



The Iby and Aladar Fleischman  
Faculty of Engineering  
Tel Aviv University

הפקולטה להנדסה  
ע"ש איבי ואלדר פליישמן  
אוניברסיטת תל אביב



# רחפן אוטונומי היוצא מחדרים

## "יתוש"

פרויקט מס' 21-2-2370

דו"ח סיכום

מבצעים:

315316935

רון פרימר

208455162

עליזה יפעי

מנחה:

אוניברסיטת חיפה

פרופ' דן פלדמן

וזיר מנחה:

מעבדת הרחפנים

רונאל שור-פקר

מקום ביצוע הפרויקט:

מעבדת הרחפנים של אוניברסיטת תל אביב

ראש מעבדת הרחפנים:

דר' דן רביב

## תוכן עניינים

4.....	תקציר .....
5.....	הקדמה .....
6.....	רקע תיאורטי .....
6.....	אלגוריתם ORB-SLAM2 .....
7.....	אלגוריתם Home Guard Drone .....
9.....	סימולציה .....
10.....	ימוש: .....
11.....	תיאור חומרה: .....
12.....	תיאור תוכנה: .....
21.....	פיתוח תוכאות: .....
22.....	השוואות בין תוכאות הסימולציה לפועל בזמן אמת: .....
23.....	השוואות בין תוכאות האלגוריתם שלנו לאלגוריתם הקיים לפני: .....
24.....	סיכום מסקנות והצעות להמשך .....
24.....	תיעוד הפרויקט .....
25.....	מקורות .....
25.....	מאמריפים: .....
25.....	מקורות מהאינטרנט: .....
25.....	רכיבים: .....
25.....	סרטוניים: .....
26.....	נספחים .....

### רישימת אירופים:

4.....	איור 1 – דיאגרמת בלוקים.....
6.....	איור 2 – דיאגרמת בלוקים המתארת את פועלות האלגוריתם ORB-SLAM
10.....	איור 3 – דיאגרמת בלוקים המתארת את אופן תחיליך מימוש האלגוריתם
10.....	איור 4 – דיאגרמת בלוקים המתארת באופן כללי את שלביים העיקריים בפעולת הרחפן מרגע הדלקתו ועד הסיום.
11.....	איור 5 - תיאור הרחפן בו השתמשו והחקקים העיקריים הנמצאים בו.....
12.....	איור 6 - דיאגרמת בלוקים של תיאור שלבי האלגוריתם שכתבנו.....
12.....	איור 7 - מפת נקודות המתקבלת מסריקה במעבדה.....
13.....	איור 8.1 - חלוקה לזרזות עם הערך angle_res=2 .....
13.....	איור 8.2 - חלוקה לזרזות עם הערך angle_res=30 .....
14.....	איור 9.1 - התפלגות המרחקים עבור גזרה שאינה מכילה את היציאה.....
14.....	איור 9.2 - התפלגות המרחקים עבור גזרה המכילה את היציאה.....
15.....	איור 10 – המכשלה על אופן בחירת הערך precent .....
16.....	איור 11 – גרפף המתאר את הנקודות לכל גזרה, עבור הגזרות הרגילות.....
17.....	איור 12 – גרפף המתאר את הנקודות לכל גזרה, עבור הגזרות הריקות.....
18.....	איור 13 - מיפוי החדר והצגת סריקת LIDAR .....
19.....	איור 14 - נתיב היציאה המתkeletal (בכחול), נקודת המוצא של הרחפן ונקודת הסופה .....
19.....	איור 15 - דיאגרמת בלוקים המתארת את אופן ביצוע האלגוריתם בפועל .....
21.....	איור 16.1 – יציאה משני חדרים בזאה אחר זה – חדר ראשון .....
21.....	איור 16.2 – יציאה משני חדרים בזאה אחר זה – חדר שני .....
22.....	איור 17 - דוגמא לתוצאות אחד החדרים בהשוואה של תוצאות הסימולציה מול התוצאות בזמן אמת .....
23.....	איור 18.1 - מפת הנקודות עם דרך היציאה עבור האלגוריתם שכתבנו.....
23.....	איור 18.2 - מפת הנקודות עם נקודות החשודות ליציאה עבור האלגוריתם הקיימן .....

### רישימת טבלאות:

22.....	טבלה 1 – השוואת הביצועים של הסימולציה מול התוצאות בזמן אמת.....
---------	---

### רישימת נוסחאות:

6.....	נוסחה 1 - מציאת נתוני מרחקים באלגוריתם ORB-SLAM2 .....
7.....	נוסחה 2.1 - שיפור כיוון המצלמה באלגוריתם ORB-SLAM2 .....
7.....	נוסחה 2.2 – טרנספורמציה מעבר באלגוריתם ORB-SLAM2 .....
13.....	נוסחה 3 – התפלגות הנקודות לאזוריות .....
14.....	נוסחה 4 – ממוצע מרחקים .....
14.....	נוסחה 5 – המשערך הלא מוטה להתפלגות נורמלית.....
14.....	נוסחה 6 – התפלגות נורמלית.....
15.....	נוסחה 7 – חישוב מספר הגזרות .....
15.....	נוסחה 8 – חישוב חזיות שבה מוכלת הדלת מכלל החדר .....
16.....	נוסחה 9 – חישוב נקודת מתאימה לזרזה .....
16.....	נוסחה 10 – השלמת התוחלת לאזוריות ריקות .....
16.....	נוסחה 11 – השלמת התוחלת לאזוריות ריקות .....
17.....	נוסחה 12 – חישוב נקודות עבור גזרות בעלות ערך שונות גובה .....
22.....	נוסחה 13 – חישוב אחוזי שינוי בתוצאות בזמן אמת .....

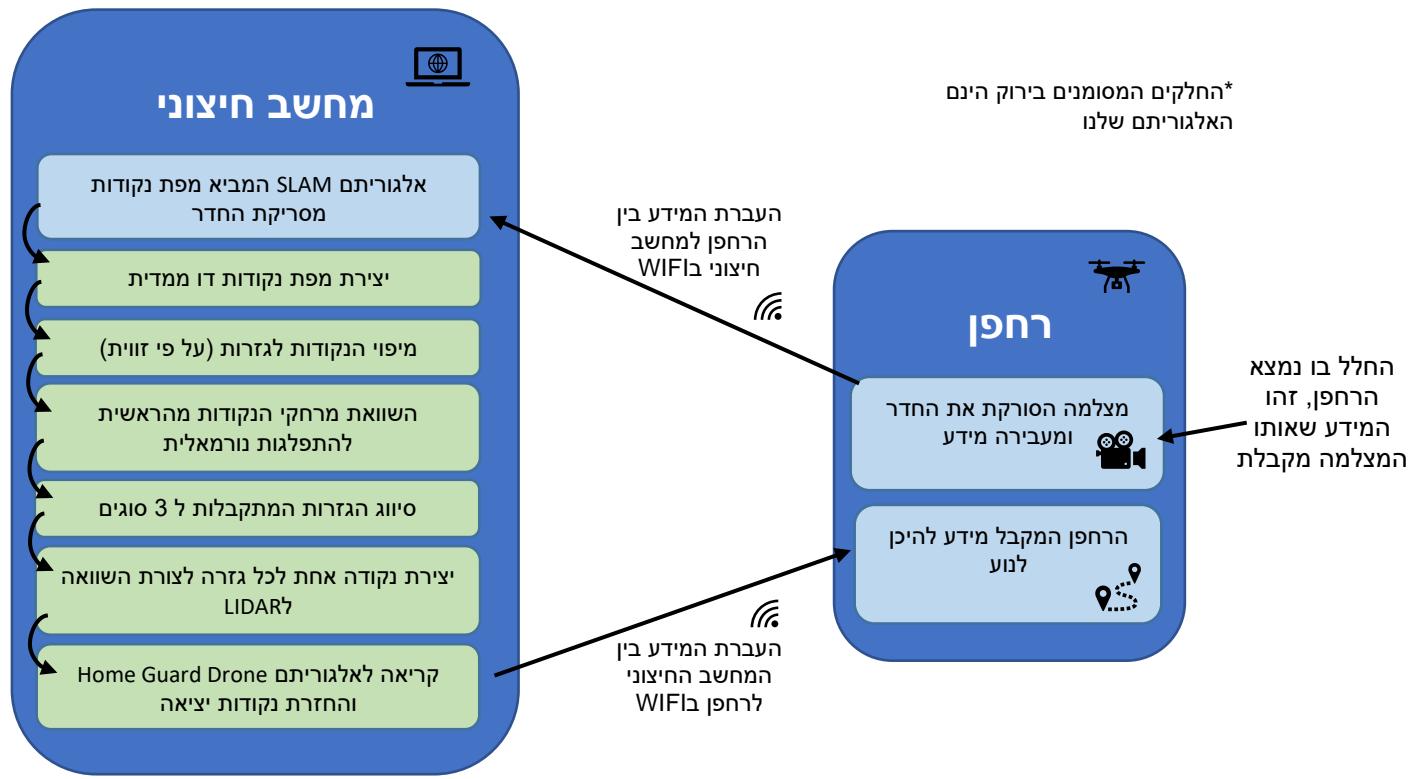
## תקציר

פרויקט זה עוסק בניוט אוטונומי של רחפן צעיר על מנת לסרוק חלל, ללא הכוונה ולא כל ידיעה מוקדמת לגבי החלל אותו הוא סורק. מטרתנו העיקרית בפרויקט היא לאגורם לרחפן שתהיה לו את היכולת לסרוק מספר חדרים על ידי קר שהוא סורק חדר אחד, מוצא את היציאה ממנו באופן עצמאי וחלוטין, עבר אותה וממשיך קר גם בחדר הבא.

לצורך קר, משתמש לרחפן בו קיימת מצלמת RGB בודדת שבעזרתה מתבצעת הסריקת החדר. לאחר קבלת סריקת החדר, הרחפן מוצא את נקודת היציאה בעזרת אלגוריתם שאותו אנו כותבו, שבסופו הוא מקבל את הנקודה אליה עליון לנוע. סריקת החדר נעשית על ידי השימוש באלגוריתם SLAM שיוצר מפת נקודות תלת ממדית של החדר תוך שימוש במכשיר בלבד.

אוףן התקשרות בין הרחפן למחשב היא על ידי WIFI, דרכו הרחפן מעביר את התמונות למחשב, לצורך הרצת אלגוריתם SLAM ובנין מפת הנקודות של סריקת החדר. התקשרות הינה זו כיוונית והרחפן מקבל פקודות מהמחשב גם כן, כמו לעוזר יציאה מהחדר.

במהלך העבודה בפרויקט, כותבו את האלגוריתם שמצוא את כיוון היציאה והביא לרחפן את הנקודה המתאימה لأن להתקדם. האלגוריתם שכתבנו מtabסס על מפת הנקודות המתקבלת מאלגוריתם ה-SLAM, חלוקת הנקודות לגזרות שונות וניתוח כל גזרה בעזרת כלים סטטיסטיים. בסופו של אלגוריתם זה מתקבל מייפוי של החדר הנוסף. מייפוי זה נעשה בהתאם למיפוי אשר היה מתתקבל מסריקת החדר על ידי חיישן LIDAR. ככלומר, בפרויקט זה הראנו דרך שבה אפשר בעזרת מצלמה בודדת למפות חדר בדומה למיפוי הנעשה על ידי חיישן LIDAR. עקב קר משתמש באלגוריתם שנועד לנתח סריקת LIDAR אשר יעזור לנו למצאו בסופו של דבר את מסלול היציאה הטוב ביותר של הרחפן מהחדר. נציג את דיאגרמת הבלוקים המתארת את האלגוריתם שנבנה:



איור 1 דיאגרמת בלוקים של כלל הפרויקט

התוצר הסופי של הפרויקט רחפן אשר סורק את החדר בעזרת מצלמת RGB בודדת, מוצא את נקודת היציאה וממנו אליה באופן אוטונומי לגמר. בנוסף בפרויקט זה הראינו כיצד ניתן ליצור מייפוי של חדר זהה למייפוי המתkeletal בעזרת חיישן LIDAR על ידי מצלמה בודדת. על גבי כל הסריקות החדרים שבחנו (באופן תיאורתי) לאלגוריתם זה מעל 70% הצלחה בזיהוי נקודת היציאה שזהו אחווד גובהה כמעט פי 3 מהאלגוריתם הנוכחי. בנוסף על כך האלגוריתם נבחן בזמן אמיתי על פני 3 חדרים שונים ובכלם הרחפן אכן יצא מהחדר. בדיקה נוספת שבוצעה ובערה בהצלחה הינה יציאה של הרחפן משני חדרים סמוכים באופן אוטונומי לומר.

## הקדמה

פרויקט זה הינו חלק מפרויקט גדול יותר המתקיים במעבדת רחפנים אוניברסיטת חיפה. במסגרת הפרויקט הגדול ישנו מספר צוותים העובדים על שימושים שונים שיכולים להיות לרחפן. המטרה המרכזית בפרויקט הכללי, שהיא החלק שלנו בפרויקט, היא שהרחפן ידע לנווט באופן אוטומטי בתוך מבנה לא ידוע. יכולתו של הרחפן לסרוק מבנה לא ידוע, מהוות אבן בסיס ליכולות רבות נוספות שאפשר להגיעה אליהן בתחוםים רבים. נביא כאן מספר דוגמאות לשימושים אפשריים:

- ביחסון - רחפן זה יוכל לשמש על מנת לסרוק מבנה אליו ישנו צורך מבצעי להיכנס ולהתריע על הימצאות אנשים, חומר נפץ (בעזרת אלגוריתמים אחרים לזרחי או בייקט/אנשים שצווותים אחרים עובדים עליהם) או סכנות אחרות שאורבות לחילום. היתרון בשימוש ברחפן זה יהיה שהוא קטן וקל לנשען והפעול שלו אינו ידרוש מחדח החילומים לדעת להטיס אותו. נציין כי על מנת שזה יהיה יישם יהיה צורך פתרון לבניית הרعش שעשויה הרחפן.
- חילוץLCDIM. במקרה של שריפה/רעידת אדמה יהיה ניתן לשולח את הרחפן על מנת לחפש לכוונים ברחבי מבנה בעלי לסן חי אדם נוספים.
- העברה של סחורות ברחבי מבנה – בעזרת אלגוריתם טוב המנות באופן אוטומטי ורחפן בעל יכולת נשיאת משקלים, יוכל להעביר סחורות מקצתו האחד של המבנה למשנהו ובכך לחסוך בכוח אדם.

