587975210	32.1100305	34.8067978	62.
42	2022/12/31/21:19:17.237477	50749	1672514357	237249124	F2001211	0	0	587975220	32.1100305	34.8067978	62.
43	2022/12/31/21:19:17.247118	50750	1672514357	246890949	F2001211	0	0	587975230	32.1100305	34.8067978	62.
44	2022/12/31/21:19:17.257480	50751	1672514357	257242987	F2001211	0	0	587975240	32.1100305	34.8067978	62.

איור 24 - ייצוא CSV של מידע ה-INS-GPS

כל שורה בקובץ מייצגת דגימה אחת של ה-INS.

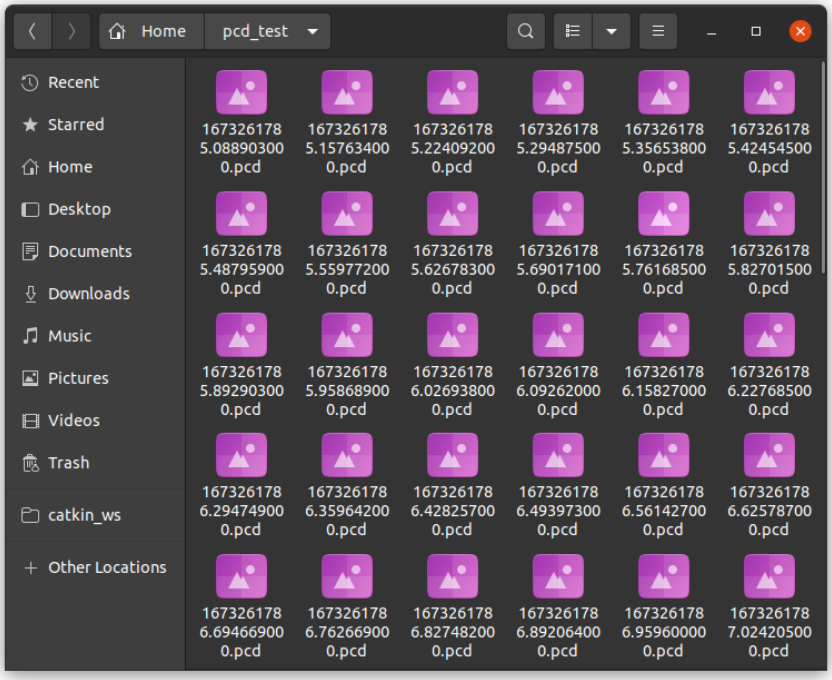
את ההקלטה של המצלמה ניתן לייצא כתיקיית תמונות JPEG, כך:



איור 25 - תיקיית תמונות מיוצאות מתוך הקלטת Rosbag

כל תמונה בתיקייה מייצגת דגימה מהמצלמה. קובץ ה-mp4 לא נוצר ישירות מתוך סקריפט הייצוא של התמונות אלא ע"י הרצת ffmpeg בתיקייה שנוצרת ע"י הסקריפט.

כך נראית תוצאת הייצוא של Pointcloud מהקלטה לתיקייה של קבצי PCD:



איור 26 - תיקיית קבצי PCD (ענני נקודות) המיוצאים מתוך Rosbag

כל קובץ PCD בתיקייה זו הוא ענן הנקודות בנקודת זמן כלשהי.

חלק 5 – סיכום, מסקנות והצעות להמשך

5.1 – סיכום

בתיאור תוצאות הפרויקט הראנו שעמדנו במטרות הפרויקט שהוצבו לנו:

- **מערכת הקלטות מסונכרנת לרכב אוטונומי:** בנינו מערכת הקלטות מסונכרנת, מימושה הוא באמצעות חבילת RQt GUI שנמצאת ב-ROS1. המערכת המאוחדת מכילה תוספים חיצוניים שנוספו, חלקם נכתבו על ידינו וחלקם היו מוכנים מראש, על מנת שהתצוגה הוויזואלית, הקלטת וניגון הקלטות החיישנים תהיה נוחה למשתמש.
- **מידע ממגוון חיישנים:** המידע נאסף בזמן אמת מכל החיישנים המותקנים, נכון לזמן כתיבת ספר זה, על הרכב. כל החיישנים הותקנו בROS וניתן לצפות בהם באחת משתי תצוגות ה - ROS המאוחדות הקיימות במערכת: RQt ב-ROS1, Rviz ב-ROS2.
- **ייצוא ההקלטות בפורמט נוח לעיבוד:** כפי שראינו בחלק 4.3 של תיאור תוצאות הפרויקט, ישנם מספר סקריפטים שנכתבו על מנת שנוכל לייצא את הקלטות החיישנים השונות בפורמט נוח לעיבוד. הייצוא מתבסס על CSV, פורמט JPEG לתמונות, MP4 לווידאו, PCDs ל-pointcloud.
- **הקמת GUI מאוחד:** הGUI כולל את הצגת כל החיישנים השונים בתצוגה ויזואלית נוחה. כמו כן, ניתן לבצע הקלטות מתוך הGUI ולהציג את תוצאות ההקלטות בתוך הGUI עצמו. קיים GUI גם במערכת ROS1 (RQt) וגם במערכת ROS2 (Rviz2) אולם, ב-ROS2 לא ניתן לבצע הקלטות דרך ה-GUI.
- **התקנת החיישנים במערכת ROS2:** חיישן ה-LIDAR של חברת Velodyne הותקן ישירות ב-ROS2 והחיישנים השונים מוצגים ב-ROS2 בעזרת ה-node: ROS1 bridge, ולכן גם מטרה זו מולאה כנדרש.
- **יכולת הצגת מידע נוחה ממספר חיישנים:** ב-GUI המאוחד, ניתן להציג את הרדאר וה-LIDAR באופן מקבילי על אותו frame. בצורה זו, ניתן לקבל תצוגה תלת ממדית של החלל מסביב לרכב.

