אוניברסיטת חיפה

החוג למדעי המחשב

דו"ח מסכם מעבדה

קישור לגיט הפרויקט -

https://github.com/amitsheffi/2021lab

שם הקורס: למידת נתונים במערכות זמן אמת

שם המרצה: פרופסור דן פלדמן

שמות הסטודנטים: עמית שפי, איתי גרוס ודביר סבג

פרטי התקשרות:

עמית שפי - 058-412-0803 amitsheffi7@gmail.com

איתי גרוס - 052-220-1543 itaig1002@gmail.com

דביר סבג - 055-667-6733 dvirsabag3@gmail.com

ת"ז:

עמית שפי (323910588), איתי גרוס (212589543) ודביר סבג (214297962)

תאריך הגשה: 16/09/2021

רקע

מטרת הפרויקט שלנו הייתה לגרום לרחפן קטן לסרוק חדר, לאתר בו את הדלת ולצאת ממנה. על מנת לעמוד במשימה הזו, עלינו היה למצוא אלגוריתם לסריקת חדר, להציג מפה שלו, לנקות ממנה רעשים בעזרת אלגוריתם המוצא נקודות רעש ומוחק אותן, למצוא את הדלת בעזרת אלגוריתם המנתח את הנקודות שנותרו ומוציא נקודה המייצגת את היציאה מהחדר ואז לנוע לשם בעזרת אלגוריתם הזזה. במהלך הדו"ח נסביר איך ביצענו כל אחד מהשלבים, נבהיר למה ביצענו את השלבים דווקא בצורה שבחרנו בה ומה גרם לנו לבחור בה על פני אופציות אחרות ונציין קשיים שעלו במהלך כל אחד מהשלבים.

אלגוריתם סריקה

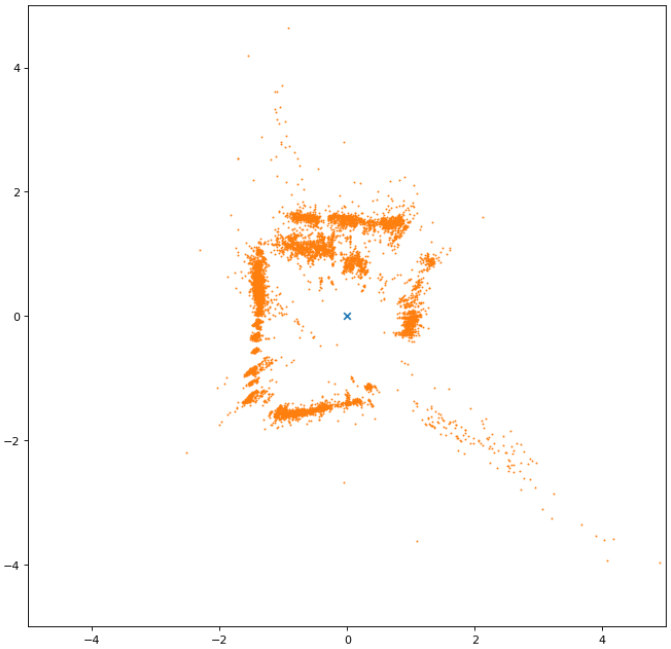
בשביל לסרוק את החדר היינו צריכים להסתובב עם הרחפן 360 מעלות, לקבל פריימים מהמצלמה ולהעביר אותם לאובייקט הSLAM על מנת לבנות מפה בסוף הסריקה. הבעיה הייתה שהרחפן איבד לוקליזציה. בשביל לטפל בבעיה זו במקום להסתובב בפקודה אחת 360 מעלות, הסתובבנו בחתיכות קטנות יותר עם הפסקה בין כל סיבוב. לאחר מספר בדיקות הגענו למסקנה שלהסתובב 24 מעלות כל פעם עובד לנו הכי טוב. הרחפן עדיין איבד לוקליזציה מדי פעם ולכן הוספנו שבין כל סיבוב הרחפן יעלה וירד וככה הרחפן יכול לזהות יותר אובייקטים בחדר.