על מנת שהרחפן יוכל לסרוק מבנה לא ידוע, מותקנת עליו מצלמת RGB בודדת אשר בעזרתה אנו יכולים ליצור מיפוי של החדר בעזרת אלגוריתם ORB-SLAM (שיтвор בהמשך). את הפלט של סריקה זו אנו מקבלים כקלט לאלגוריתם שלנו שմعبد את המידע המתkeletal ומהזיר את נקודת היציאה מהחדר באופן אוטונומי לגמר. לאחר שהתקבלה נקודה זו על הרחפן לנוט אליה בבטחה ולמצוא את היציאה מהחדר.

בפרויקט זה, נדרש לחשב על אלגוריתם חדש שיחיליף את האלגוריתם המקורי שתוכאטו אין מספקות מכיוון שאין מוצאת נקודת היציאה הנכונה מהחדר באחוז גבוה מהמרקם ועל כן אין מספק על מנת לתת לרחפן לנוט בכל מבנה לא מוכך. עביה נוספת של האלגוריתם המקורי הוא שהוא מחייב מספר נקודות אפשריות לניטוט גם במקרים בהם ישנה יציאה אחת ועל כן הוא אינו דומה ומדויק מספיק. שיפור נוסף שהוספנו היה הוספה של דרך הניטוט ולא רק את הנקודה שאליה יש להגעה על מנת שנוכל לוודא שהרחפן מנותם מיקומים בהם אין מכשולים כלל.

נוסף על האלגוריתם שמצוע בפרויקט זה, היה علينا לבצע אינטגרציה מלאה שלו אל המערכת הקיימת על מנת שנוכל לבחון אותו במבנים שונים ולוודא כי הרחפן אכן מנות אל היציאה מהחדר באופן מעשי ושבעצם האלגוריתם התיאורטי אכן עומד בבחן המציגות.

על מנת להגעה אל האלגוריתם היעיל ביותר הגורם לרחפן לצאת מהחדר, נדרש תחילת לעבור על מספר נושאים.

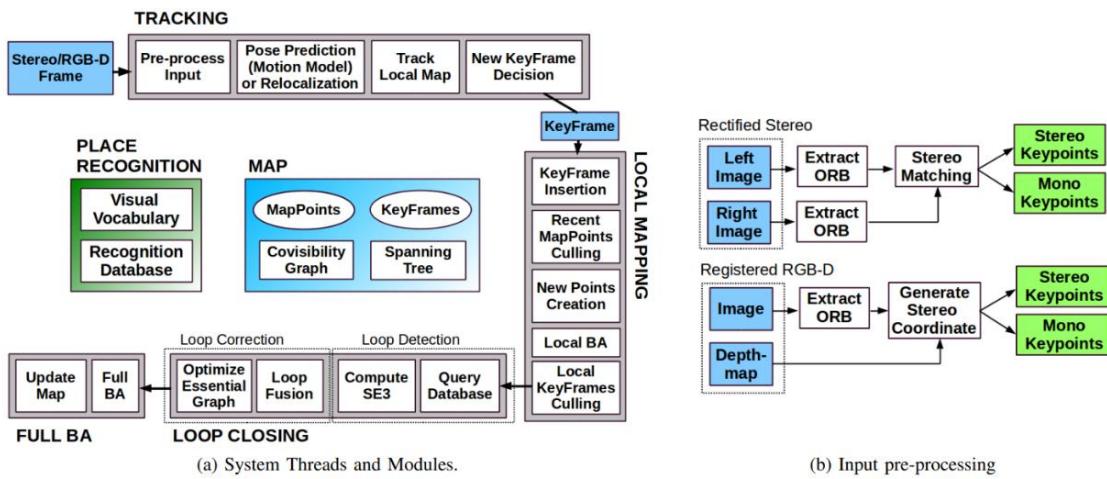
ראשית, נלמד על הקוד המקורי המביא פקודות בסיסיות לרחפן ואופן הפעלתו. בנוסף נלמד על המגבילות של הרחפן כגון זמן הטישה המשפע מהטוללה, הגודל, המהירות, הדיקון וכדומה. שנית, נעבור על אופן סריקת החדר ועל האלגוריתם ORB-SLAM שאחראי על פועלה זו. לאחר שנבנין את פועלות האלגוריתם זה ואת העיקרונות שעומדים מאחוריו, יוכל לגשת את הפלט שלו שהינו מפת הנקודות ולהבין היכן נמצאת היציאה ולהחשב על האלגוריתם המתאים ביותר למציאת היציאה. בעזרת רעיונות שונים שלמדו במהלך כל התואר ומארמים וגיישות שקראננו עליו (שיפורטו יותר בפרק תיאורטי) חקרו וניסינו מספר רב של אלגוריתמים עד שהגענו לאלגוריתם שתוכאוון מספקות והינו אלגוריתם משופר ביחס לאלגוריתם המקורי שמצא את היציאה באחוז רב יותר מהמרקם. אלגוריתם זה בנוסף על כך שהוא סכימה מדוקית ביותר של החדר בצורה טובה מאוד שבעזרתה נוכל ליצור סכימה מדוקית ביותר של החדר. בנוסף, אלגוריתם נעשה שימוש בצורה בה מנתחים סכימה של חדר שנעשה על ידי חישון LIDAR וכן בօנוסף למטרת הפרויקט העיקרית, מטרה נוספת תוקן כדי העבודה על הפרויקט הינה להראות כי ניתן ליצור סכימה זהה של החדר בעזרת מצלמת RGB בודדת כפי שמתකבלת על ידי סריקה בעזרת חישון LIDAR.

## רקע תיאורתי

במהלך העבודה על הפרויקט, השתמשנו במספר אלגוריתמים קיימים על מנת שנוכל להגיע אל הרעיון הטוב ביותר. נציג את האלגוריתמים בהם השתמשנו ובהמשך (בפרק 'הימוש') נראה ונראה אודות אופן הפעולה של האלגוריתם אותו אנו כתבו.

### אלגוריתם <sup>1</sup>: ORB-SLAM2

זהו אלגוריתם השיר לתחום ה-SLAM-Simultaneous Localization and Mapping. מטרת האלגוריתם היא להצליח לסרוק את החדר בו נמצא הרחפן וליצור מסריקה מפת נקודות תלת מימדית המשקפת באופן הטוב ביותר את החלל. אחד היתרונות הגודלים של האלגוריתם הוא של מנוף למפות את החדר, נעשה שימוש במצלמה אחת בלבד ולא חישון או LIDAR. האלגוריתם, מורכב משלושה שרשרות מקבילים עיקריים: מעקב, מיפוי מקומי וסגירת לולאה. נראה בדיאגרמה הבאה את השלבים, נציג שהתמונה נלקחה מתוך המאמר של האלגוריתם:



איור 2 דיאגרמת בלוקים המתארת את פעולה האלגוריתם ORB-SLAM

פירוט מלא של האלגוריתם נמצא במאמר, וסבירCut את עיקרי השלבים של האלגוריתם:

**1. Tracking:** האלגוריתם בשלב זה מקבל את המידע מתוך המצלמה ומזהה את מיקומו במרחב. המידע המועבר לאלגוריתם הוא מידע מצלמה בזדנת או מסוג סטראיו (זהו מידע המכיל בתוכו גם נתוני מרחוק). האלגוריתם מוצא עבור הקלט את הנקודות הרלוונטיות המעניינות אותו (לדוגמא נקודות המעידות על שינויים חדים של צבע) הנקראות KeyPoints ומרגע זה והילך אחר הקלט לא רלוונטי אלא רק הנקודות הללו. נציג בשלב זה, שמאחר והקלט שלו מגיע מעתה מצלמה ולא נתונים מרחוק, האלגוריתם מחשב את נתונים המרחוק ובכך הופך את הקלט לנקודה הדומה לנקודות סטראיו. נתונים המרחוק מחושבים באופן הבא:

$$u_R = u_L - \frac{f_x b}{d} \quad (1)$$

כאשר  $f_x$  הוא האורך האופקי ו-  $b$  זהoko בסיס בין מקור האור למצלמה.  $u_R$  - הקורדינטת של התמונה הימנית,  $u_L$  - הקורדינטת של התמונה השמאלית,  $d$  - עומק עבור התמונה השמאלית.

כרגע כל פריים שמתקיים מהצלמה, נקבעות KeyPoints המתאימות ומתקיימות במקביל השוואה לפריים הקודם על מנת למצוא שגיאות.

**2. Local-Mapping:** שלב זה מתרחש במקביל לשלב הקודם, בכל פריים מתבצעת ייצור מפה המתאימה לנקודות הרלוונטיות מהפריים, KeyPoints. המפה המקומית בשלב זה מבצעת אופטימיזציות וחישובי שגיאות. אחת הדריכים לכך היא באמצעות שיטת

<sup>1</sup> המאמר על האלגוריתם נמצא במקורות מספר [2]

הקוואורדינאטות הטלת מדדיות של המפה ולמצוא את המיקום הנוכחי באופן טוב יותר. הנוסחה בה משתמשים עבור motion-only-BA (R) ומיקומה (t) היא:

$$\{\mathbf{R}, \mathbf{t}\} = \underset{\mathbf{R}, \mathbf{t}}{\operatorname{argmin}} \sum_{i \in \mathcal{X}} \rho \left( \left\| \mathbf{x}_{(.)}^i - \pi_{(.)} (\mathbf{R} \mathbf{X}^i + \mathbf{t}) \right\|_{\Sigma}^2 \right) \quad (2.1)$$

כאשר  $\mathbf{R}$  היא פונקציית העלות- $\text{Huber}^2$ - $\mathbf{x}_{(.)}^i$  - ה- KeyPoints ( $\Sigma$  - מטריצת קוואריאנס המתקשרת לסקאלת - KeyPoints,  $\pi_{(.)}$  - הטרנספורמציה הבאה (יכולה להיות מונו/סטריאו):

$$\pi_m \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_z \frac{X-b}{Z} + c_x \end{bmatrix}, \pi_s \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_z \frac{X-b}{Z} + c_x \end{bmatrix} \quad (2.2)$$

כאשר  $(f_x, f_y)$  - אורך מהמקד,  $(c_x, c_y, b)$  - פרמטרים המתקבלים מהכיוול. לאחר וcut המפה לא הושלמה מתבצע BA מקומי בלבד (כלומר רק על מספר הפרימרים ה'נראים' למצלמה).

**3. Loop-Closing:** בשלב זה, האלגוריתם מזהה שהתקבלו פרימרים שכבר התקבלו בהתחלה, כלומר הסריקה הסטטית.Cut, מבוצעות בדיקות של שגיאות על מנת לוודא שלא קיבלנו שגיאה גדולה מצברת במהלך כל הסריקה. גם כאן מתבצע שימוש ב- full BA על מנת למצוא את המקום הנוכחי של הרחפן המדוקין וכן להשלים את כל מפת הנקודות המלאה.

בסיום שלושת השלבים העיקריים ולאחר שהמפה הושלמה, מתחבצע לקליזציה (Localization). בחלק זה, המפה המתתקבל נשמרת לתוך קובץ וכאשר נרצה לשוב את הרחפן, נוכל להשתמש בסריקה קיימת ולא לבצע סריקה חדשה. האלגוריתם משווה את הפרימרים המת接连ים מהמצלמה ברחפן לנקודות הקיימות במפה ומוצא את מיקום הרחפן במפה הקיימת. (בנהנזה שלא בוצעו שינויים משמעותיים במהלך החדר ויהיה ניתן לבצע השוואה לסריקה הקיימת).

### אלגוריתם <sup>3</sup>: Home Guard Drone

אלגוריתם זה נכתב על ידי דן מגזיניק ונitin לנוב אדיבוטו. דן מגזיניק הינו חלק מכלל הצוותים העובדים במעבדת הרוחפנים והוא חקר גם כדרך למצואו את נתיב הייזה של הרחפן מהחדר. בעבודתו של דן מגזיניק, הרחפן סורק את החלל בו הוא נמצא על ידי חישון LIDAR ולא מצלמת RGB כמו בפרויקט שלהם. עקב העובדה של דן ובעצת מנהחה הפROYיקט שלנו, החלטנו לנסתות ולדמתות את הסריקה המתתקבלת מצלמת RGB ייחודה לסריקה המתתקבלת מחישון LIDAR. לאחר שנצליח בכך, נוכל להשתמש באלגוריתם המוצע על ידי דן (קישור) שעיל פי מאמרנו מביב תוצאות טובות אף יותר לאלגוריתם הפופולרי <sup>4</sup>.

נרצה לתאר את שלבי האלגוריתם הרלוונטיים לפROYיקט זה כפי שתוארם במאמרו של דן מגזיניק:

1. המראה
2. סריקה בעדרת חישון <sup>5</sup>-TOF. בשלב זה הרחפן מבצע סריקה מלאה של החדר כלומר הוא מבצע סריקה של  $360^\circ$ . בתהיליך זה הרחפן מסתובב (סביב עצמו) במהירות קבועה. במהלך הסיבוב, חישון ה-TOF מודד את המרחק של האובייקט הקרוב ביותר מהרחפן. בנוסף,

<sup>2</sup> מאמר אודות פונקציות העלות נמצוא במקורות מס' 4 [11] וכן קישור להסביר במקורות [1]

<sup>3</sup> המאמר על האלגוריתם נמצא במקורות [1]

<sup>4</sup> המאמר על האלגוריתם נמצא במקורות [3]

<sup>5</sup> הסבר על TOF נמצא במקורות [10]

מהרhone מחולצת הדווית שבה הרhone נמצא מתחילה הסריקה. לאחר זה, בסוף הסריקה יוכל לקבל מיפוי של החדר וליצור 'מפה' של החדר. בשלב זה, ישנו סינון נוסף של אזורים שבהם התקבלו מעט מאוד נקודות שנובע מטעויות ובסופו מקבל את המפה המשוערת של החדר הנסרך.