כמו כן, חשוב לציין שהמערכת העומדת במרכזו של פרויקט זה היא מערכת מודולרית, שבה ניתן להוסיף או להסיר חיישנים לפי הצורך – למשל לצורך שדרוג של אחד החיישנים או לצורך הרחבת יכולות המערכת, יכולת זו מתאפשרת עקב השימוש ב-ROS כבסיס לבניית מערכת ההקלטות המסונכרנת, כפי שצוין בפרקים קודמים.

5.2 – מסקנות

במהלך העבודה על הפרויקט הגענו לשתי מסקנות מרכזיות באשר לשימוש בROS:

- ישנה בעייתיות לעבוד עם ROS2 בשלב זה. הסיבה היא שיצרני החיישנים מפיצים חבילות התקנה ל-ROS1 ולא מפיצות חבילות תואמות גם לROS2. לכן, על מנת להתקין את החיישנים ישירות ב-ROS2 יש צורך לשנות את הקוד שמותאם ל-ROS1 על מנת להמיר אותו ל-ROS2. מדובר בתהליך מורכב שדורש מפתח תוכנה בשפת C++ ו-Python.
- מערכת מאוחדת שתומכת גם ב-ROS1 וגם ב-ROS2 צריכה להיות מוקמת, בשלב זה, אך ורק בUbuntu 20.04 מכיוון שבמערכת זו קיימת התמיכה הטובה ביותר להפצת Noetic של ROS1 וגם להפצת Foxy של ROS2. כמו כן, אם יש צורך לעבוד עם שתי הגרסאות של ROS יש צורך לעבוד ב-Python 3 ולא ב-Python 2, ולכן נכון לעכשיו יש צורך לעבוד בהפצת Noetic של ROS1, ולא בהפצות מוקדמות יותר על מנת לעבוד עם Python 3, וזו סיבה נוספת בה בחרנו בהפצת Noetic.

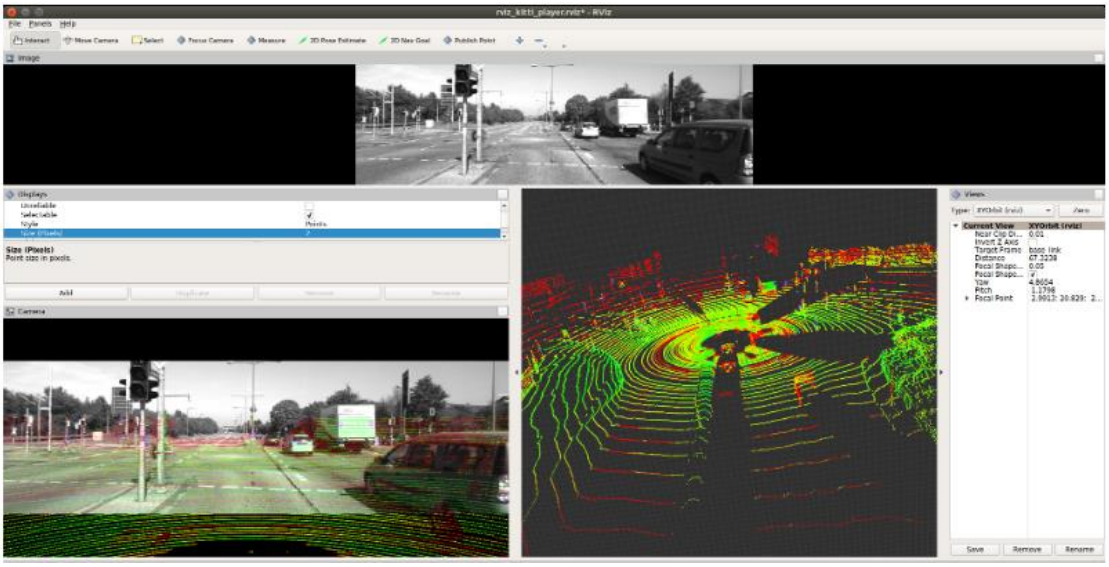
5.3 – הצעות להמשך

ישנם מספר כיוונים להמשך עליהם חשבנו:

- **הרחבת המערכת** – ישנם חיישנים חדשים שאפשר להתקין ברכב על מנת לשדרג את יכולות המערכת. למשל, התקבל LIDAR מדגם Ouster OS1-128 לאחרונה, ובהתאם לחבילות שמסופקות על ידי היצרן, ניתן לבצע התקנה של החיישן בROS. לאחר מכן, ניתן להוסיף את החיישן למערכת ההקלטות המסונכרנת בהתאם להוראות המסופקות במסמך התיעוד שנמצא בGitHub של הפרויקט. [22]
- **העברת ה-node של INS לROS2** – כפי שצוין במימוש הפרויקט (חלק 3.2), ה – node של ROS bridge לא מאפשר להעביר את המידע מה-INS ל-ROS2 מפני שההודעות שהוא מפרסם אינן בפורמט המוכר על ידי ה-Bridge בברירת מחדל. על מנת שה-Bridge יכיר בפורמט, יש להוסיף לקוד המקור שלו את המיפויים עבור ההודעות שרוצים להוסיף הכרה בהן ולהדר אותו מחדש. תהליך זה מפורט במסמך ROS bridge שנמצא בGitHub של הפרויקט.
- **התקנת החיישנים ב-ROS2 ללא שימוש בROS bridge** – מלבד ה-LIDAR של Velodyne, החיישנים השונים ברכב הותקנו ב-ROS1 והועברו ל-ROS2 דרך ה-node של ROS1 bridge. כפי שהוסבר במימוש הפרויקט, הסיבה לכך היא שיצרני החיישנים מפרסמים חבילות ל-ROS1 ולא ל-ROS2 ולכן על מנת להתקין את החיישן בROS2 יש צורך לשנות את הקוד שמותאם ל-ROS1 על מנת להמיר אותו ל-ROS2. מדובר בתהליך מורכב שדורש מפתח תוכנה בשפת C++ ו-Python.

- **הקמת תשתית חיישנים מאוחדת לכלים האוטונומיים בפקולטה להנדסה** – ישנם מגוון פרויקטים בפקולטה להנדסה שבהם משתמשים בROS על מנת להציג את החיישנים השונים באותה המערכת. לכן, על מנת למנוע בזבוז משאבים שלא לצורך ועבודה כפולה שנעשתה כבר בפרויקט אחר, יש צורך בתשתית חיישנים מאוחדת שתהיה משותפת לסירה, לרכב הנוסעים ולפורמולה.
- **היתוך בין המצלמה לLIDAR של Innoviz** – כאשר סיימנו את העבודה על מערכת ההקלטות המסונכרנת, התחלנו לחשוב על כיוונים מעניינים לביצוע בהמשך. אחד הכיוונים שמשך את עינינו הינו ביצוע fusion בין מצלמה ל-LIDAR. מכיוון שתהליך זה הינו מורכב לא הספקנו לנסות זאת במערכת שלנו, אולם בתור כיוון להמשך ניתן לבצע שימוש בGitHub שמכיל מימוש אפשרי של fusion בין מצלמה ל-LIDAR. [23]

ב-GitHub מוסבר באילו packages של ROS משתמשים על מנת לבצע את ההיתוך, בנוסף לכך מוסבר שיש שימוש ב-KITTI dataset על מנת לזהות עצמים שונים במרחב.



איור 27 - מימוש אפשרי של fusion בין מצלמה לlidar

חלק 6 – תיעוד

הפרויקט מתועד בספר הדרכה מפורט שנכתב על ידינו והועלה יחד עם קוד המקור לגיטהאב [22]. מסמך התיעוד בנוי כמדריך למשתמש, ונכתב כך שבעזרתו יהיה ניתן לבנות את מערכת ההקלטות מאפס. המסמך מוליך את הקורא דרך כל שלבי ההתקנה – התקנת ROS, התקנת החיישנים השונים וחבילות ה-ROS שלהם, התקנת ה-Bridge והתקנת רכיבי מערכת ההקלטות – כך שבסופו של דבר המשתמש מסוגל לשחזר את המערכת מאפס על התקנה נקייה של לינוקס. כמו כן, ספר התיעוד כולל גם הדרכה על הוספת חיישנים למערכת ההקלטות ול-Node המסנכרן, כך שבמידת הצורך המשתמש יכול להרחיב את המערכת כפי שעשינו בפרויקט זה.