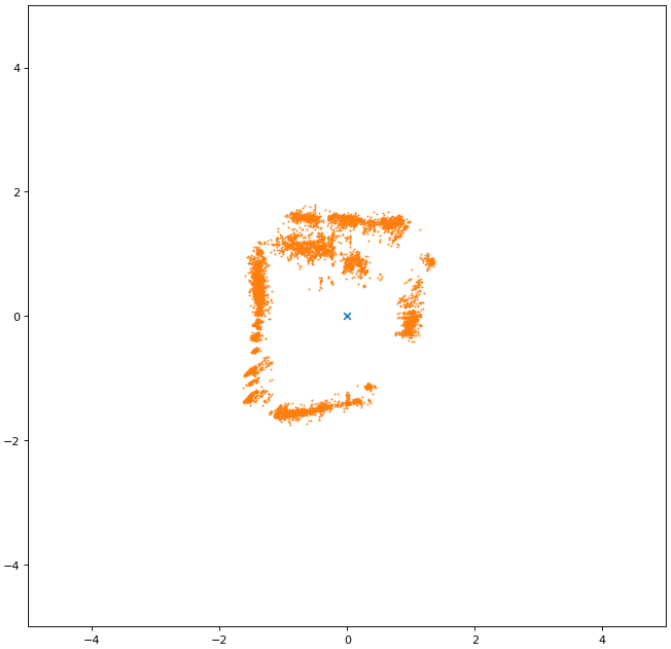
אלגוריתם לניקוי רעשים

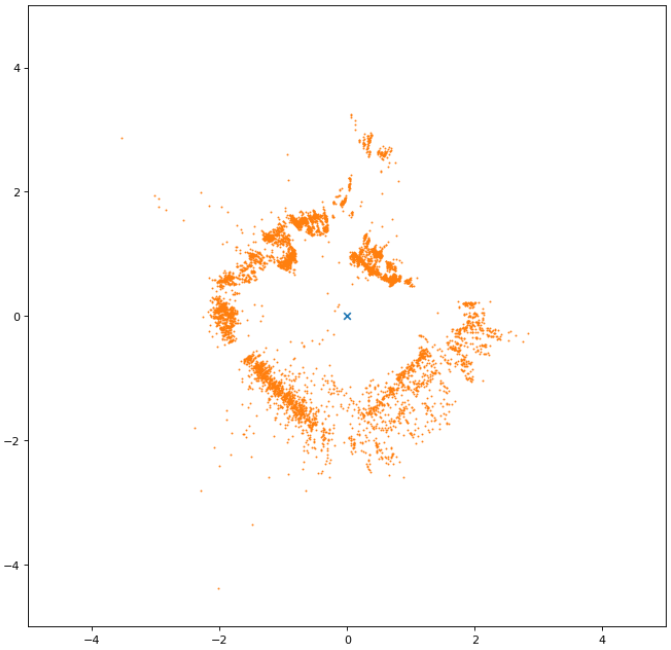
מכיוון שהמפה שהרחפן מחזיר אחרי שהוא סורק את החדר היא לא מפה מושלמת של החדר, וחלק מהנקודות הן לא נקודות אמיתיות אלא רעשים רצינו ליצור אלגוריתם שינקה את הרעשים וכך המפה תהיה אמיתית והאלגוריתם למציאת הדלת יעבוד הכי טוב. ראשית, שקלנו לקחת אלגוריתם ניקוי מוכן כמו DBSCAN או KD-tree אבל החלטנו שלא כי העדפנו למצוא רעיון מקורי משלנו. הרעיון הראשון שלנו היה לנקות את הרעשים לפי כמות השכנים שיש לכל נקודה. הגדרנו ששכן זה מי שבמרחק קטן שווה ל 0.1 ובשביל שנקודה לא תהיה רעש צריך להיות לה לפחות 10 שכנים. האלגוריתם הראשון שחשבנו עליו היה לעבור על כל הנקודות בוקטור הנקודות ועבור כל נקודה לעבור על כל הנקודות, לבדוק את המרחק אליהן ולסכום את כמות השכנים. הבעיה באלגוריתם זה היא שהאלגוריתם רץ בסיבוכיות זמן של $O(n^2)$ולכן אי אפשר להשתמש בו כי הוא לוקח יותר מדי זמן.
האלגוריתם השני והסופי שרשמנו היה על אותו עיקרון של למצוא כמות שכנים אך הוא הצליח לצמצם את זמן הריצה מ $O(n^2)$ לזמן ריצה של $O(n * log\, n)$ בהסתברות גבוהה.