**3. Occupancy grid<sup>6</sup>** – בשלב זה ישנו מידול בוליאני של הסביבה המתבסס על המפה המתתקבל בסוף השלב הקודם. בשלב זה נחלק את המרחק לריבועים קטנים (בגודל 3X3 ס"מ) ועבור כל ריבוע כזה ישן 3 אפשרויות שמתארות את מצבו:

- לבן- אין מכשולים
- שחור- מקום שיש בו מכשול
- אפור- מקום שאנו לא יודעים מה מצבו כי אינו נסרך

צבעו של כל ריבוע נקבע בהסתמך על המידע שהתקבל מהסריקה ובעזרת נוסחה פשוטה שמתיחסת לפילוג והמיון של הריבועים השחורים כפי שמוסבר בהפניה מהמאמר<sup>7</sup>.

**4. עיבוד-id grid**: בעזרת פילטר בדומה לשלב 2, ישנו סינון קבוצות קטנות של תאים לא ידועים. בנוסף, אנו מעבים את המכשולים על מנת להקטין את הסיכוי שהרhone יתגש בהם. אזורים אלו יוגדרו להיות עם ערך 1 (כמו שאר המכשולים) כאשר ערך 0 מתאר אזור חופשי ממכשולים והערך 0.5 מתאר אזור לא ידוע.

**5. Exploration**: בשלב זה ישנו חיפוש של נקודות היציאה אליה אנו רוצים להגיע. בשלב זה נעביר חלון בגודל של 6X6 תאים ונחפש מקומות בהן התפלגות של התאים היא צדו שלפחות 25% מהתאים יהיו לבנים (ריקים מכשולים) ולפחות 35% מהתאים יהיו אפורים (מקומות לא ידועים) על מנת להבטיח שהרhone תריה דרך למצוקם זה (התאים הלבנים) ושנחקור מקומות שאנו לא ידועים מה יש בהם והם בעלי עניין (התאים האפורים). במידה ולא נמצא אף חלון העומד בדרישות נקטין את גודל החלון לגודל 5X5 וכן הלאה עד שנעמוד בדרכו. מתוך כל החולנות המתאימים הללו, החלון הרחוק ביותר שנמצא נבחר להיות נקודות הניטות הבא של הרhone.

**6. מציאת נתיב יציאה**: לאחר מציאת הנקודה נרצה שהרhone יגוט אל נקודה זו בדרך הקצרה ומהירה ביותר (על מנת לחסוך סוללה). תחילתה נעשו שימוש באלגוריתם \* A Ark מס' Line – off LOS – P Sight Path. אלגוריתם זה מחפש תא 'נראה' (שניתן להגעה אליו) גם מנקודות המוצאת של הרhone וגם מנקודות הניטות שאליה הוא צריך להגיע. תא 'נראה' אחד ניתן לחבר בין נקודה נוספת (המוצאת של הרhone או נקודות הניטות) קו ליניארי יחיד שעובר רק דרך תאים לבנים (כלומר נקיים מכשולים). התא הנראה הסופי שנבחר הינו המוצע של כל התאים הנראים שנמצאו מוקדם ובייחד אליו נקבע 2 את דרך הניטות של הרhone בעזרת מקטיעים ליניארים: מנקודה המוצאת אל הנקודה הנראית ומהנקודה הנראית אל נקודה הייציאה שנמצאה בסוף שלב 5.

<sup>6</sup> הסבר על האלגוריתם נמצא במקורות [8]

<sup>7</sup> הסבר על הנוסחה המתאימה צבע נכוון לכל ריבוע נמצא כהפניה מתוך המאמר של דן [5]

## סימולציה

על מנת לבצע את הסימולציה היה علينا לכתוב את הקוד שלנו ולבוחן את נכונותו ואחיזי הצלחה שלו, נציג בשלב הראשון התחלנו לכתוב את הקוד באופן ישיר לתוך הפרויקט הכתוב בשפת C++ ונמצא על RP ובמערכת הפעלה UNIX. לאחר שהתחלנו במלואה, נתקלנו בקשיים רבים שגרמו לעיכוב בפרויקט והבנו כי علينا לבצע דברים אחרת. RP הינו מחשב מאוד איטי והוא מאד קשה לעבוד עליו עם האלגוריתם שלנו ולבצע את הסימולציות הדרושים. לכן, הבנו כי علينا לבצע שניי והתחלנו לרשום את האלגוריתם בשפה עליית על המחשב האישי שלנו (חזק בהרבה) ורק לאחר מכן לנסות להטמיינו בפרויקט הקים. על כן, הסימולציה נכתבה בתוכנה MATLAB בשילוב עם תוכנת

*python* על גבי מחשב עם מערכת הפעלה Windows 10.

לקחנו מפת נקודות קיימת ממאגר מפות הנקודות ובדקנו את האלגוריתם שלנו על המפה (נתאר את פעולה האלגוריתם שאנו כtabנו בהמשך, בפרק המימוש). מפת הנקודות המתקבלת הייתה בקובץ CSV ואotta העברנו לטור סיבת העבודה הראשונית בה השתמשנו- MATLAB. לאחר מכן קיבלנו את התוצאות שציפינו לקבל וראינו שהקוד שכתבנו אכן פועל נכון, העברנו את הפלט שלנו אל האלגוריתם מהמאמר של דז<sup>8</sup> הכתוב ב- *python*. לשם כך, היה علينا להמיר את הנקודות המתקבלות בסיום האלגוריתם שלנו אל קובץ מסוג mat (שם שהנקודות נשמרו באופן של מטריצה) וביצענו קריאה אל הקובץ בקוד ה- *python* על מנת לקבל את הנקודות (קלט).

לאחר קבלת תוצאה טובה על מפת הנקודות זו, הורדנו מתיקית הפרויקט באתר GitHub מספר רב של מפות ובחנו את האלגוריתם עליהם וביצענו שיפורים על מנת לקבל אלגוריתם כולל המתאים לכמה שיטור מפות.

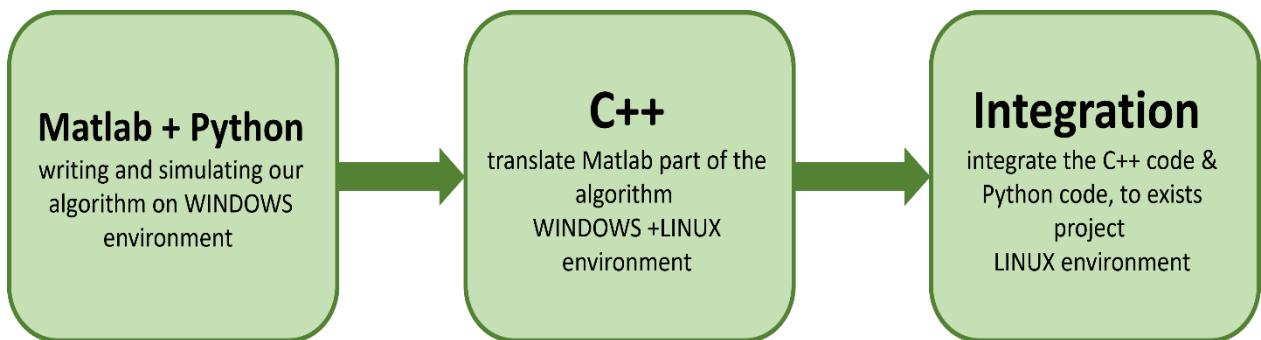
לאחר בינה של מספר רב של מפות והגעה ל贤איה הרצiosa מבחינת ביצועי האלגוריתם, רצינו לבצע סימולציה על החדרים שעלייהם תיכנו לבדוק את הרחפן בסוף התהילה. לכן, עוד לפני הטעמאת האלגוריתם בפרויקט הקים ובדיקה-real-time, הلقנו למבודה וביצענו סריקה מלאה של החדרים הללו בעזרת הרחפן. לאחר סיום הסריקה, הוציאנו את מפות הנקודות שנשמרו מכל החדרים העברנו אותם למחשב האישי ובחנו את האלגוריתם על המפות שהתקבלו. לאחר כל הבדיקות רأינו כי שהרחפן אכן מוצא את הנקרה המתאימה במרקחה זה. יש לציין שבניגוד לקרים הקודמים שבהם קיבלנו מפות ממאגר קיימ ובחילוקן לפעם היה ספק היכן הייצאה המדוייקת.

לאחר שראינו שתוצאות הסימולציה תואמות את המזופה, 'המראנו' את הקוד שלנו הכתוב בשפת MATLAB לקוד בשפת C++ (בסביבה 10 Windows) ובדקנו את נכונותו על פי התוצאות שהתקבלו בתוכנת MATLAB.

<sup>8</sup> המאמר נמצא במקורות [1]

## מימוש:

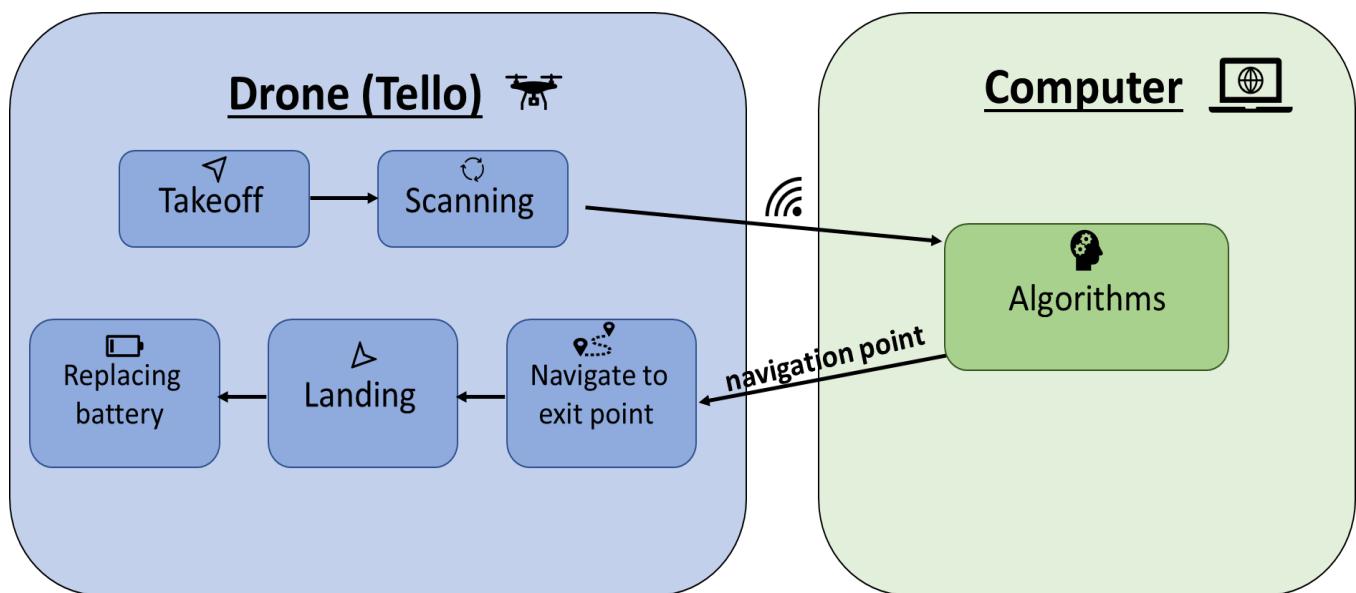
חלק זה הינו מהתאריך פרויקט זה והוא כולל בתוכו את הפעלה של הרחפן בפועל יחד עם האלגוריתם החדש שאנו כותבנו. לאחר מחקר عمמיק ווחשיבת ובוחנה של האלגוריתם החדש שפיתחנו, היה علينا להתחילה ולבחון את פעולה הרחפן יחד עם אלגוריתם זה בזמן אמת ולא רק על מנת נקודות קיימות מהמארגר. חלק זה היה מתאים מאוד בו היינו צריכים לחתוך אלגוריתם שעבוד תיאורטי ולהציגו במצב שהרחפן בפועל אכן יוצאה מהחדר. תחילתה נציג באופן כולל את אופן פיתוח האלגוריתם והטמעתו בדיאגרמה הבאה:



איור 3 – דיאגרמת בלוקים המתארת את אופן תהליך מימוש האלגוריתם שלנו והעברתו למחשב מבוסס מערכת הפעלה LINUX

חלק זה נעשה לאחר הפיתוח הסופי של האלגוריתם על מחשבינו האישיים שתჩילה נכתב בשפת MATLAB ו-PYTHON. לאחר בדיקת האלגוריתם על מספר רב של מפות (שהורדנו מתיקית הפרויקט ב-GitHub) וקבלת אחזוי הצלחה טובים, היה علينا למדוד את שפת C++ ולימוד על מערכת הפעלה LINUX על מנת שנוכל להתאים את האלגוריתם התיאורטי שלנו אל המערכת המעשית בפועל. שלב זה הינו גם כן חלק ממשמעותי בפרויקט מסוים שהמטרה הסופית של פרויקט זה, מלבד שייפור האלגוריתם, הייתה לבחון את הרחפן בזמן אמת ולראות כי אכן ישנה הצלחה מעשית והוא יכול לנוט וליצאת מחדדים באופן אוטונומי לגמר.