הקבצים שנמצאים בגיטהאב הם קבצים שיצרנו בעצמנו או שערכנו בהם שינויים רבים:

- תיקיות המכילות את כל התוספים ל-RQt שאינם סטנדרטיים:
  - תוסף השליטה בהקלטות
  - תוסף הצגת המידע מה-INS
  - תוסף ה-Playback (עבר שינויים כמתואר בחלק 3)
- תיקיית חבילת ה-ROS הממשת את סנכרון החיישנים
- סקריפטים וקבצי Launch שונים שיצרנו:
  - Launch File מאוחד של כל המערכת
  - סקריפט ה-bridge_setup.bash – הפעלת Bridge בטרמינל שבו רץ הסקריפט, ללא צורך באפשור ROS מראש.

כמו כן, צולמו מספר סרטוני הדגמה של פעולת המערכת, הזמינים ב-YouTube [24].

חלק 7 – מקורות

מידע על ROS

[1] "ROS/Concepts – ROS Wiki" – http://wiki.ros.org/ROS/Concepts

[2] "ROS2 vs. ROS1— key differences and which one is better?" – https://medium.com/@oelmofty/ros2-how-is-it-better-than-ros1-881632e1979a

מידע על הפרוטוקולים המשמשים אותנו

[3] "CAN bus – Wikipedia" – https://en.wikipedia.org/wiki/CAN_bus

[4] "CAN bus structure (Source: ISO 11898-2)" – https://www.researchgate.net/figure/CAN-bus-structure-Source-ISO-11898-2_fig77_321527129

[5] "TCP/IP – ויקיפדיה" – https://he.wikipedia.org/wiki/TCP/IP

[6] "RS-232 – ויקיפדיה" – https://he.wikipedia.org/wiki/RS-232

דפי הנתונים, הדרייברים וחבילות ה-ROS של החיישנים השונים

[7] "Velodyne LiDAR PUCK™" – Velodyne Acoustics Incorporated, 2015 – https://www.amtechs.co.jp/product/VLP-16-Puck.pdf

[8] "ros-drivers/velodyne at ros2" – https://github.com/ros-drivers/velodyne/tree/ros2

[9] "LiDAR Capabilities for Innoviz One" - https://innoviz.tech/innovizone

[10] "Innoviz API – GitHub" – https://github.com/InnovizTechnologies/InnovizAPI.git

[11] "Prosilica GT1930" – https://www.alliedvision.com/en/camera-selector/detail/prosilica-gt/1930/

[12] "Vimba SDK for machine vision cameras – Allied Vision" – https://www.alliedvision.com/en/products/vimba-sdk/#c1497

[13] "ros-drivers/prosilica_gige_sdk" – https://github.com/ros-drivers/prosilica_gige_sdk

[14] "ros-drivers/prosilica_driver" – https://github.com/ros-drivers/prosilica_driver

[15] "FLIR Boson" – FLIR Systems Inc., 2019 – https://f.hubspotusercontent10.net/hubfs/20335613/flir-boson-datasheet.pdf

[16] "astuff/flir_boson_usb" – https://github.com/astuff/flir_boson_usb

[17] "Delphi Electronically Scanning RADAR" – AutonomousStuff – https://hexagondownloads.blob.core.windows.net/public/AutonomouStuff/wp-content/uploads/2019/05/delphi-esr-whitelabel.pdf

[18] "unizg-fer-lamor/radar_interface" – https://bitbucket.org/unizg-fer-lamor/radar_interface/src/master/

[19] "Dual Antenna GPS-Aided Inertial Navigation System – INS-DL OEM" – InertialLabs, 2019 – https://inertiallabs.com/wp-content/uploads/2020/08/INS-D-OEM-Datasheet.rev2_.5_August_2020.pdf

[20] "Browse INS-MRU-AHRSII-IMU / inertiallabs-ros-pkgs – Stash" – https://us.inertiallabs.com:31443/projects/INS/repos/inertiallabs-ros-pkgs/browse

מידע על אלגוריתם הסנכרון

[21] "message_filters/ApproximateTime – ROS Wiki" – http://wiki.ros.org/message_filters/ApproximateTime

קישור לעמוד ה-GitHub שבו מאוחסן הפרויקט

[22] "shakednathan/RQt_synced_recording_system" – https://github.com/shakednathan/RQt_synced_recording_system

הצעה ל-Fusion בין מצלמה ל-LIDAR

[23] "LiuFG/Camera-Lidar-Fusion ROS: fully applied in ROS. Simply fuse the category and location information" – https://github.com/LiuFG/Camera-Lidar-Fusion-ROS

סרטוני הדגמה של הפרויקט

[24] "ROS Recording System Demo – YouTube" – https://www.youtube.com/playlist?list=PL6P-P3hs1jH7vE9yxNVtruY3rPBRnYQwL