בשביל להוריד את הסיבוכיות, קודם כל מיינו את וקטור הנקודות לפי מרחק מראשית הצירים, כאשר פונקציית המרחק שלנו מתייחסת למישור X,Z שהוא המישור המקביל לרצפה. לאחר שמיינו את וקטור הנקודות, עבור נקודה באינדקס i, קודם כל עברנו על כל האינדקסים הגדולים ממנה לפי הסדר ובדקנו האם הם שכנים של הנקודה. בשביל שלא נצטרך לעבור על כל האינדקסים ואז הסיבוכיות עדיין תהיה $O(n^2)$, לפני שנבדוק אם נקודה היא שכן, קודם כל נבדוק את ההפרש בין המרחקים של הנקודה באינדקס i לבין ראשית הצירים לבין הנקודה שהיא שכן פוטנציאלי לבין ראשית הצירים, אם ההפרש גדול מ 0.1 סימן שהנקודה הפוטנציאלית לא יכולה להיות שכן ושכל נקודה שהאינדקס שלה גדול מהאינדקס שלה לא יכולה להיות שכן. נעשה אותו דבר עבור האינדקסים שקטנים מ i. לכן בהסתברות גבוהה נצטרך לבדוק רק מספר קבוע של נקודות עבור כל נקודה ולכן סך כל הבדיקות ייקח $O(n)$.
אחרי שסיימנו לתכנת את האלגוריתם שלנו, הרצנו אותו על מספר מפות לדוגמא שקיבלנו מקבוצות אחרות בשביל לראות האם הוא עובד טוב והאם הוא מסנן את הרעשים כמו שצריך.
בשביל להוציא תמונה השתמשנו בקוד של הקבוצה של קארין ואסף מכיוון שהספרייה matplot לא קיימת בcpp ואנחנו לא ידענו איך לעבוד עם python.
לאחר הסתכלות על המפות ראינו שהאלגוריתם עובד והוספנו אותו לקוד.

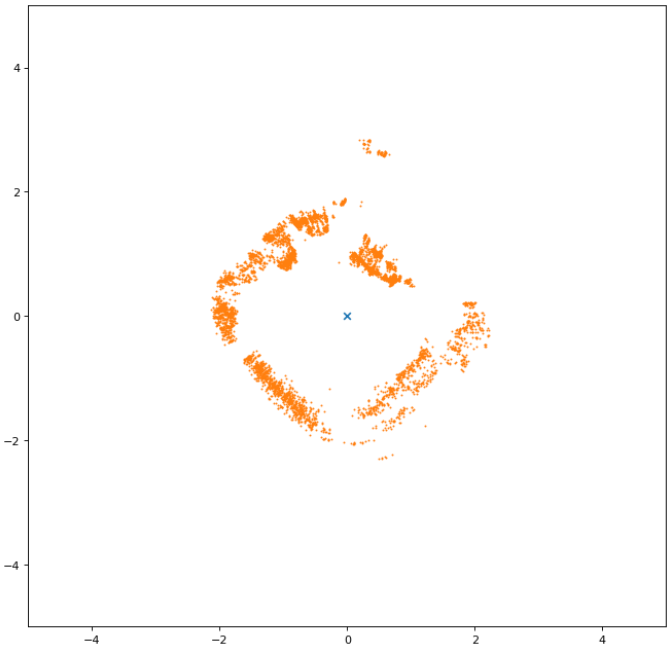
דוגמאות לניקוי הרעשים:

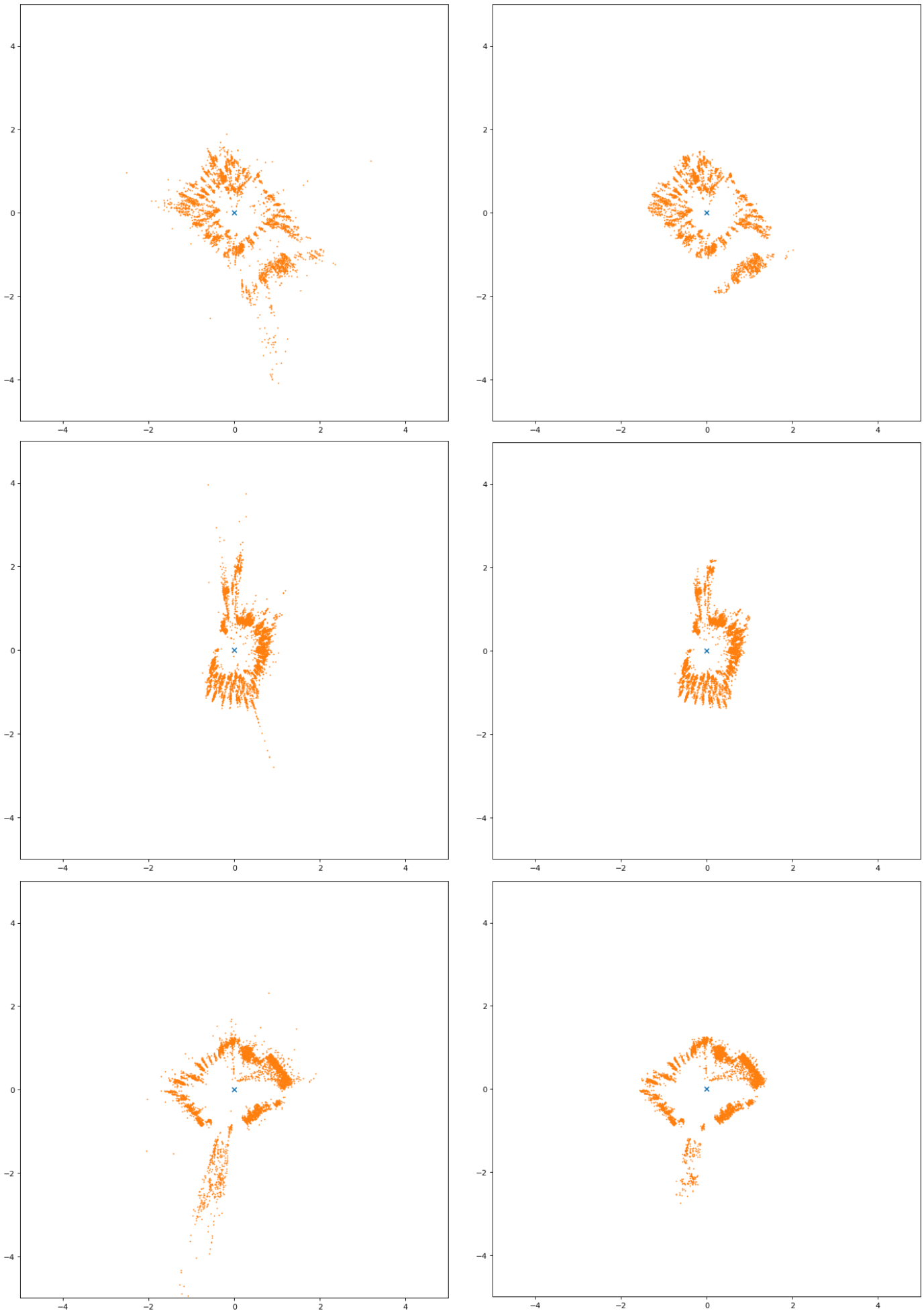
מצד ימין אפשר לראות את המפות לאחר ניקוי הרעשים ומצד שמאל את המפות לפני ניקוי הרעשים.