נציג כעת את השלבים העיקריים בפעולת הרחפן החל מריגע הדלקתו ועד הרגע שבו הוא מנוט אל עבר היציאה (האלגוריתם שפיתחנו יתואר בחלק מימוש תכנה בהמשך):

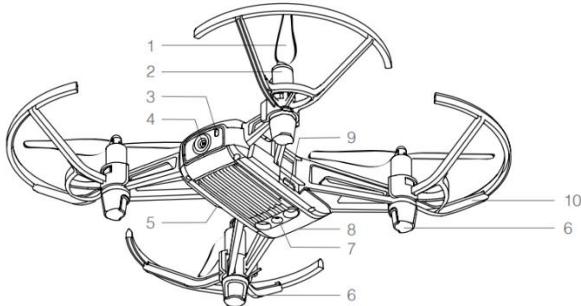


איור 4 – דיאגרמת בלוקים המתארת באופן כללי את השלבים העיקריים בפעולת הרחפן מריגע הדלקתו ועד הסיום

### תיאור חומרה:

החלק העיקרי בפרויקט זה הינו החלק התכנוני שבו אנו מוצאים אלגוריתם המוצא את נקודת היציאה מחדר. עם זאת הפרויקט כולל רחפן שהינו החומרה המרכזית בפרויקט ועל כן נפרט עליו בקצרה. הרחפן בו השתמשנו הינו רחפן של חברת DJI והנקרא Tello:

1. Propellers
2. Motors
3. Aircraft Status Indicator
4. Camera
5. Power Button
6. Antennas
7. Vision Positioning System
8. Flight Battery
9. Micro USB Port
10. Propeller Guards



איור 5 – תיאור הרחפן בו השתמשנו וחלקים העיקריים הנמצאים בו

\*התמונה נלקחה מתוך המפרט הטכני מהאתר המצורף בפרק המקורות<sup>9</sup>

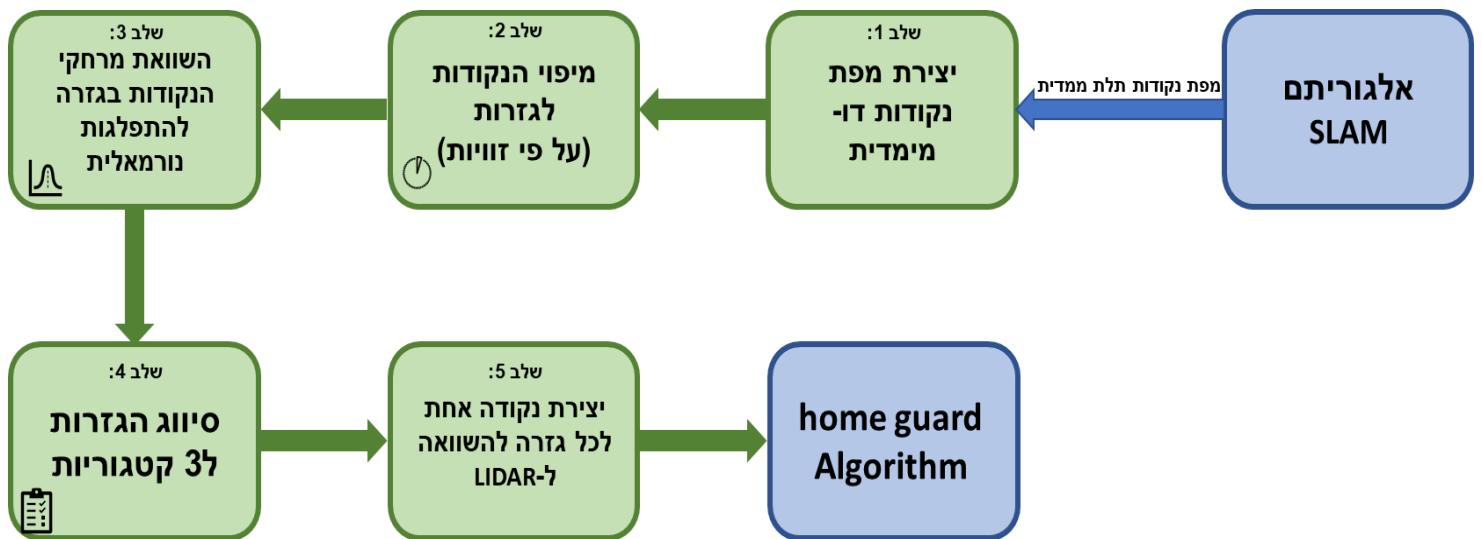
הרחפן מכיל מצלמת RGB קדמית שבעזרתה מבוצעת הסריקה של החדר. בנוסף, הוא כולל חיישן INFRARED בתחתית שיעזר ליצב את הרחפן ולשמור על מיקומו הוא כולל ארבעה מנועים פרופולרים ובטריות חלופיות. זהו רחפן קטן משקלו 87 גרם שעלוותו לא גבוהה מאוד (500 ש"ח) והוא בעל זמן טיסה מקסימלי של 13 דקות. על כן במהלך הפרויקט הינו צרכים להתחשב בביצועיו של הרחפן על מנת שישפיך לסרוק את החדר ולנות אל היציאה לפניו שהסולות שלו יגמרו. בנוסף היה علينا לדאוג לשינויו ליציאה באופן בטוח ולא יתקל במכשולים על מנת שלא יחרס.

חלק חומרתי נוסף הינו המחשב עליי רץ הקוד המפעיל את הרחפן. נציין שבתחילת העבודה, המחשב החיצוני בו השתמשנו היה RP. המטרה הייתה, להשתמש במחשב פשוט וזול, עם מערכת הפעלה XUNIN וקר לתקשר עם הרחפן. אך במהלך העבודה רأינו כי מאחר והמחשב RP איטי מאוד מעבד חזק בהרבה מה-RP שעליו הותקנה מערכת ההפעלה XUNIN ועליו כtabנו את הקוד של האלגוריתמים שלנו. בסוף התהליך המחשב החיצוני הנבחר היה המחשב הניד בעל מעבד חזק שאיפשר לקודים המתאימים לרווח ולבצע את עובdetם באופן טוב וכן גם התקשרות מול הרחפן בזעפה באופן מספק.

<sup>9</sup> המפרט הטכני נמצא בקישור הנמצא במקורות [12]

### תיאור תוכנה:

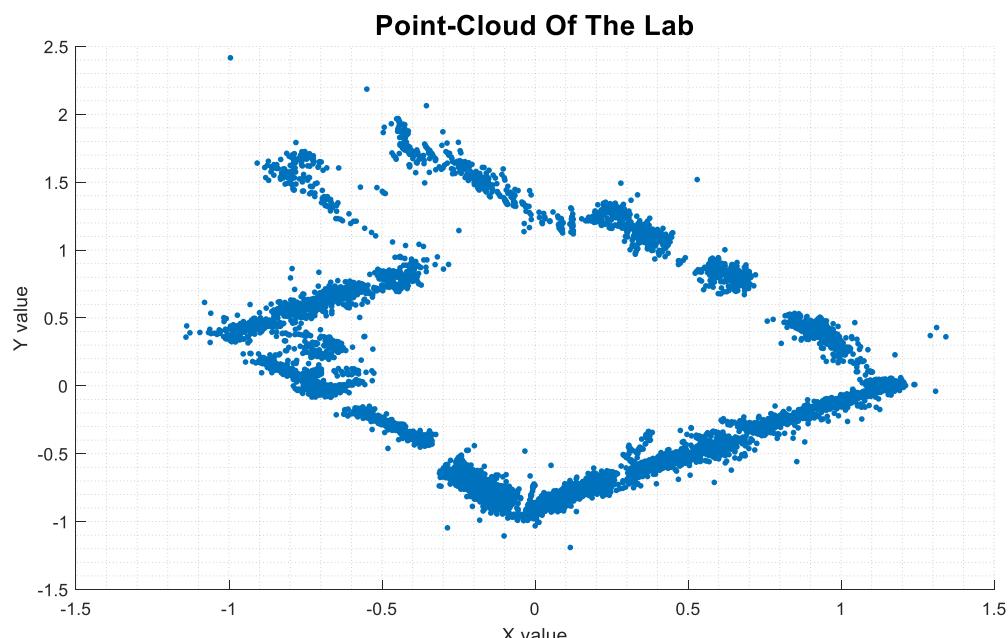
לאחר סיום סריקת הרחפן, מתקבלת מפת הנקודות מאלגוריתם ORB-SLAM2. מרגע קבלת מפה זו מתבצע האלגוריתם שמטרתו למצוא את היציאה ואוטו אותנו פיתחנו והוא מורכב ממספר שלבים. סיום של אלגוריתם זה נעזר באלגוריתם *home guard* שמוצג ברקע התיאורתי. תחילה נציג דיאגרמת בלוקים המתארת את שלבי האלגוריתם ולאחר מכן נפרט את המהות והעקרונות העומדים מאחורי כל שלב. (נציין שהחלקים הצבועים בירוק הם החלקים באלגוריתם אותם אנו כתבו)



איור 6 דיאגרמת בלוקים של תיאור שלבי האלגוריתם שכתבנו

#### 1. מציאת מפת הנקודות הדו-ממדית

ראשית, אנו מקבלים את הנקודות המתאorate אובייקטים בחדר כאשר כל נקודה מתוארת על ידי 3 קואורדינטות קרטזיות ביחס למיקומו של הרחפן למרחב (זהי הנקודה (0,0,0)). הקירוב הראשון שנעשה על מנת לפשט את הניתוח הינו להזניח את ציר z, ולהניח כי הרחפן נמצא בגובה בו הוא יכול לצאת מה חדר ללא הפרעה. הזנחה זו אינה פוגעת משמעותית בתוצאות ומטרתה להפוך את הבעיה לבעה דו-ממדית. נציג את מפת הנקודות הדו-ממדית של המעבדה:



איור 7 מפת נקודות המתקבלת מסריקה במעבדה

## 2. מיפוי הנקודות לגזרות

לאחר השלב הראשון יש בידנו דוח מדידת המכילה נקודות מפוזרות לפי ערכי  $X$  ו- $Z$  שליהן. כעת, נרצה לסואג את הנקודות הללו לגזרות על פי הזווית בין בין החלק החיבובי של ציר  $X$ - $Z$  לבין נקודה  $(0,0)$ . תחילה, עלינו לקבוע את מספר הגזרות שאנו רוצים ולשם כך

נקבע את הרזולוציה של החלוקה שאנו רוצים ונסמן  $\text{ang\_res}$  ומכאן מספר הגזרות הינו

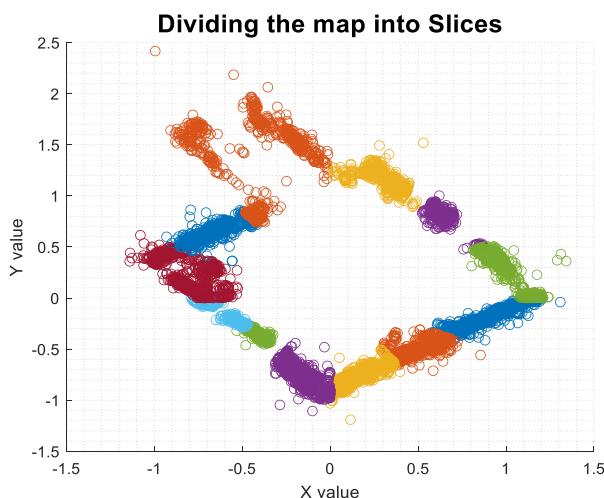
$$\frac{360}{\text{ang\_res}}$$

לאחר מכן, נמיר את ייצוג הנקודות מקואורדינטות קרטזיות  $(p_x, p_y)$ , לקואורדינטות פולריות ונסואג כל נקודה לגזרה המתאימה לה על פי הנוסחה:

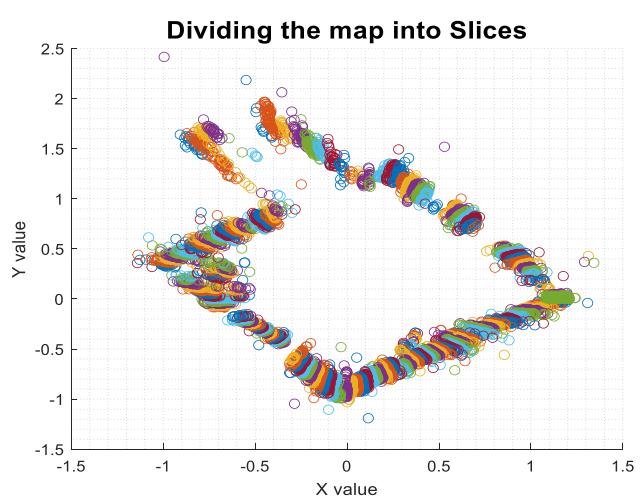
$$\text{sliceNum} = \left\lfloor \left( \arctan \left( \frac{p_y}{p_x} \right) + 360 \right) \% 360 \right\rfloor / \text{ang\_res} \quad (3)$$

בסיום שלב זה, הנקודות במרחב מוחלקות לקבוצות על פי הזווית שלן ביחס לחלקו החיבובי של ציר  $X$  ככלمر, הזווית ביחס לרוחפן.

הकושי בשלב זה הינו בקביעת ערך הפרמטר  $\text{ang\_res}$  משומש שמצד אחד נרצה שהוא יהיה קטן ככל האפשר על מנת לקבל רזולוציה טוביה של החדר, ומצד שני הוא קטן מאוד מקבל המונע גזרות ריקות או בעלות מספר נמוך של נקודות מה שיכל להביא לסתירות רבות וטעויות בחישובים בהמשך האלגוריתם. נציג מספר גרפים המציגים את הגזרות המתפלבות כתלות בפרמטר  $\text{ang\_res}$ :



איור 8.2 חלוקה לגזרות עם הערך  $\text{angle\_res} = 30^\circ$



איור 8.1 חלוקה לגזרות עם הערך  $\text{angle\_res} = 2^\circ$

נשים לב כי עברו החלוקה עם הערך  $\text{angle\_res} = 30^\circ$  בכל גזרה יש נקודות רבות, שכן כאשר נרצה למצוא את התפלגות הנורמללית המתאימה לכל גזרה נקבל סטיות גדולות בערכי השונות משומש שהגזרה מכילה חלק גדול מהחדר (לדוגמא אם הגזרה נמצאת על פינת החדר השוננות תושפף). בנוסף, נשים לב שעבור חלוקה לגזרות גדולות, נקודות היציאה שתתקבל לא תהיה מדויקת כל שכן כל גזרה מתארת חדר רחב מהחדר (אך יותר מגודל הדלת) ולכן גם אם נבחר את הגזרה הנכונה קיימים סיכוי בו עדין לא נמצא את הנקודה הנכונה. מכאן הסקנו כי ככל שנקטין את גודל הגזרה (כלומר הרבה גזרות קטנות), נקבל תוצאות מתאימות יותר למציאות. אך, אם נקטין מדי, נקבל שיש מספר מועט מדי של נקודות בגזרה וניהיה מאוד רגילים לסתירות. עקב נקודות לא מציאותיות שיישפיעו מאוד על התוצאות. על כן, לאחר מספר בדיקות וניסויים והערך האידיאלי אליו הגיענו הינו  $\text{angle\_res} = 2^\circ$ .

