דף סיכום הפרויקט

הפעלת המערכת: בכדי להריץ את המערכת יש צורך בכמה קבצים לפני כן במיקומים מסויימים:

1. mono_tum.cc שיהיה בתיקייה ORB_SLAM2/Monocular/examples
2. tello.yaml שאמור להימצא באותה תיקייה כמו לעיל.
3. algo.py אשר לא משנה איפה הוא נמצא

בכדי להריץ את המערכת יש להריץ שני קבצי קוד ביחד, יש להריץ את ה ORB_SLAM על ידי הפקודה הבאה(כמובן שכאשר משנים את ה monotum.cc יש בהתחלה לבצע build.sh מחדש):

```
./Examples/Monocular/mono_tum Vocabulary/ORBvoc.txt
Examples/Monocular/tello.yaml
```

לאחר מכן יש להריץ את הקוד algo.py בעזרת idle של python מיד לאחר הרצת פקודת ORB_SLAM(התקשורת בין שני הקודים monotum,algo מתבצעת על ידי קבצים אשר נוצרים בתיקיית tmp, ולכן לאחר כל הרצה יש צורך למחוק את שני הקבצים pointdata.csv,Result.csv מתיקייה זו, ואז לבצע שוב את תהליך ההרצה).

אתגרים ופתרונות:

1. האתגר הראשון שנתקלנו בפניו היה התקנת תוכנת הOrb-slam במהלך ההתקנה נתקלנו בשגיאות רבות פתרון הבעיה נעשה על ידי חיפוש אינטנסיבי באינטרנט וניסוי של פתרונות רבים לכל אחת מהבעיות.

2. התקנת ספריית ctello כל שלב בהתקנה כלל errors ושגיאות שונות שנאלצנו להתמודד איתן על ידי בדיקת מקורות שונים ופורומים שונים באינטרנט והתייעצות קלה עם חברים לקורס.

3. חיבור בין מצלמת הרחפן לתוכנת Orb-Slam. לא הצלחנו לקבל תמונות לעיבוד בOrb-slam ניסינו פתרונות שונים ביניהם התקנה על מגוון מחשבים וניסיון להתקנה על מחשב ללא מכונה וירטואלית, חלק מהפתרונות גרמו לפגיעה במערכת ההפעלה הווירטואלית מה שהוביל להתקנה מחדש של מערכת ההפעלה. לבסוף הצלחנו לבצע את החיבור על ידי מספר שלבים ראשית עברנו למחשב אחר ושנית שינינו את הגדרות המכונה הווירטואלית.

4. בניית אלגוריתם למציאת יציאה מהחדר. הניסיון הראשון שלנו היה למצוא את גבולות החדר בעזרת התמרת האף. עשינו זאת ע"י כך שיצרנו תמונה של פיזור הנקודות בעזרת matplotlib , לקחנו תמונה זו וטשטשנו אותה בשיטת medianblur בעזרת opencv להסרת רעש ומציאת הקלאסטרים ואז מצאנו את הקצוות של הקלאסטרים בעזרת canny edge detection ב opencv. על התוצאה הרצנו את

התמרת האף כפי שהיא ממומשת ב opencv. חשנו שעל מנת לקבל גבולות חדר מכל הקווים הנוצרים כדאי לקחת את נקודות החיתוך שלהם ולמצוא bounding box מתאים. אך נתקלנו ב2 בעיות:

א. לא ידענו כיצד להשיג את bounding box זה.

ב. על מנת שתוצאת האלגוריתם תהיה שימושית במציאת נקודת היציאה על הנקודות בה להיות ביחידות של map points בorbslam (ביחידות שקיבלנו לפני שהמרנו לתמונה) אך לא מצאנו דרך להמיר את נקודות החיתוך הנתונות בפיקסלים לנקודות בעולם.

עקב בעיות אלו החלטנו לנסות אלגוריתם אחר. באלגוריתם זה אנו מחלקים את הנקודות ממיפוי החדר לslices של זוויות, גודל הslice ניתן לבחירה ובחרנו אותו להיות מעלה אחת, ואז הסתכלנו על כל 10 מעלות ולקחנו את החציון של כל sub-slice של 10 מעלות אלו, לאחר מכן התקדמות במעלה אחת לעשר המעלות הבאות(9 מעלות נשארות אותו הדבר, לדוגמה במידה והסתכלנו על החציון של המעלות 1...10, אז לאחר מכן אנו מסתכלים על החציון של 2...11 ושומרים את החציונים שלהם), החציון הרחוק ביותר שמתקבל הינו נקודת היציאה של הרחפן מן החדר.

בעזרת אלגוריתם זה הצלחנו לזהות נכונה את היציאות בחדרים שונים שניתנו לנו כקבצי csv. עקב הצלחת האלגוריתם בחרנו בו לשילוב במערכת הסופית.

5. אלגוריתם יציאה מהחדר. הפתרון הראשון שניסינו היה לכוון את הרחפן ישירות לכיוון היציאה והצלחנו כך לצאת במספר מקרים מהחדר אך בחרנו לנסות לשפר את אחוזי ההצלחה של האלגוריתם כי זיהינו בעיה עיקרית אחת באלגוריתם הראשון: הרחפן אינו מדויק בתנועותיו ולכן משנה מיקום באופן שלא בשליטתנו במהלך מיפוי החדר, כך נוצר מצב שהנקודה והכיוון בו מסיים הרחפן את המיפוי שונה מהנקודה והכיוון בו התחיל אותו. בכדי לפתור זאת, הוספנו פונקציות חישוב זווית מהיציאה ממיקומו הנוכחי, ואז סיבוב ולאחר מכן התקדמות לאט לאט תוך כדי בדיקה אם יש סטייה חדשה מכיוון היציאה אשר תגרור לסיבוב נוסף בהתאם לכך. חוץ מזה בכדי לשפר את היציאה מהחדר בצורה טובה יותר היינו צריכים לשפר את המיפוי של החדר (היו פעמים אשר ORBSLAM איבד את הtrack שלו), ולכן הוספנו קטע קוד אשר מבצע בדיקה האם ORBSLAM מצליח לנתח את התמונות, במידה ולא אז הוא חוזר על הפעולה האחרונה שהוא ביצע שוב ושוב עד שהORBSLAM מתאפס ועובד מחדש.

לאור כל האתגרים לעיל שהצלחנו להתמודד איתם באופן מיטבי ועצמאי תוך השקעת זמן ואנרגיה רבים ובגלל שהתוצאה הסופית שקיבלנו עובדת על מגוון חדרים ומצבים שונים, אנו חושבים שמגיע לנו ציון גבוה. יתר על כן, עזרנו לקבוצות אחרות על ידי הסברים וקטעי קוד.