4
2
0
−2
−4
−4
−2
0
2
4

אלגוריתם למציאת הדלת

על מנת למצוא את הדלת חשבנו על מספר אלגוריתמים עד שהגענו לאלגוריתם הסופי. הראשון מביניהם היה הנאיבי ביותר - לנקות רעשים ולהחזיר את הנקודה הרחוקה ביותר. אך בגלל שניקוי הרעשים לא מנקה 100% מהרעשים ובגלל שישנם חדרים חריגים ומפות חריגות שבהן יש נקודות קצה החלטנו שאלגוריתם כזה לא יעבוד ביותר מדי מקרים.
לאחר מכן חשבנו לעשות ממוצע לכלל הנקודות ולהחזיר את הנקודה הכי קרובה לתוצאה מתוך הנחה שהדלת אמורה לבלוט במפת הנקודות אבל האלגוריתם הזה החזיר נקודות גבוליות לרוב. את האלגוריתם הזה שכללנו לאלגוריתם העושה ממוצע לכל נקודות המקסימום לפי המרחק מנקודת ה-0 (את מספר הנקודות החלטנו מראש), וכך גם אם הנקודה המקסימלית לא הייתה נקודת הדלת עדיין בממוצע רוב נקודות המקסימום היו ליד הדלת. אך אלגוריתם זה עדיין לא היה מספיק טוב.
לבסוף הגענו לשני אלגוריתמים שסיפקו תוצאות דומות - האחד היה חלוקה של המפה ל'סלייסים' לפי זווית ומציאת ה"סלייס" עם המרחק הממוצע הגדול ביותר של כל נקודה ב"סלייס" מראשית הצירים. ההיגיון הוא שהנקודות של הדלת יהיו באותו "סלייס" ולכן בגלל שהנקודות בדלת הכי רחוקות ה"סלייס" של הדלת יהיה המקסימלי.
השני היה אלגוריתם המוצא את האינטרוול בעל 10 הנקודות שסכום המרחקים בו מה-0 הוא הגדול ביותר, כולמר 10 הנקודות הרציפות בסריקת החדר כך שסכום מרחקיהן מנקודת ה-0 הוא המקסימלי. ההיגיון מאחורי אלגוריתם זה הוא שנקודות הדלת הן הנקודות הכי רחוקות ואם בסריקה נקודות יצאו ברצף זאת אומרת שהן קרובות אחת לשנייה במציאות.
למרות ששני האלגוריתמים סיפקו תוצאות טובות והסיבוכיות שלהם הייתה שווה, העדפנו ללכת על אלגוריתם האינטרוולים משום שחשבנו שהוא ייחודי יותר ומייחד אותנו. שמנו לב שהאלגוריתם השני היה נפוץ מאוד במספר וריאציות מעט שונות והעדפנו ללכת על אלגוריתם יותר מקורי.
הקושי המרכזי בשלב מציאת הדלת היה להתמודד עם מפות שיש בהן נקודות קצה, למשל מפה עם גוש נקודות מרוחק מהדלת. בגלל הדרישה של אינטרוול הצלחנו להתמודד בצורה טובה עם אותם מקרי קצה בגלל הדרישה לצפיפות (בנוסף לניקוי הרעשים).

כל אלגוריתם בדקנו על מפות לדוגמא והשווינו בעזרת קובץ אקסל:
בתמונות: מימין טבלת ההשוואה הראשונה שמשווה בין שני האלגוריתמים הראשונים ומשמאל טבלת ההשוואה השנייה אשר משווה בין שני האלגוריתמים השניים. כאשר האלגוריתמים רצו על המפות שקיבלנו מקבוצות אחרות.

map	algorithm	X	Z
pointData	interval	0.19	2.17
pointData	slices	-0.21	1.96
pointData0	interval	-0.61	-2.75
pointData0	slices	-0.61	-2.75
pointData1	interval	0.33	2.83
pointData1	slices	-2.12	0.29
pointData8	interval	-1.61	-1.4
pointData8	slices	-1.61	-1.4
pointDataVered	interval	2.01	-0.88
pointDataVered	slices	2.01	-0.88

	algorithem with X points	X	Z
pointData	sheffi 50	0.267701	-1.18559
pointData	sheffi 10	0.117132	2.16174
pointData	sabag 50	0.020493	1.76787
pointData0	sheffi 50	-1.429	0.193504
pointData0	sheffi 10	-0.527592	-3.50549
pointData0	sabag 50	-0.905015	-2.15516
pointData1	sheffi 50	1.984	0.361642
pointData1	sheffi 10	0.374514	2.91682
pointData1	sabag 50	0.562776	-0.3287
pointData8	sheffi 50	-1.07932	-1.48688
pointData8	sheffi 10	2.54509	-2.32751
pointData8	sabag 50	0.281325	-1.34949
pointDataVered	sheffi 50	-1.56311	0.304813
pointDataVered	sheffi 10	0.638853	-3.09102
pointDataVered	sabag 50	0.760976	-1.06909

אלגוריתם הזזה

אלגוריתם ההזזה שלנו השתמש בנקודה של הדלת שקיבלנו מאלגוריתם למציאת הדלת, מהנקודה הנוכחית של הרחפן שקיבלנו מפונקציית TrackMonocular באמצעות מטריצה ואת זווית הרחפן שגם אותה קיבלנו מפונקציית TrackMonocular. בשביל להוציא מהמטריצה ש TrackMonocular מחזירה השתמשנו בקוד מהאינטרנט שהוציא לנו את שיעור הנקודה הנוכחית ואת הזווית שלנו במישור המקביל לרצפה. לאחר מכן חישבנו את זווית הוקטור בין הנקודה הנוכחית לנקודת הדלת ביחס לציר ה X באמצעות שתי הנקודות. אחרי שהיה לנו את הזווית של הרחפן ואת הזווית בין הוקטור, חיסרנו בין שתי הזוויות וככה קיבלנו את הזווית שאנחנו צריכים להסתובב בה. הסתובבנו עם הרחפן כך שיסתכל על הדלת וטסנו קדימה לצאת מהדלת.

הסבר קוד

את כל הקוד העלנו לגיט בקישור - https://github.com/amitsheffi/2021lab

כל הקוד הוא קוד מקורי שאנחנו כתבנו למעט הקוד שהגיע עם התקנת ORB_SLAM2 ו ctello, הקובץ tello.yaml שלקחנו מהקבוצה של אלעד הירשל, אראל אפוטה ודוד דונר, והחלק של הוצאת הנקודה הנוכחית וזווית הרחפן הנוכחית מתוך מטריצת סיבוב, את חלק זה לקחנו מהאינטרנט, הוא נמצא בתוך הקוד של mono_tum.cc ורשום בהערות שלקחנו אותו מהאינטרנט.

הקבצים שאנחנו שינינו הם:

קובץ ה CMakeLists.txt כך שיתקמפל עם הtello ועם הפונקציות שלנו

קובץ FindingDoorAlgorithem.cpp אשר נמצא בתיקייה הראשית של ה ORB_SLAM2, בקובץ נמצאות כל הפונקציות שבהן השתמשנו בשביל אלגוריתם מציאת הדלת, כולל הפונקציה לניקוי רעשים. כל הקוד עם הערות מפורטות.

את הקובץ point.h אשר נמצא בתוך תיקיית include. קובץ זה הוא קובץ שיצרנו לגמרי לבד, יצרנו אובייקט חדש מסוג point בשביל שנוכל לעבוד בנוחות עם המפות. בקוד הקובץ הערות מפורטות.

את הקובץ point.cpp שנמצא בתיקייה src. קובץ זה הוא הקובץ מימוש של האובייקט point. בקוד הקובץ הערות מפורטות.

הקובץ mono_tum.cc אשר נמצא בתיקייה Monocular בתוך תיקיית Examples. זהו קובץ ההרצה הראשי, בתוך הקובץ במצא הthreads שסורקים את החדר, שמזיזים את הרחפן ושיוצאים מהחדר. בנוסף הקובץ יש את כל הפונקציות למציאת החדר שאנחנו משתמשים בהן. בקוד הקובץ הערות מפורטות.

הקובץ tello.yaml שנמצא בתיקייה Monocular בתוך תיקיית Examples. זהו קובץ הקליברציה של הרחפן, את הקובץ קיבלנו מהקבוצה של אלעד הירשל, אראל אפוטה ודוד דונר.

סרטון

את סרטון ההרצה ניתן לראות בקישורים הבאים, בקישור הראשון את צילום הרחפן ובשני את צילום המחשב -

סרטון הרצה רחפן

סרטון הרצה מחשב

עדכונים שבועיים

15/07/2021 - למדנו את הgit בשביל להבין איך משתמשים בו, פתחנו את הrepository הבא - https://github.com/amitsheffi/2021lab. לא הבנתי אם אתם רוצים שנהפוך גם אתכם לcollaborator או רק שתצפו בgit.
בנוסף הrepository כרגע ריק לגמרי ולא אתחלנו אותו כי היה לנו היום מועד ג' שלמדנו אליו, נסיים את האתחול בסופ"ש.
התקנו את המכונה הוירטואלית עד השלב של ההתקנת סיפריות כי לא הספקנו, נסיים את ההתקנה בסופ"ש.

משימות לשבוע הבא - לסיים את ההתקנה של הספרייה ולוודא שכל חברי הקבוצה מבינים איך לעבוד נכון בgit ולסיים את האתחול של הrepository. בנוסף לעשות את המשימות שיפורסמו בהמשך.

22/07/2021 -
הוספנו קובץ readme לגיט אבל עוד לא הספקנו ממש להתעסק עם זה כי עדיין יש לנו הרצאות להשלים כי את השבוע הראשון התחלנו בדיליי כי היה לנו מועד ג', אנחנו מקווים לסיים את זה בסופ"ש.
במכונה הוירטואלית התקנו הכל עד השלב של ההתקנת סיפריות python כי יש לנו בעיה בהתקנה של הספריות OS, ננסה להסתכל באינטרנט ולהבין מה הבעיה כרגע לא מצאנו פיתרון.

משימות לשבוע הבא - למצוא פיתרון לבעיית התקנת הספריות, סיום אתחול הrepository

30/07/2021 -
לא הספקנו להתעסק בגיט כי עדיין יש לנו בעיות עם המכונה הוירטואלית. אנחנו בקשר עם בר.

משימות לשבוע הבא - לסיים את האתחול של הrepository ולהתקדם עם המכונה הוירטואלית

05/08/2021 -
סוף סוף הצלחנו להתקין את המכונה הוירטואלית.
משימות לשבוע הבא:
לסיים את אתחול תיקיית הגיט, ולהתחיל לעבוד עם המכונה הוירטואלית עם הטלפון ובתקווה לעבור לרחפן עוד השבוע

12/08/2021 -
סיימנו לאתחל את התיקיית גיט. הרצנו את orb_slam על הסרטון מהאינטרנט וניסינו להתחיל למפות.

משימות לשבוע הבא
להמשיך למפות את הסרטון בתקווה להגיע לקראת חמישי לאוניברסיטה אחרי שהצלחנו לצפות על יבש

19/08/2021 -
התחלנו להתעסק עם הopencv והsavemap. לא כל כך הספקנו להתעסק לעומק בגלל המבחנים אבל ננסה להתעסק בזה בסופש.
משימות לשבוע הבא
להתכונן למבחן במבני נתונים מתקדמים ובמקביל לנסות להתעסק עם הopencv והsavemap

27/08/2021 -
עכשיו שמתי לב שזה לא שלח לי את הדיווח אתמול כי לא היה קליטה אז זה שמר את זה כטיוטא.
השבוע לא הספקנו המון כי למדנו למבחן.
משימות לשבוע הבא - לעבוד בשישי שבת בבית הרבה ולהגיע למעבדה בראשון מוכנים

02/09/2021 -
הגענו השבוע למעבדה בשלישי רביעי חמישי והצלחנו להתקדם הרבה. התקנו את המערכת גם על הלפטופים שלנו והתחלנו לעבוד עם הרחפן. בשלישי ורביעי היה לנו בעיות להעביר תמונות מהרחפן לSLAM אבל לפני שהלכנו בחמישי הצלחנו לסדר את זה ועכשיו אנחנו יודעים איך לשלוח פקודות לרחפן, איך לקבל ממנו תמונות ואיך להעביר אותם

לSLAM בהצלחה. בנוסף קיבלנו מקבוצה שיש לה מפה של חדר את המפה למייל בשביל שנוכל במקביל גם לעבוד על האלגוריתם למציאת הדלת.

משימות לשבוע הבא - ללמוד בסופש למבחן בלינארית, לעבור את המבחן בראשון בהצלחה. ביום שני עד רביעי לנסות לעבוד בבית על אלגוריתם למציאת הדלת עם המפה שיש לנו בתקווה להצליח להגיע לאלגוריתם שמצליח למצוא עד חמישי בשביל שנצליח לבדוק אותו בלייב, גם אם האלגוריתם לא הכי יעיל וטוב שאפשר. בחמישי להגיע למעבדה להמשך עבודה עם הרחפן ולהצליח לגרום לו לסרוק את החדר ולשמור את המפה בסוף הריצה, ואם יהיה לנו את האלגוריתם מוכן אז גם להצליח לנתח את המפה עם האלגוריתם ואם לא אז נתפצל כך שחלק יעבדו בחמישי על הרחפן וחלק על האלגוריתם.

- 09/09/2021
הצלחנו לעבוד קצת במהלך החגים, הצלחנו לגרום לרחפן לשמור מפה והתחלנו במקביל לעבוד על האלגוריתם למציאת הדלת.
משימות לשבוע הבא - לסיים את הפרויקט עד תאריך ההגשה. בסופש לסיים עם האלגוריתם ולהגיע בראשון בבוקר שהרוב כבר מוכן

סיכום

לסיכום, אנחנו חושבים ששמנו דגש על ייחודיות, יצירתיות ומקוריות,במהלך ניקוי הרעשים למשל, שהעדפנו לחשוב על אחד משלנו במקום לקחת מימוש מוכן של אלגוריתם אחר מהאינטרנט, ויצא ניקוי שעובד מעולה, או במהלך מציאת הדלת שהעדפנו לחשוב על אלגוריתם ייחודי ולא הלכנו לפי העיקרון שראינו שהרבה קבוצות בחרו בו. אנחנו חושבים שעמדנו בדרישות בצורה טובה תוך כדי עבודה רצינית ועניינית למרות כל הקשיים. השתמשנו באופן מושכל בפונקציות מוכנות, חשבנו בעצמנו על האלגוריתמים, הקפדנו לתעד ניסיונות ואת העבודה בכלל ובסופו של דבר - עמדנו ביעד. לכן אנחנו חושבים שהפרויקט והדרך להשלמתו היו נכונים וטובים.

קרדיטים ותודות

עירד נוריאל - על כל העזרה כשעלו קשיים בכתיבת הקוד, התמיכה והנכונות לעזור.
צמד המתרגלים בר ושגיא - על העזרה בבעיות ההתקנה ועל העזרה בפתרון בעיות קוד שצצו מדי פעם.
הקבוצה של אלירון רחימי ושחף גיל דגן - על העזרות בהתקנה והעברת קבצים שהזדקקנו להם.
הקבוצה של קארין ספרי ואסף סולומיאק - על העזרה בהוצאת המפות.
הקבוצה של אלעד הירשל, אראל אפוטה ודוד דונר - על קובץ המפות לדוגמא וקובץ הקליברציה של הרחפן שהם העבירו לנו.