### 3. השוואת התפלגות הנקודות בכל גזירה להטפלגות נורמלית

שלב זה נעשה כל גזירה בנפרד, ללא תלות בגזרות האחרות. עבור כל הנקודות בגזרה מסוימת, חושב המרחק האוקלידי של הנקודה מהרכף שנמצא בראשית הצירים. עבור כל המרחקים המוחושבים בכל גזירה, ביצענו התאמת להטפלגות הנורמלית הטובה ביותר. ככלומר היה לנו למצוא את התוחלת והשונות של הנקודות.

המשער בו השתמשנו עבור התוחלת הינו הממוצע של המרחקים:

$$\hat{\mu} = \frac{1}{n} \sum_{i=0}^{n-1} d_i , \text{ where } n \text{ is the number of points, and } d_i \text{ is each point distance. } \quad (4)$$

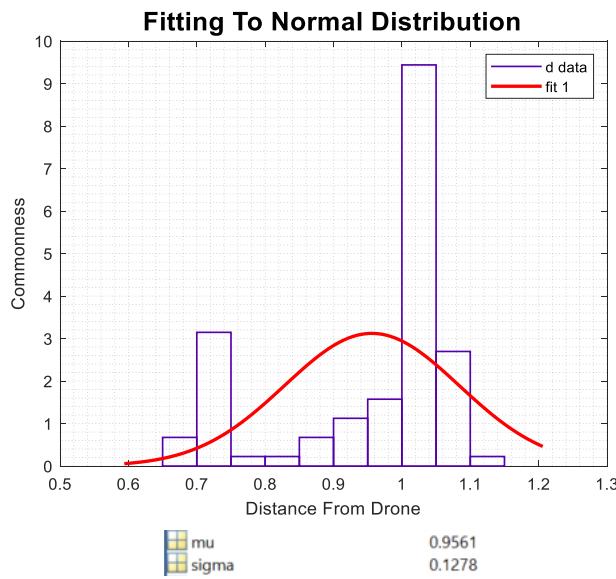
ובעבור השונות השתמשנו במשער הלא מוטה:<sup>10</sup>

$$\hat{\sigma} = \sqrt{\frac{1}{n-1.5} \cdot \sum_{i=0}^{n-1} (d_i - \hat{\mu})^2} \quad (5)$$

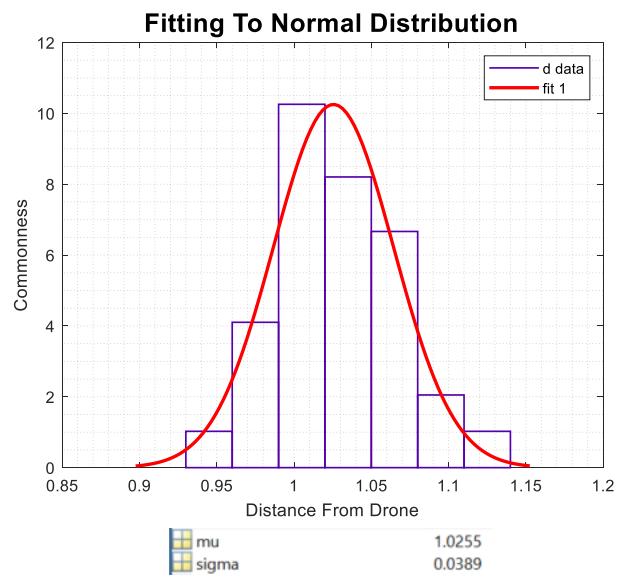
ובעזרת פרמטרים אלו נוכל לתאר את ההטפלגות הנורמלית של המרחקים בכל גזירה בטור

$$distance \sim N(\hat{\mu}, \hat{\sigma}^2) \quad (6)$$

נציג דוגמאות לגזרות המתאימות את היציאה לעומת גזרות המתאירות קיר:



איור 9.2 – התפלגות המרחקים עבור גזירה המכילה את היציאה



איור 9.1 – התפלגות המרחקים עבור גזירה שנייה מכילה את היציאה

ונכל לראות כי אכן בגזרה שבה היציאה נמצאת, ערך השונות גבוהה בערך פי 3 מערכת כאשר מדובר בגזרה ללא היציאה.

בסיום שלב זה, אנו מקבלים עבור כל גזירה את ההטפלגות הנורמלית המתאימה להטפלגות מרחקיה הנקודות מהרכף המתואמת על ידי התוחלת והשונות.

<sup>10</sup> קישור לאופן ביצוע החישוב נמצא במקורות [7]

#### 4. סיווג כל גזרה

בשלב זה הגזרות חולקו ל 3 קבוצות שונות:

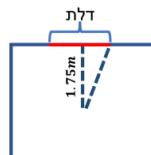
- גזרות רגילות
- גזרות בעלות התפלגות עם שנות "גבוהה" (בஹשך נסbir את ההגדירה ל-"גבוהה")
- גזרות ריקות – אלו גזרות בהן אין נקודות כלל

סיווג הגזרות בעלות השנות הגבוהה נעשה באופן יחסי, כלומר אחוז מסוים של גזרות מסך הגזרות שסומן בתור *precent* הגדרת את מספר הגזרות בעלות השנות הגבוהה . כלומר מספר הגזרות בקבוצה השנייה יהיה:

$$\text{high variance slices number} = \frac{360}{\text{angle\_res}} \cdot \text{precent}, \text{precent} \in [0,1] \quad (7)$$

ועבורן שנות התפלגות המרתקים גבוהה משאר הגזרות.  
על מנת לקבוע את מספר הגזרות בעלות השנות הגבוהה, נצטרך לבחור את המשתנה *precent*.  
משתנה זה מגדר כמה אחוז מurface החדר הוא להערכתנו דלת היציאה, המשתנה נבחר באופן הבא:

גודלו של חדר ממוצע הוא בערך  $3.5m \times 3.5m = 12.25m^2$  וגודלה של דלת ממוצעת הוא בטווח ( $55_{cm}, 65_{cm}$ ) כלומר תמנונת החדר היא להלן:



איור 10 – המכשה על אופן בחרת הערך *precent*

כך שמאחר ואני יודעים את הגודלים של הדלת ושל הניצב אליה, אנו יכולים באופן פשוט לחשב את גודל הזרויות שהדלת מוכלת בה (מרכז החדר):

$$\theta_{door} = 2 * \tan^{-1} \left( \frac{30_{cm}}{175_{cm}} \right) \approx 20^\circ \quad (8)$$

כלומר הזרויות של הדלת מסך החדר מhoeה  $= \frac{20}{360} = 5.4\%$  . על כן, ביצענו מספר בדיקות בסימולציה לערך *precent* סביר  $5.4\%$  (הוא כמובן היה ערך משוער ממשום שלא תמיד הדלת נמצאת במרכז החדר, הגודלים יכולים להיות שונים וכו') והמספר האופטימלי שהגענו אליו הינו  $0.05$  *precent*.

העקרון העומד מאחורי שני השלבים האחרונים הינו שעבור האזורי בפתח הנקודות החשוד כיציאה, ישנו "שביל" של נקודות עקב הימצאות מספר רב של אובייקטים במרקטים שונים לאורך הדריך (מהדלת כלפי חוץ). לעומת זאת, כאשר ישנו קיר או מקום אחר שנויות התפלגות של הנקודות הינה קטנה משמעותית (משמעותו הרבה הנקודות יהיו מרכזיות על הקיר עצמו ללא 'שביל'). על כן בחרנו להשוו את התפלגות של הנקודות לתפלגות נורמלית. וכן גזרות עם שנות גבוהה ייחסו כיציאה גזרות עם שנות נמוכה יהיו לרוב חלק מקירות החדר.

#### 5. ייצוג כל גזרה בעזרת נקודה בודדת על מנת למפות את החדר

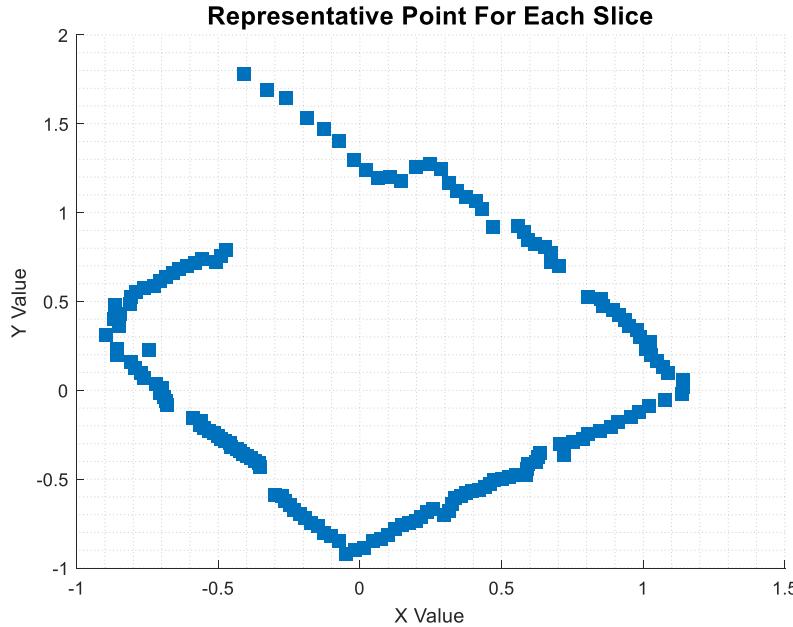
cutut הטיפול בכל קבוצת גזרות נעשה בצורה שונה לגמרי.

- **גזרות 'רגילות':**

עבור קבוצת גזרות זו, נבחר את הנקודה המייצגת את כל הנקודות בגזרה זו על פי התוחלת ומספר הגזרה (שמתאר את הזרויות של הגזרה). כלומר הנקודה המייצגת של גזרה  $i$  בקואורדינטות פולריות תהיה  $(r_i, \theta_i)$  כך:

$$r_i = \hat{\mu}_i \quad , \quad \theta_i = sliceNum * angle_{res} + \frac{angle_{res}}{2} \quad (9)$$

נציג גרף המתאר את הנקודות המיצגות עבור הגזרות הרגליות:



איור 11 – גרפ המתאר את הנקודות לכל גזרה, עבור הגזרות הרגליות

#### גזרות ללא נקודות כלל:

- עבור גזרות אלו, נרצה לבצע שערור של הנקודה המיצגת את גזרה זו בעזרת המידע מגזרות אחרות. הרעיון העומד מאחורי השלמה זו הוא שעקב רציפות החדר אותו אנו סורקים והסימטריה שקיימת בו מוכן להסיק בעזרת הגזרות "השכנות" והסימטריות מידע על גזרה בה לא קיבלנו נקודות כלל מאלגוריתם SLAM. גזרה זו היא ככל הנראה חלק מהחדר שאין בו אובייקטים כלל ולמעשה היא ככל הנראה מתארת קיר שזהו מקום שאנו לא רוצים שהרחפן יגיע אליו.

נתאר את ביצוע השלמה עבור גזרה ריקה  $i$ . לשם כך יהיה علينا למצוא מספר גדלים:

- התוחלת של הגזרה הקודמת:  $\hat{\mu}_{i-1}$
- הגזרות הסימטריות של גזרות  $1 - i$ , שנסמן בתור  $1 - j$ , בהתאם. הגזרה הסימטרית- היא הגזרה שהפרש הזרויות בין לה בין הגזרה שעבורה נחפש את הסימטרית הינה  $180^\circ$  (לדוגמה עבור גזרה בעלת זווית  $90^\circ$  הגזרה הסימטרית הינה גזרה בעלת זווית של  $270^\circ$ ).
- התוחלת של הגזרות הקודמות והסימטריות:  $\hat{\mu}_j, \hat{\mu}_{j-1}$
- יחס ההמרה:  $ratio\ factor = \frac{\hat{\mu}_{i-1}}{\hat{\mu}_{j-1}}$

ובעזרת הגדים הללו נשלים את התוחלת של הגזרה הנוכחית להיות:

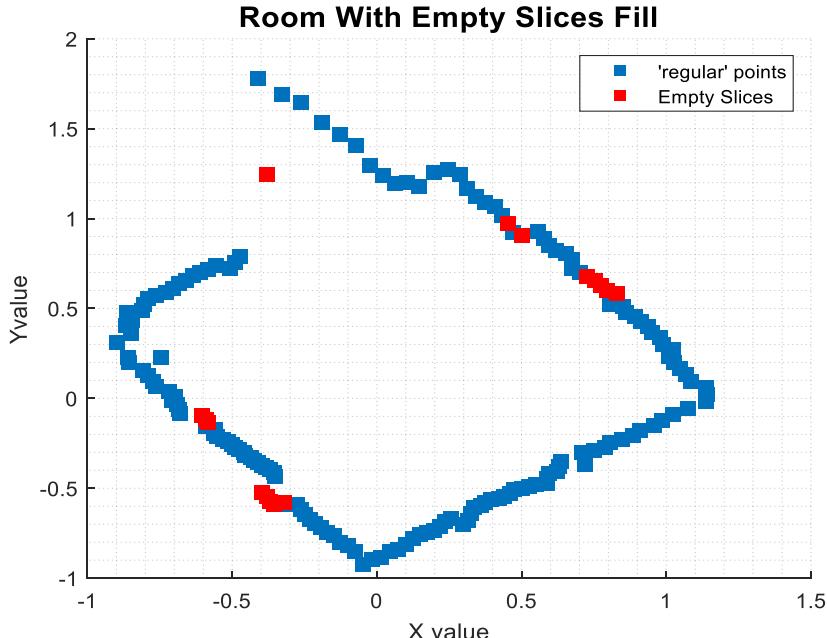
$$\hat{\mu}_i = \hat{\mu}_j \cdot ratio\ factor \quad (10)$$

במידה ואחד מהגדלים הללו הינו 0, נבחר להשלים את התוחלת של הגזרה הנוכחית בעזרת התוחלת של הגזרה הקודמת כולם:

$$\hat{\mu}_i = \hat{\mu}_{i-1} \quad (11)$$

ולבסוף גם במקרה זה נגיע לנקודה בקואורדינטות פולריות המתארת את גזרה זו בדיק כמו שתואר מוקדם.

מציג כעת את הגרף שמתקיים המכיל את הנקודות המייצגות עבורי הגזרות הרגילות והגזרות הריקות:



איור 12 – גרף המתאר את הנקודות לכל גזרה, עבורי הגזרות הריקות

כפי שניתן לראות ההשלמה מתבצעת כפי שציפינו ואכן הגזרות הריקות מושלמות בצורה טוביה מאוד ואנו מקבלים קיר כמעט רציף אחד. על כן נסיק שחלק זה עובד כפי שנרצה והוא יעזר לנו להבדיל בין גזרות אלו לבין הגזרות המתארות יציאה באופן ברור יותר. נשים לב שחלק זה של האלגוריתם שכתבנו חשוב מאוד למרקמים בהם הסריקה לא מתבצעת ממרכז החדר אלא מנקודה פינטנית יחסית, שכן במצב כזה האלגוריתם יודע להשלים (במידה וישנם גזרות ריקות) באופן טוב מאוד את החדר כדי לקבל סריקה טובה!

#### גזרות בעלות שונות "גובהה"

- גזרות אלו הן גזרות שחשודות בתור כיוון היוצרת של הרחפן מהחדר. על מנת שנוכל להשתמש במידע זה בשלב הבא של האלגוריתם علينا לבחור עבורי גזרות אלו נקודה מרוחקת מאוד מהרחפן (מרכז הczytym) על מנת להבדיל אותן משאר הנקודות שמתארות את קירות החדר. בנוסף, בשימוש ב-*angle\_res* נמוך (כלומר ישנה חלוקה להרבה גזרות), נצפה כי היוצרת תואר על ידי מספר גזרות רצופות. על כן עבורי גזרות אלו, ערכיה של התוחלת המשוערת בכל גזרה (שהיה זהה לכל הגזרות הרצופות) יהיה ממוצע של המרחקים הגדולים ביותר בכל הגזרות הרצופות הללו. לדוגמה, עבור רצף של *n* גזרות, מתוך כל הגזרות בקבוצה זו, שמהוות סדרה עולה ורציפה של *sliceNum* הערך של התוחלת יהיה:

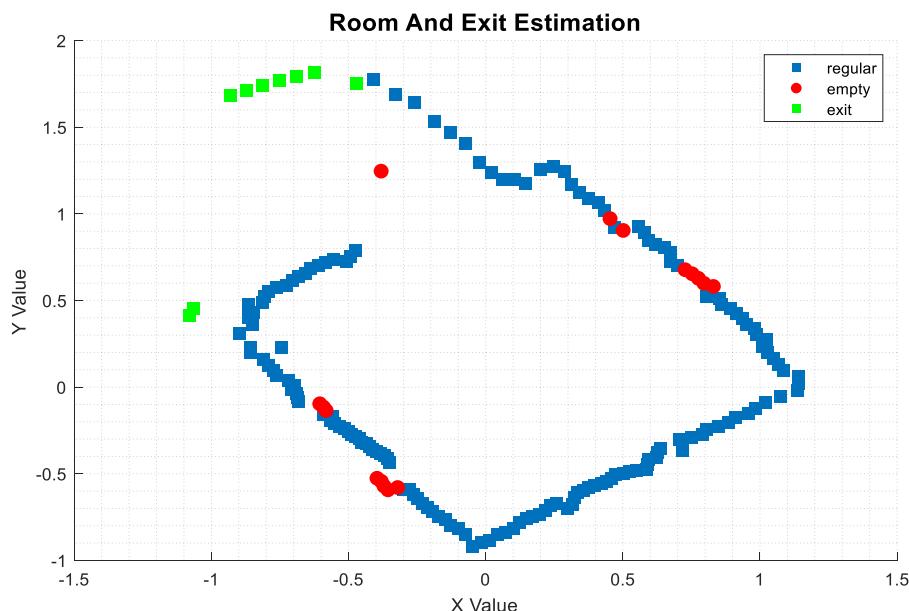
$$\hat{\mu}_k = \frac{1}{n} \sum_{i=0}^{n-1} d_{m,i} \quad , \text{where } d_{m,i} \text{ is the maximum distance in slice} \quad (12)$$

\* חשוב לציין כי *i* אינו ה-*sliceNum* אלא רק משתנה איטרציה מייצג את הספירה של רצף *n* הגזרות.

את הנקודות האלה שמנן באופן מכון רחוק יותר משאר הנקודות משום שבאופן זה נוכל להשוו את הסריקה לסדרת *LIDAR* בצורה הטובה ביותר.

ולבסוף גם במקרה זה נגיע לנקודות בקואורדינטות פולריות המתארות כל אחת מהגזרות, בדיקן כמו קודם.

מציג את הגרף הסופי הכלול את הנקודות המיצגות של כל שלושת סוגי הגזרות:



איור 13 – גראף המתאר את הנקודות עבור שלושת הגזרות וכן- הצגת החדר כסריקת LIDAR

בסיומו של שלב זה, עברו כל גזרה נקודה בקואורדינטות פולריות אשר מייצגת את האפיקון של הגזרה. המטרה של נקודות אלו לקובוארדינטות קרטזיות והציגן הינה דרך לתאר את החדר אותו הרחפן סרף כפי שניתן לראות באירוע.

בעזרת השלב הזה יצרנו באמצעות סריקת  ${}^{\circ}360$  של החדר על ידי מצלמת *RGB* בודדת מייפוי של החדר אשר זהה במופיעינו ובצורתו לסריקה שמבצעת על ידי *LIDAR*<sup>11</sup>.

## 6. שימוש באלגוריתם *Home Guard*<sup>12</sup>:

השלב האחרון הינו מציאת נקודות היציאה שנעשתה בעזרת שימוש בפלט מהשלב הקודם. מכיוון שקיבלנו תמונה משוערת של החדר ובה בכל גזרה ישנה נקודה אחת שמייצגת את המתאר של החדר, נוכל לנצל זאת על מנת להשתמש באלגוריתם המוצא נקודות יציאה מסריקת *LIDAR* שתואר ברקע התיאורתי.

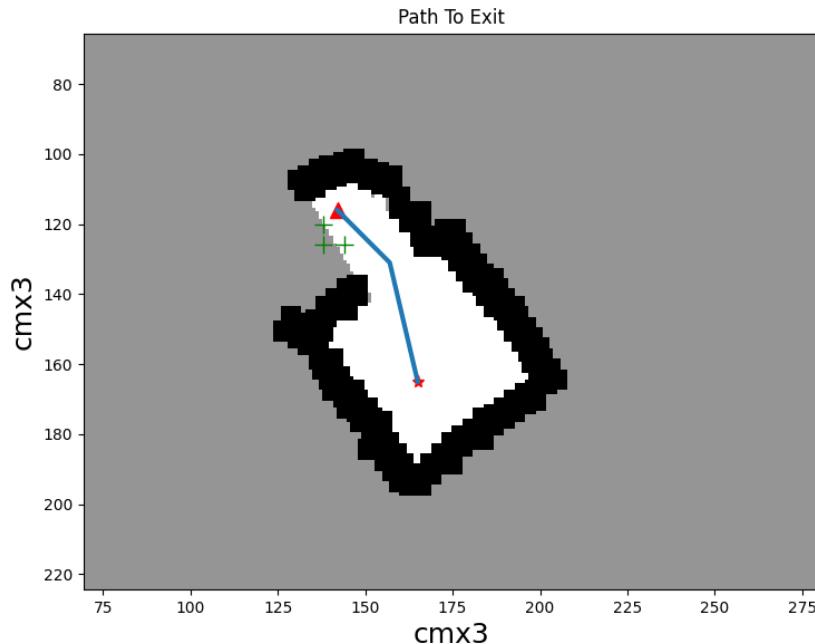
אל אלגוריתם זה נכנס את הנקודות שקיבלנו בסוף שלב הקודם, ונקלט בסופו של דבר את מסלול היציאה של הרחפן מהחדר. המסלול יתואר על ידי מקטעים ליניארים כך שנתקבל בモץ נקודות של צוית ומרחק, ביחס לנקודה הקודמת. בין כל 2 נקודות כאלו עבר קו ישר של המסלול (כך שהנקודה הראשונה היא יחסית לנקודה התחלה של הרחפן- ראשית הצירום).

ההצלחה של מציאת נתיב היציאה בשלב זה בעצם מראה כי אכן הצליחו להראות כי בעזרת מצלמת *RGB* בודדת אנו יכולים למפות את החדר כפי שניתן באמצעות סריקת *LIDAR*.

<sup>11</sup> הסבר נרחב מעט יותר על אופן ה- *LIDAR* נמצא במקורות [9]

<sup>12</sup> המאמר המתאר את האלגוריתם נמצא במקורות [1]

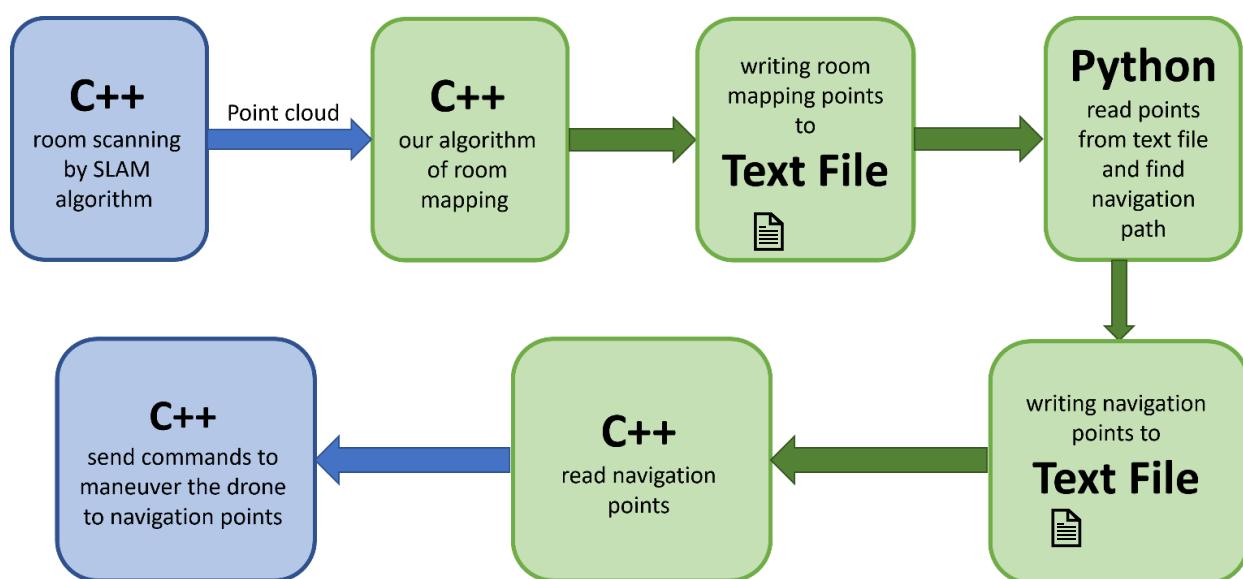
נציג כעת את הגרף המתkeletal לאחר הרצת אלגוריתם זה ובו נכלול את נתיב היציאה של הרחפן מהחדר:



איור 14 – נתיב היציאה המתkeletal (בכחול), נקודת המוצא של הרחפן (כוכב אדום) והנקודה הסופית (כוכב אדום)

כפי שניתן לראות בגרף המתkeletal, אנו רואים את המסלול שעל הרחפן לעבור על מנת להגיע ליציאה. שלב זה הינו השלב האחרון באלגוריתם שלנו ואחריו מתבצע הניוט של הרחפן אל נקודת היציאה.

כעת נתאר את המימוש התוכנוני של הפרויקט בפועל. בחלק היסימולציה הוסבו התכונות והכללים בהם השתמשנו על מנת ליצור את אלגוריתם זה. בשלב המימוש היה علينا לקחת את האלגוריתם בשפת C++ שנכתב על גבי מערכת הפעלה WINDOWS 10 ואת קוד ה-PYTHON ולחוברים אל מערכת הפעלה LINUX שליה רשום כל הפרויקט וכן ניתן לשנות ברחפן. שלב זה לווה במעט קשיים טכניים במעבר למערכת הפעלה אחרת, והאינטגרציה בין קוד C++ לבין PYTHON ולאחר התגברות על מכשולים אלו נציג דיאגרמה המתארת את פעולות המערכת על גבי המחשב הנידי בעל מערכת הפעלה LINUX:



איור 15 – דיאגרמת בלוקים המתארת את אופן ביצוע האלגוריתם בפועל

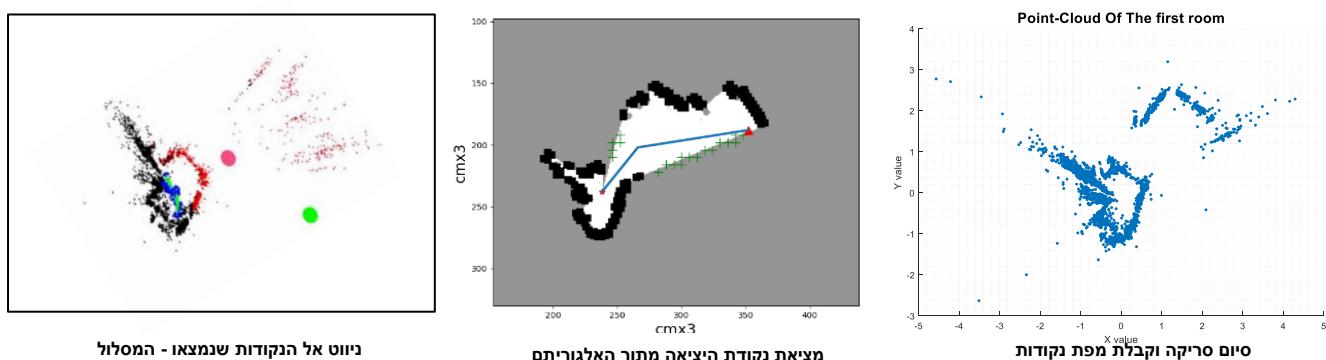
החלקים הירוקים בדיאגרמה, הם החלקים העיקריים שהוספנו אל הפרויקט. לאחר שהרחפן סרק את החדר ע"י שימוש באלגוריתם ORB SLAM2 ובעזרת הקודים הקיימים, מתקבלת מפת נקודות.

הנקודות האלו מועברות ישירות אל חלקו הראשון האלגוריתם שאנו חנכו בתבנו ב++ C ומןנו קיבל נקודות המטארות באופן ברור את החדר (בדומה לנקודות שיתקבלו מסריקת LIDAR, כפי שפורסם בפרקים הקודמים). כתה היה علينا לבצע אינטגרציה לקוד בשפת PYTHON אל הקוד הקיים משום שעד כה הוא אינו היה חלק מהפרויקט הכלול. לכן, תחילת רשםנו את הנקודות המטארות את החדר לקובץ טקסט, ביצענו השהייה של הקוד C++ שרצ, בזמן זה קראנו לקוד ה-NON PYTHON שקרא מקובץ זה את הנקודות והן היו הקלט אליו. לאחר סיום ריצתו, נקודות הייצאה נרשמו אל קובץ טקסט אחר, נגמרה השהייתו של הקוד C++ שקרא מקובץ זה את נקודות הייצאה והעבירן להמשך אלגוריתם. בהמשך האלגוריתם נקודות אלו מתרגמות לפקודות שMOVBROTS לרחפן אל מנת שינוי באופן אוטונומי אל נקודות אלו.

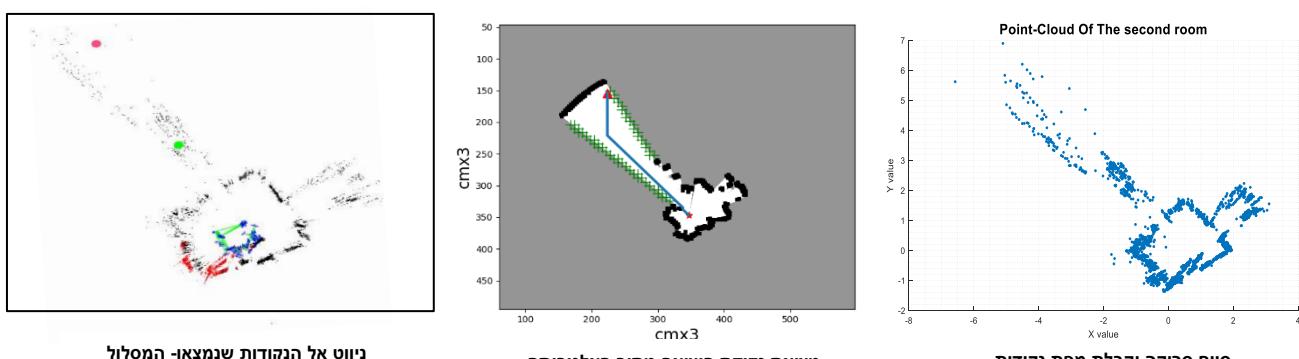
## ניתוח תוצאות:

הדרך הטובה ביותר לבודק את האלגוריתם היא לבדוק את פועלתו בזמן אמת כאשר הרחפן סורק את החדר, מרייך את האלגוריתם החדש ומונוט אל היציאה. הצלחה של האלגוריתם תהיה ניווט מוצלח ובטוח של הרחפן אל נקודת היציאה ללא התקלות במכשורים או קירות. האלגוריתם נבחן ב-3 חדרים שנמצאים בסמוך למעבדת הרחפינים ו בשלושת החדרים הללו קיבלנו כי האלגוריתם מוצא את נקודת היציאה הנכונה ומונוט באופן אוטונומי אל היציאה כפי שרצינו.

נוסף על כך, מטרת הפרויקט הייתה גם כי הרחפן יצא ממספר חדרים ברכיפות בצורה אוטונומית ולא התערבות. בדיקה של האלגוריתם שהתבצעה עבורי 2 חדרים רצופים הראה כי במקרה זה הרחפן אכן יצא מהחדר הראשון, מבצע סריקה של החדר השני ויוצא אל עבר היציאה ממנו גם כן ללא התערבות כלל. על כן נוכל להגיד שזמן אמת אכן עדנו בבדיקה התוצאה והרחפן יצא מהחדר אחד ואך מ-2 חדרים רצופים באופן אוטונומי למגרי<sup>13</sup>. נראה את המפות של שני החדרים הללו מהם יצא הרחפן בהזאה אחר זה (בנספח מצורפת הדוגמה כללית של המבנה אותן אותו הרחפן סרק):



איור 16.1 – יציאה משני חדרים בהזאה אחר זה - חדר ראשון



איור 16.2 – יציאה משני חדרים בהזאה אחר זה - חדר שני

נשים לב שבסריקת החדר השני ניתן לראות חלק קטן מהחדר הראשון שהרחפן יצא ממנו אך לא את כלו. בנוסף, נוכל לראות כי החדר הראשון שנسورק בעל רהיטים רבים כך שהסריקה יוצאה מעט לא בורורה, ובכל זאת האלגוריתם מוצא את נקודת היציאה הנכונה.

<sup>13</sup> מצורף קישור לסרטון המראה את פעולה האלגוריתם היוצא משני חדרים במקורות [13]

**השוואות בין תוצאות הסימולציה לפועל בזמן אמת:**  
 נמchio בutablella הבאה את ההבדלים בין התוצאות המתקבלות בסימולציה (MATLAB) לעומת חדרים נתוניים, לעומת התוצאות המתקבלות עבורה המימוש בפועל בזמן אמת (דרך הקוד ב-C++ וכאשר הרחפן סורק את החדר ומונוט אל נקודת היציאה):

\* תיאור הנקודה הוא (מרחק, דווית)

אחזוי השינוי				זמן-אמת				סימולציה		מספר סריקה
נקודות יציאה 2		נקודות יציאה 1		נקודות יציאה 2		נקודות יציאה 1		נקודות יציאה 2	נקודות יציאה 1	
מרקם	דווית	מרקם	דווית	מרקם	דווית (אם קיימת)	מרקם	דווית (אם קיימת)	(אם קיימת)	(אם קיימת)	
0%	0%	0%	0%	(350.75 , 87.13)	(307.8 , 45.6)	(350.75 , 87.13)	(307.8 , 45.6)	(307.8 , 45.6)	(307.8 , 45.6)	1 (מטבחון)
		3.1%	1702%		(217.92 , 229.43)	(238.08 , 216.45)	(225 , 12.727)			2 (חדר מעבר)
0.47%	1.7%	0.24%	3.17%	(303.69 , 32.45)	(336.037 , 29.54)	(305.13 , 33.015)	(335.22 , 28.63)			3 (חדר בכינוסה למעבדה)

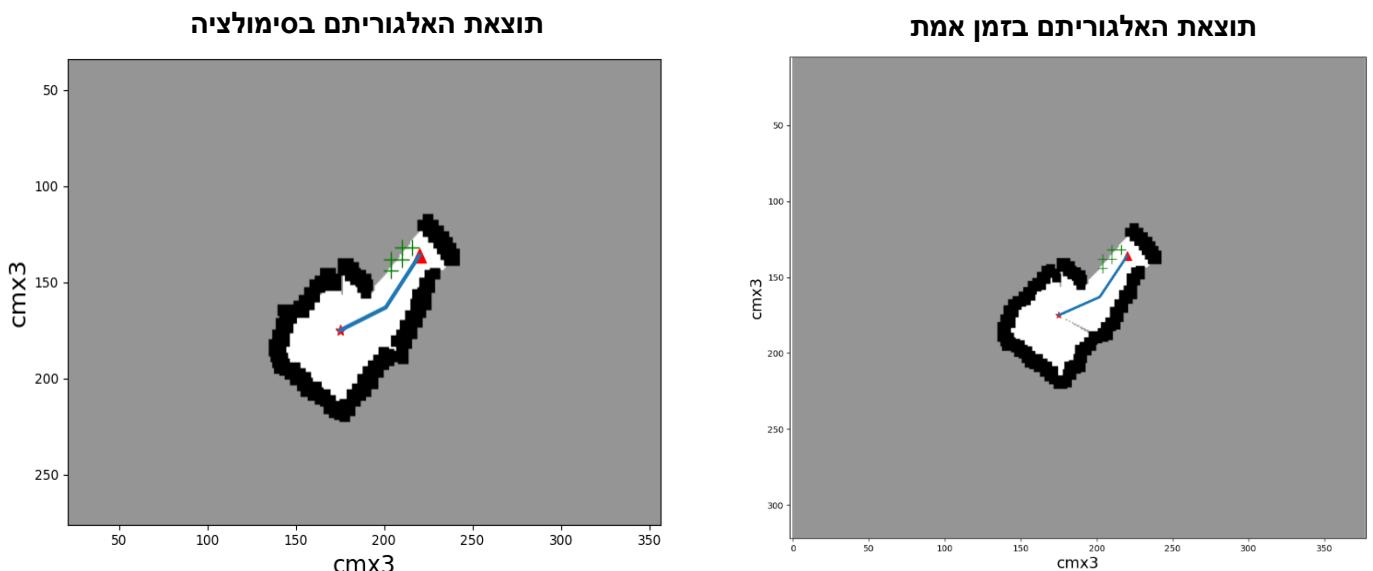
utablella 1 – השוואת הביצועים של הסימולציה מול התוצאות בזמן אמת

הנוסחה בה השתמשנו לצורך חישוב אחוז השינוי הינה:

$$\% = \frac{|RealTime - Simulation|}{Simulation} * 100 \quad (13)$$

נציין כי משלושת החדרים לעיל, עליהם בוצעה המדידה בזמן אמת, הרחפן יצא באופן אוטונומי. כמו כן נשים לב כי כאשר משווים בין הזרויות המתקבלת בזמן אמת לעומת הסימולציה מתקבלים הבדלים קטנים. כאשר השוון בין תוצאות המרחק בשני המקרים, קיבלנו כי במקרה אחד ישנה סטייה גדולה מאוד הנובעת ככל הנראה בעיה בקנה המידה בין החדר בפועל לעומת הסימולציה.

נראה דוגמא לתוצאות הסימולציה והתוצאות בזמן אמת עבור אחד החדרים מהutablella לעיל (חדר 3):



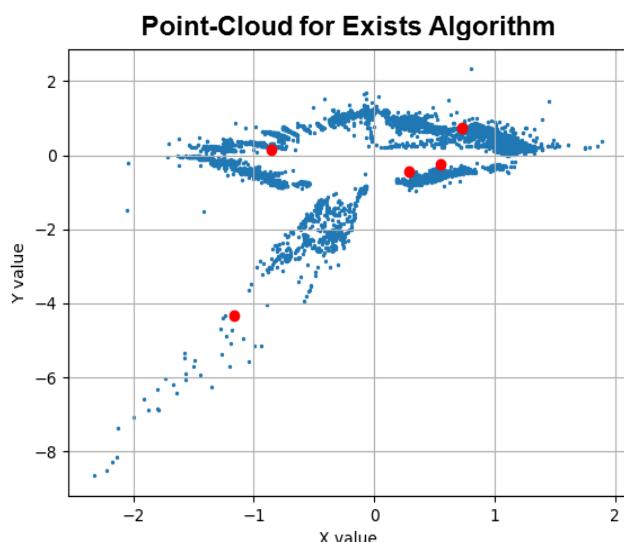
itablella 17 – דוגמא לתוצאות אחד החדרים בהשוואה של תוצאות הסימולציה מול התוצאות בזמן אמת

### השוואות בין תוצאות האלגוריתם שלנו לאלגוריתם הקיים לפני:

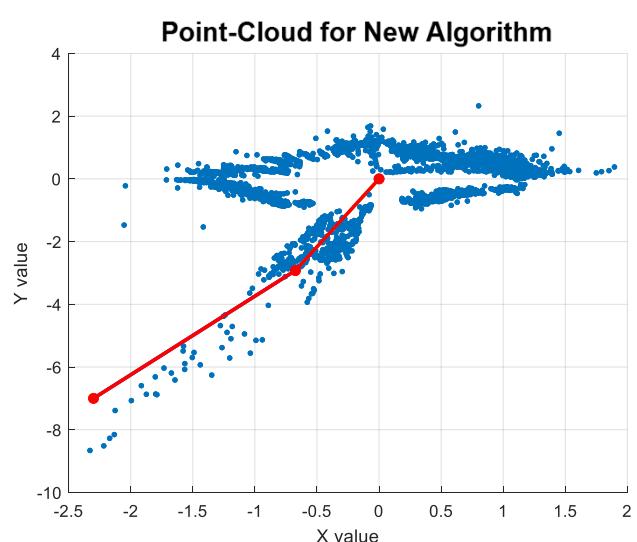
כעת נראה את ההבדלים כאשר אנו מրיצים את האלגוריתם החלופי הנוכחי לעומת האלגוריתם המקורי. לשם כך, לקחנו מפות נקודות קיימות רבות וعليן ביצענו את האלגוריתם שלנו ועבור כל מפה, בדקנו האם האלגוריתם מוצא את היציאה או לא וכן חישבנו את אחוז הצלחה שלו. לאחר בדיקה מקופה על מעל 30 סריקות שונות של חדרים שונים, ראיינו כי האלגוריתם שלנו מוצא את נתיב היציאה הנכון במעלה 70% מהמקרים! כמו כן, בחלק מהמקרים הצלחה היא לא רק ביציאה מחדר אחד אלא היכולת למצוא בחדרים שבהם אחר זה באופן רצוף. נציין שallow אחוז הצלחה של ממש עוטה אחוז הצלחה שהוילו לאלגוריתם הקודם שעמדו על 25%. אחוז הצלחה של האלגוריתם הנוכחי, כבר ניתן לנו מראש (ונכתבו במצבה המוצע) על כן לא היינו צריכים לבדוק אותו.

הסריקות עליון בדקנו את טיבו של האלגוריתם שלנו, מתחלקות לסריקות קיימות ממגרס סריקות רב של חדרים שאינם מוכרים לנו (אלך רק קיבלנו את מפת הנקודות שלהם) וכן סריקות שביצענו באופן עצמאי במעבדת הרחפנים.

נראה דוגמא לתוצאות האלגוריתמים עבור מפת נקודות אחת (מתוך כל המפות שנבדקו):



איור 18.2 – מפת הנקודות עם נקודות החשודות ליציאה עבור האלגוריתם המקורי



איור 18.1 – מפת הנקודות עם דרך היציאה עבור האלגוריתם החדש שתבנו

נוכל לראות כי עבור האלגוריתם הקודם שהיה מתkowski מספר נקודות חשודות ליציאה. נקודות אלו הן נקודות *negative – false* משום שהן מצויות על מקומות המוגדרים כ'יציאה' למטרות שאינן נכון. הבעיה העיקרית הנובעת מכך, היא שהרבה פעמים נבחרת אחת מנקודות *false – positive* להיות היציאה וכאשר הרחפן ינווט לכיוונה הוא ככל הנראה יתקע בקייר ויתרסק (בנוסף לכך שהוא ינווע לכיוון אחר לחלוין מהיציאה).

האלגוריתם החדש לעומת זאת, מחשיך כיון יציאה אחד בלבד ללא מספר אפשרויות. נשים לב כי מעבר למציאת נקודות היציאה, הוא מחשיך את המסלול שעל הרחפן לעبور לצורך הגעה לנקודה זו באופן מהיר וקל (תוקף כדי תשומת לב לקיים ורחייטים שלא יתקע בהם בדרך). יתרון נוסף של האלגוריתם שלנו הינו שהוא אינו רגש לנקודה בה הרחפן נמצא בתחילת הסריקה שכן גם עבור סריקה מפינה של החדר ההשלמה של מפת החדר תבוצע באופן טוב שיאפשר לרחפן למציאת נקודות היציאה במקרה זה גם כן!

## **סיכום מسكنות והצעות להמשך**

המטרה המרכזית של פרויקט זה הייתה לרשום אלגוריתם שינתח את מפת הנקודות המתකבלת מסריקה של הרחפן ויחזר לנו את נקודת היציאה של הרחפן מהחדר הנסרק. הצלחנו לעמוד במטרה זו והאלגוריתם הנכתב אכן נותן תוצאות טובות ומוצא את נקודת היציאה באחוז נכבד מהמפות שבדקנו. נוסף על כך האלגוריתם הנוכחי משפר את האחוז מציאת נקודת היציאה פי 2 מהאלגוריתם המקורי עד כה ולכן אנו סבורים כי עמדנו במטרה המרצכית של פרויקט זה. חוץ משיפור האחוז דהו היציאה המשמעותי, אלגוריתם זה מחייב מסלול ללא מכשולים אל היציאה. יתרון נוסף הינו שאלגוריתם זה מפנה את החדר בצורה טובה מאוד והוא אינו תלוי במקום התחלת הסריקה שכן ההשלמה מתבצעת באופן יחס'י למיקום הרחפן.

בנוסף, הצלחנו לבצע אינטגרציה לקוד אל תוך הרחפן ואף ביצענו ניסויים בחדרים אמיתיים וראינו את הרחפן יוצא מן החדר. יתרה מזאת, הרחפן אף סרק חדר אחד ומוצא את נקודות הייצאה ממנו, התקדם ממנו אל הייצאה ולאחר מכן סרק את החדר אליו הגיעו ואף יצא גם ממנו כל זאת באופן אוטונומי מלא. המטרה היחידה שאיננה מולאה בפרויקט הינה יימוש האלגוריתם על גבי ה-RP.

מטרה זו לא בוצעה עקב בעיות טכניות מסוימות שה-RP אינו מחשב בעל מפרט מספק שעליו יכולנו למשוך אלגוריתם זה ועל כן מימשנו אותו על מחשב בעל נתונים טובים יותר. נציג כי בעבר אכן יתכן היה ליצור אופטימיזציה לקוד ואך לכל הפרויקט על מנת להפעיל את הרחפן באמצעות RP.

על מנת לשפר את המערכת הנווכחית שתעבד בצורה טובה אף יותר ניתן לנסוט ולבצע שילוב בין האלגוריתם החדש לאלגוריתם הנוכחי ולנסות ליצור מערכת חדשה שתקבל החלטות על סמך שני/algo<sup>2</sup> האלגוריתמים הללו אשר מסתמכים על כלים שונים לניתוח הנזקודות. בנוסף ניתן לנסוט, בעזרה שני/algo<sup>2</sup> הסריקה הקיימת, למצוא דרך על מנת לקבל את ממד' החדר ובכך נוכל לדעת את גודל החדר ולקבל נזקודות בעלות משמעות אמיתית שבאזורתן נוכל גם לקבוע את המשטנה *precedent* ואף נוכל להיעזר בפרק *benign* הרחפן ליציאה.

ישן אפשרות רבתה להמשך העבודה על פרויקט זה במספר כיוונים רבים. ראשית, כפי שהוזכר קודם לכן ניתן לנסוט לשפר את האלגוריתם הנוכחיים ואף לשלבו עם מספר אלגוריתמים קיימים/חדשים אחרים. אפשרות נוספת נספתח הינה להסתכל על מפת הנקודות ה-3-ממדית המתkbלת מהסրיקה הראשונית ולמצוא דרכי שבahn המידע מציר Z יכול לעזור לנו לנתח את צורת החדר והמכשולים הנמצאים בו. כלומר, במקרה הפיכת מפת הנקודות לדו-ממדית כפי שנעשה באלגוריתם זה ניתן למצוא דרכי לנצל את מלאה המידע המתkbבל מהסרייקה. אופציה נוספת הינה להוסיף לאלגוריתם אופציה של 'זיכרון' של הסרייקות הקודמות וכך אליו יוכל למפות מספר רב של חדרים / קומה שלמה בעזרת הרחפן והוא יוכל לנוט בכל רחבי הקומה ולסרוק אותה. אפשרות נוספת הינה להרכיב על רחפן זה חישון LIDAR ולבצע בו זמנית סריקה של החדר בעזרת מצלמת RGB ובעזרת חישון LIDAR ולחזון את הסרייקות המתkbבלות והתאמתן האחת לשניה על פניו מספר של חדרים. כמו כן מחקר עתידי יכול לנסוט לשלב את מפת הנקודות המתkbلت יחד עם הידע או המתkbבל מהצלמה ולבנות אלגוריתם משולב המתבסס על אלגוריתם SLAM ועל אלגוריתמים שמסתמכים על ניתוח תМОנות/הידע וכן בנוסף למציאת היציאה מהחדר וסרייקתו, הרחפן יוכל גם לזהות חפצים/אנשים ולהשתמש במידע לצרכים שונים.

תיעוד הפרויקט

הפרויקט מתועד כלו באתר GitHub.

- **תיעוד הפרויקט נמצא באתר GitHub בקישור:**  
<https://github.com/ronfrimmer/autonomous-drone>
  - **בתיעוד הפרויקט באתר נמצאים קבצים המכילים את החלקים שאנו חנו הוספנו לפרויקט (משום ששאר הקוד שייר לחברות ולא ניתן להעלותו), ביניהם:**
    1. **הסימולציות בתוכנת MATLAB נמצאות בתיקייה**  
matlab\_scripts
    2. **קוד ה-C++ נמצא בתיקייה** C++\_scripts
    3. **מסמך דוקומנטציה** לפרויקט המתאר באופן מפורט וברור את הجاد והפונקציות שאנו חנו כתובנו ושם documentation

## מקורות

### מאמריפ:

- [1] D.Magazinnik, I.Klein and D. Feldman, "Home Guard Drone", manuscript.
- [2] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," in IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262, Oct. 2017, doi: 10.1109/TRO.2017.2705103.
- [3] F. Duchon, A. Babinec, M. Kajan, P. Be'no, M. Florek, T. Fico, and L. Jurisica, "Path planning with modified a star algorithm for a mobile robot," Procedia Engineering, vol. 96, pp. 59–69, 2014
- [4] K. MacTavish and T. D. Barfoot, "At all Costs: A Comparison of Robust Cost Functions for Camera Correspondence Outliers," 2015 12th Conference on Computer and Robot Vision, Halifax, NS, Canada, 2015, pp. 62-69, doi: 10.1109/CRV.2015.52
- [5] S. Thrun, W. Burgard, and D. Fox, "Probabilistic robotics," Kybernetes, 2006

### מקורות מהאינטרנט:

- [6] [https://en.wikipedia.org/wiki/Simultaneous\\_localization\\_and\\_mapping](https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping)
- [7] [https://en.wikipedia.org/wiki/Unbiased\\_estimation\\_of\\_standard\\_deviation](https://en.wikipedia.org/wiki/Unbiased_estimation_of_standard_deviation)
- [8] [https://en.wikipedia.org/wiki/Occupancy\\_grid\\_mapping](https://en.wikipedia.org/wiki/Occupancy_grid_mapping)
- [9] <https://en.wikipedia.org/wiki/Lidar>
- [10] [https://en.wikipedia.org/wiki/Time-of-flight\\_camera](https://en.wikipedia.org/wiki/Time-of-flight_camera)
- [11] [https://en.wikipedia.org/wiki/Huber\\_loss](https://en.wikipedia.org/wiki/Huber_loss)

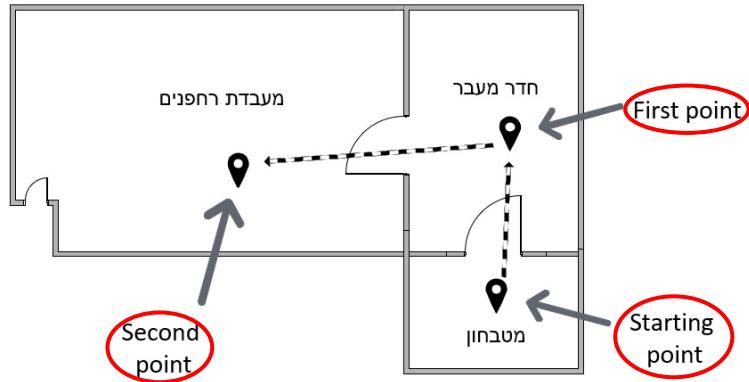
### רכיבים:

- [12] [https://dlcdn.ryzerobotics.com/downloads/Tello/20180404/Tello\\_User\\_Manual\\_V1.2\\_EN.pdf](https://dlcdn.ryzerobotics.com/downloads/Tello/20180404/Tello_User_Manual_V1.2_EN.pdf)

### סרטוניים:

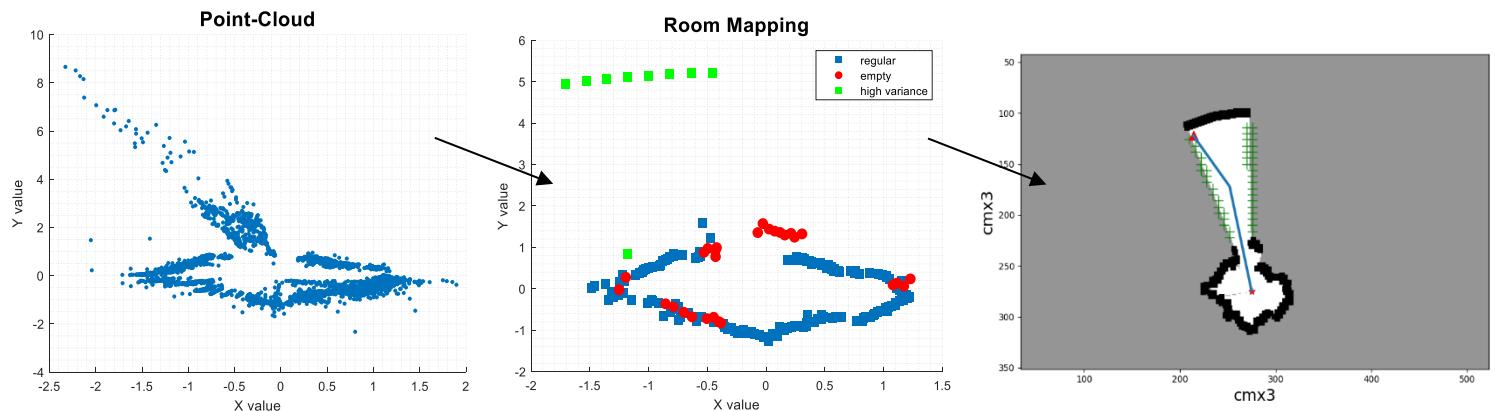
- [13] <https://youtu.be/bIkXOpA7snw>

נדים באיר הבא את המבנה עבורי הרחפן עבר מושני חדרים רצופים, שהוצגו בחלק ה'תוצאות':



#### דוגמאות נוספות לסריקות חדרים ומיציאת נתיב היציאה:

דוגמא 1:



דוגמא 2:

