The Iby and Aladar Fleischman
Faculty of Engineering
Tel Aviv University

הפקולטה להנדסה
ע"ש איבי ואלדר פליישמן
אוניברסיטת תל אביב

ניווט אוויריי באמצעות רחפן אוטונומי

פרויקט מס' 19-1-1-1758

דו"ח סיכום

מבצעים:

אופיר מירן 206564189
רוני כדורי 208580274

מנחה:

יונתן מנדל אוניברסיטת ת"א

מקום ביצוע הפרויקט:

אוניברסיטת תל אביב בניין וולפסון – מעבדת רחפנים אוטונומיים

תוכן עניינים

רשימת איורים

רשימת טבלאות

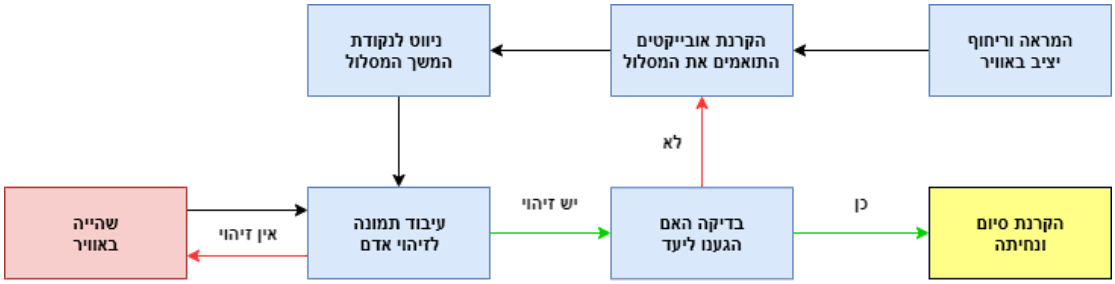
תקציר

תחום העבודה המרכזי של הפרויקט הינו תחום הרחפנים בכלל ורחפנים אוטונומיים בפרט. הפרויקט עוסק במימוש רחפן אוטונומי שבאפשרותו להדריך ולנווט אדם חיצוני (או קבוצה של אנשים), במטרה לאפשר לו להגיע לייעדו. הרחפן יקבל כקלט מסלול טיסה מבין מספר סופי של מסלולים אפשריים ויממש את תהליך הניווט. תהליך זה יכול להתבצע הן בתנאי חוץ והן בתנאי פנים ללא שימוש בטכנולוגיית GPS. מכאן שהנושאים המרכזיים בהם עוסק הפרויקט הינם ניווט, אלגוריתמיקה, בקרה ועיבוד תמונה.

מטרת הפרויקט הינה פיתוח תחום הרחפניים האוטונומיים וניווט ללא GPS. כיום השילוב של מקרנים ורחפנים יחד אינו נפוץ בעולם. למעשה, אף אזכור בפרויקט העוסק בנושא זה לא נמצא על ידנו ומכאן הרצון שלנו לפתח תחום חדשני זה.

מהות ומטרת הפרויקט הכתיבו דרישות שונות לגבי אופן מימוש הפרויקט. כך למשל, בשל הדרישה לפעולה בתוך מבנה, גודל הרחפן צריך להיות מותאם גם לתעופה בחדר. בנוסף, על הרחפן עצמו מורכבים אלמנטים שונים: מצלמה אחת שתפקידה לספק קלט לאלגוריתם עיבוד התמונה של פעולות האדם; מצלמה נוספת המספקת לאלגוריתם הבקרה את קורדינטות המיקום של הרחפן במרחב; מקרן, להקרנת מידע והדרכה על גבי הרצפה או הקיר.

באיור הבא מוצגת דיאגרמת בלוקים כללית המתארת את פרויקט זה.



איור 1. דיאגרמת בלוקים כוללת של הפרויקט.

נקודת המוצא של המערכת היא ב-"המראה בטוחה וריחוף יציב באוויר". מיד לאחר מכן הרחפן מתחיל במסלול הניווט ובמקביל מקרין חצים ירוקים על גבי הרצפה בכיוון ההתקדמות (החצים מופיעים בנספחים). בהגיעו לנקודת ביניים במסלול הרחפן מתייצב באוויר ללא מהירות ומתחיל בשלב זיהוי האדם. ראשית, המקרן שעל גבי הרחפן מפסיק את ההקרנה הנוכחית ומקרין סימן עצור על גבי הרצפה (סימן זה מופיע בנספחים). לאחר מכן מופעל האלגוריתם לזיהוי אדם שבמהלכו הרחפן יוכל להחליט האם המשתמש הגיע לנקודת הביניים ולכן ניתן להמשיך בניווט. כל עוד לא בוצע זיהוי, הרחפן ימשיך בריחוף יציב באותה נקודת עצירה. אחרת, סימן העצור יוחלף בהקרנת חץ ירוק והרחפן ימשיך אל נקודת העצירה הבאה. עם ההגעה לנקודת סיום המסלול, יוקרן על גבי הקרקע סימן המעיד על הגעה ליעד ויחל תהליך הנחיתה (סימן זה מופיע בנספחים).

1 הקדמה

מטרות הפרויקט:

בשנים האחרונות עולם כלי הטיס הבלתי מאוישים ובפרט הרחפנים בא לידי ביטוי במגוון תחומים בחיי היום-יום. בין היתר ניתן למצוא את הרחפנים בשירותי הביטחון, שירותי חיפוש והצלה, חקלאות, צילום ואף בתחום משלוחי המזון והדואר. בפרויקט זה נתמקד בתחום אחד עם מטרה מרכזית שהיא הדרכת אדם אל נקודת יעד מוגדרת מראש **וללא שימוש בטכנולוגית GPS**.

מטרות הפרויקט מסודרות בחשיבותן באופן הבא:

א. **יציבות הרחפן באוויר -** המראה של הרחפן, התייצבותו במשך פרק זמן לא מבוטל ונחיתה מבוקרת. נדרוש שתעופת הרחפן באוויר תהיה יציבה גם בסביבה רועשת המושפעת מרוחות ומגורמים חיצוניים אחרים.

ב. **יכולות ניווט -** ניווט בין מספר סופי של נקודות שנבחרו מראש ותעופת הרחפן בצורה יציבה ורציפה. כמו כן, נרצה להעניק למשתמש מגוון רחב של אפשרויות בחירה של מסלולים ולהבטיח שזמן ביצוע כל מסלול יהיה מותאם לקצב הליכה אופייני לאדם.

ג. **ניווט בשילוב הקרנה** – מעבר על מסלול מוגדר מראש תוך הקרנת סימנים מתאימים למסלול על גבי הקרקע. סימנים אלו עשויים להיות חצים, מפה, תצלום/תמונה ועוד. הרחפן יקרין את הפרטים המתאימים בהתאם למסלול והיעד.

ד. **עיבוד תמונה** – עם עצירת הרחפן בנקודת ביניים לאורך המסלול והתייצבותו באוויר, יופעל אלגוריתם לעיבוד תמונה לצורך זיהוי הגעתו של אדם לנקודה זו. לאחר זיהוי מוצלח של האדם הרחפן ימשיך במסלולו לנקודות הביניים הבאה.

ה. **הימנעות ממכשולים** – זיהוי מכשול במסלול על ידי הרחפן, ולאחר מכן התייצבות והמתנה באוויר לפרק זמן קבוע. ובמידה והמכשול לא יוסר הרחפן יבצע נחיתה.

מוטיבציה:

המוטיבציה העיקרית לפרויקט היא פיתוח תחום הרחפנים בכלל ושילובם בתחום הניווט בפרט. בכך נשאף לתרום לתחום התחבורה והביטחון. מוטיבציה נוספת המהווה חידוש לתחום הרחפנים היא השילוב של מקרן בפרויקט על גבי הרחפן.

הגישה לפתרון הבעיה:

כיום תעופת הרחפנים האוטונומיים מתבצעת בעיקר בעזרת טכנולוגיית ה- GPS אשר מספקת לרחפן את קורדינטות המיקום המדויק שלו במרחב. אולם, כיום קיימת בעיה והיא שבמקומות ללא GPS טכנולוגיה זו אינה יעילה ולכן יש צורך בחלופות שונות. בפרויקט שלנו פתרנו בעיה זו על ידי שימוש בטכניקת SLAM. כפי שיפורט בפרק הרקע תיאורטי, טכנולוגיה זו עושה שימוש בשתי מצלמות או יותר על מנת לחלץ את המיקום המרחבי של הרחפן ולמפות את סביבתו.

פרויקטים הקיימים בנושא:

המערכת הכוללת בפרויקט זה הינה חדשנית בתחומה. ההשראה לשילוב טכנולוגיית הרחפנים האוטונומיים עם מקרנים נלקחה מסרטון ההדמיה המצורף[1]. נבחן כעת את שלושת התחומים המרכזיים של הפרויקט.

עיבוד תמונה: בפרויקט שלנו נעשה שימוש באלגוריתם לזיהוי אובייקטים המתבסס על זיהוי צבעים במרחב. את המטרה הכוללת של זיהוי אדם ניתן לממש בעזרת אלגוריתמים אחרים המתבססים על למידה עמוקה. אלגוריתמים אלה נעזרים בסביבות תוכנה שונות כדוגמת caffe אשר מצריכות תוספת מיוחדת של זיכרון על גבי סביבת הפיתוח של הרחפן. כמו כן, ישנם אלגוריתמים שונים שלא מבוססים על למידת מכונה אך מבוססים על זיהוי ספציפי של בני אדם או אובייקטים מוגדרים מראש כדוגמת שולחן או ציפור. זוויות הצילום שונות שעלולות

[1] https://www.youtube.com/watch?v=cqU_hR2_ILU

להיווצר באופן טבעי במהלך תעופת הרחפן משפיעות לרעה על יעילות הזיהוי של האלגוריתמים האלה. לכן, בשל מגבלות מערכת ובעיות זיכרון בחרנו להשתמש באלגוריתם המבוסס על זיהוי צבעים כפי שיפורט בהמשך.

בקרת הרחפן: לשם כתיבת אלגוריתם הבקרה ניתן לנו אלגוריתם ראשוני על ידי המנחה למימוש תעופת הרחפן בעזרת טכנולוגית ה SLAM. המטרה שלנו הייתה להתאים את הקוד לרחפן הספציפי בו נעשה שימוש בפרויקט, להתאים את הפרמטרים הדרושים וכן לייעל את האלגוריתם. בעקבות כך, לא היה צורך להיעזר באלגוריתמים חיצוניים אחרים.

חלק מרכזי נוסף בפרויקט הוא **הקרנת אובייקטים** אך לצורך כך לא נעזרנו באלגוריתמים חיצוניים.

2 רקע תיאורטי

בפרק זה נתאר את הידע התאורטי עליו מתבססים האלגוריתמים בהם נעשה שימוש בפרויקט. בנוסף, נתאר את הפרמטרים והמאפיינים השונים על פיהם נבחרו המצלמות והמקרן הנמצאים על גבי הרחפן.

Proportional-derivative controller:

האלגוריתם הראשון בו נתמקד הוא אלגוריתם הבקרה. לצורך הטסת הרחפן לאורך מסלול הניווט המוגדר, על אלגוריתם הבקרה לחשב את מהירויות התעופה הרלוונטיות למסלול. לשם כך נעזרנו בפיתוח אשר נלמד בקורס "מבוא לתורת הבקרה" לצורך מציאת מהירות המוצא של בקר על ידי חישוב שגיאת מצב יציב. הנושא שעליו התבססנו הוא Proportional-derivative controller. במערכת זו פונקצית התמסורת של המערכת מתוארת באופן הבא:

$$(1), \qquad C(s) = K_p + K_d s$$

כאשר K_p ו K_d מסמלים את פרמטר הפרופורציה ופרמטר הנגזרת בהתאמה. במרחב הזמן, אם נסמן ב e את השגיאה של הסיגנל וב- v את מוצא הבקר אז מתקיים הקשר האיטרטיבי הבא:

$$(2), \qquad v = K_p e + K_d \dot{e}$$

במקרה שלנו v מתאר את המהירות הלינארית של הרחפן בכל אחד מהצירים x ו- y ואילו e מתאר את ההפרש בין המיקום הנוכחי של הרחפן במרחב לבין המיקום של נקודת העצירה הבאה במסלול. מכיוון שאנו עוסקים במערכת קורדינטות וקטורית, נוסחה (2) מתייחסת הן לציר x והן לציר y (הצירים הרוחביים). עבור הציר האנכי, ציר z, המהירות לאחר ההמראה לא משתנה וזאת על מנת לשמור על יציבות הרחפן באוויר. יחד עם זאת, נציין כי האלגוריתם בו נעשה שימוש הוא איטרטיבי ועל כן נגזרת השגיאה $\dot{e}$ מחושבת גם היא בצורה איטרטיבית.

הבחירה לחישוב מהירות הרחפן על פי בקר Proportional-derivative, נבעה מכמה סיבות. ראשית, זהו אלגוריתם נפוץ אשר מציג ביצועים טובים הן עבור השהיה לא גדולה מדי והן מבחינת התכנסות טובה לערך האמיתי. כמו כן, גם מבחינת ה overshoot, אלגוריתם זה בעל ביצועים טובים ללא צריכה מופרזת של הספק המנוע. תכונה זו חשובה במיוחד בפרויקט שלנו שכן נעדיף שהרחפן יתייצב באוויר ולא יבצע תנודות חדות סביב נקודת היעד.

Simultaneous Localization and Mapping

נושא נוסף בו נרצה להתמקד אשר בא לידי ביטוי באלגוריתם הבקרה הוא הבעיה החישובית של Simultaneous Localization and Mapping, או בקיצור SLAM. SLAM הוא טכניקה שבעזרתה רובוטים יכולים להתמצא במרחב לא מוכר בו הם נמצאים, ברמת דיוק גבוהה וללא שימוש בטכנולוגיית GPS. SLAM מאפשר לרובוטים לגלות את מיקומם ביחס לסביבה שלהם וכן למפות את סביבתם בזמן אמת, כלומר, במקביל לתזוזת הרובוטים. בנוסף לרמת הדיוק הגבוהה המתקבלת, הטכניקה של SLAM מאפשרת לרובוטים לנווט גם בסביבה הכוללת רעשי רקע רבים.

כיום SLAM מתייחס לשיטות חישוב שונות ומגוונות. אחת מהשיטות המקובלות לגילוי מיקום הרחפן בסביבה ומיפויה, מתבצע על ידי שימוש בקומבינציה של מצלמות וחיישנים והטלת קרני לייזר על מסלול התנועה של הרובוטים. כתוצאה מכך ניתן לחשב את המיקום של נקודות הפגיעה של הקרניים ביחס למיקום המצלמה במרחב תלת ממד. בשיטה זו המצלמות בהן נעשה שימוש דוגמות את הסביבה בקצב גבוה על ידי לקיחת תמונות מזוויות שונות. בעזרת מדידות אלו ניתן לשערך את המיקום היחסי של הרובוט. מדידות אלו מתוארות בשם - depth-image. כאשר הרובוט זז, נקודת האינפורמציה שממנה דוגמות המצלמות משתנה גם כן, כתוצאה מכך ניתן לאמוד כמה התרחק הרובוט מנקודת האינפורמציה הקודמת וכן לשערך את מיקום הרובוט על המפה.

לסיכום, החשיבות הרבה בשימוש בטכניקת SLAM היא בכך ש-SLAM מאפשר לרחפן להתמצא במרחב ברמת דיוק גבוהה יותר מאשר זו הניתנת על ידי הבקרה המובנית המגיעה עם הרחפן. כתוצאה מכך ניתן למנוע התנגשויות עם עצמים שונים לאורך מסלול התעופה ובכך לייעל אופן טיסת הרחפן.

אופן הטסת הרחפן

הטסת הרחפן ניתנת לביצוע בעזרת שני מגנונים. הראשון הינו בעזרת שלט וירטואלי הנקרא "FreeFlight Pro" אשר שולט על מהירות וכיוון הטיסה. בעזרת השלט ניתן לראות את שדה הראייה של הרחפן על ידי המצלמה המובנית המגיעה עם הרובוט. בנוסף, מוצגים נתוני טיסה חשובים כגון גובה, מהירות ומצב סוללה. יחד עם זאת, כדי לתפעל את הרחפן במצב אוטונומי נעזרנו במנגנון השני. על פי מנגנון זה נתוני הטיסה מן הרחפן נקלטים בעזרת חלונית terminal שבמערכת ההפעלה linux. על מנת לאפשר זאת, נעזרנו בספרייה מובנית של הרחפן, הנקראת bebop_autonomy המכילה פקודות שונות הכתובות בשפת ROS. בין פקודות אלו ישנן פקודות המראה, נחיתה, וכן פקודות המתייחסות למהירות הטיסה והניווט. התייחסות לשפת ROS מתוארת בפרק תיאור תוכנה.

אלגוריתם עיבוד התמונה

אלגוריתם נוסף שבו נרצה להתמקד בחלק זה הוא אלגוריתם עיבוד התמונה. כפי שנאמר בפרק הקודם, בפרויקט זה בחרנו להשתמש בזיהוי אובייקטים המבוסס על מרחבי הצבע (color space).

מרחב צבע הוא שיטה על פיה ניתן למיין ולתאר צבעים שונים. בדרך כלל צבעים מתוארים בעזרת שלושה מאפיינים או קורדינטות, המתארות את המיקום שלהם במרחב הצבע. ערכי הקורדינטות המתארות צבע במרחב צבע מסוים יקבלו ערכים שונים במרחב צבע אחר.

בין מרחבי הצבע הנפוצים ישנו מרחב הצבע RGB ומרחב הצבע HSV. בפרויקט שלנו, הקלט המגיע לאלגוריתם עיבוד התמונה מתואר על ידי שלוש הקורדינטות של מרחב ה RGB. במרחב זה כל צבע יתואר על ידי הקומבינציה של צבעי אדום, כחול וירוק מהם הוא מורכב. יחד עם זאת, לצורך עיבוד התמונה בחרנו לעבור למרחב הצבע HSV. הסיבה העיקרית למעבר זה היא שבניגוד ל RGB הנעזר בצבעי יסוד, HSV קרוב יותר לאופן שבו העין האנושית תופסת צבע. ה HSV כולל שלושה רכיבים:

- Hue – מונח המבטא את הגוון של הצבע.
- Saturation – רוויה, עוצמה. מונח המתאר את האינטנסיביות/ריכוז של הצבע.
- Value – מונח המתאר את הבהירות ואת רמת הערבוב של הצבע עם שחור ולבן.

בפרויקט זה אנו מעוניינים לזהות עצמים בתנאי תאורה שונים. על כן נרצה שרמת ההצללה על האובייקטים לא תשפיע בצורה ניכרת על טיב הזיהוי. HSV עונה על דרישה זאת מכיוון שההבדל בין מצבי תאורה שונים לא משפיע על רכיב ה Hue בעוד שעבור מרחב הצבע RGB, ייתכן שינוי רב לכל אחת הקורדינטות.

כמו כן, באלגוריתם זה נעזרנו בספרית OpenCV של python. ספריה זו נפוצה בקרב מתכנתי אלגוריתמי עיבוד תמונה מכיוון שהיא מכילה פונקציות רבות וכן מסננים שונים שניתן להפעיל ביעילות על תמונות הקלט. למשל, מסנן שבו השתמשנו באלגוריתם עיבוד התמונה הינו מסנן ה- median filter. עבור כל פיקסל בתמונה, מסנן זה מסתכל על סביבת פיקסלים המקיפה את אותו פיקסל ומחשב את ערך החציון של סביבה זו. לאחר מכן המסנן מחליף את הפיקסל בערך החציון. המטרה בבחירת מסנן זה הינה הקטנת רעשי הרקע בתמונה.

הזיהוי עצמו של בני אדם כפי שנעשה באלגוריתם, מתבסס על זיהוי של עצמים "המפריעים" לתמונה המוקרנת על גבי הקרקע. במילים אחרות, התמונה המוקרנת ידועה מראש למתכנת וכאשר אדם מסתיר תמונה זו האלגוריתם מפרש זאת כזיהוי מוצלח. את האלגוריתם נתאר בעזרת שני שלבים עיקריים. ראשית, כל תמונה שנקלטת מומרת ממרחב הצבע RGB אל HSV. לאחר מכן מופעלת מסכת "איבר-איבר" (mask) על כל תמונה. כתוצאה מכך, כל פיקסל יזוהה כ-"הפרעה" (חלק מהאדם) אם הוא לא נכלל בטווח ערכי HSV שהוגדר מראש. טווח זה מציין למעשה את תחום ערכי HSV האפשרי שעבורו האלגוריתם יפרש את הפיקסלים כחלק מהתמונה המוקרנת. בשלב השני יופעל מסנן median להקטנת רעשי הרקע. לבסוף, בהתאם לכמות הפיקסלים המסווגים כ-"הפרעה" יוחלט אם יש זיהוי (אדם הגיע לנקודת העצירה) או אין זיהוי.

נציין כי על מנת לחסוך בסיבוכיות וזמן ריצה ומכיוון שהפריימים המהווים קלט לאלגוריתם מגיעים בקצב דגימה גבוה, בחרנו להוריד את קצב הדגימה. בחירה זו מתבצעת על ידי ניתוח של פריים אחד (תמונה אחת) מבין אוסף של פריימים עוקבים. ההצדקה לפעולה זו נובעת מכך שקצב הצילום הוא גבוה הרבה יותר מאשר קצב השינוי הנובע מתזוזה של האדם.

כמו כן, על מנת להימנע מזיהוי שגוי של עצמים הכלולים בפריים אך אינם שייכים לאדם, יצרנו מעין מנגנוני הגנה שונים. מנגנון הגנה ראשון הינו דרישה לזיהוי מוצלח ברצף של מספר פריימים עד להכרזה על "זיהוי מוצלח". במילים אחרות, האלגוריתם לא יתבסס על זיהוי עבור פריים יחיד, אלא ידרוש זיהוי ברצף של פריימים. מנגנון נוסף הינו דרישה למספר מינימלי של פיקסלים אשר יזוהו כפיקסלים של האדם. כלומר, כדי להימנע ממצב בו

הזיהוי הוא שגוי אך האלגוריתם יגדירו כמוצלח בשל מספר מועט של פיקסלים, האלגוריתם דורש זיהוי עבור כמות פיקסלים שצריכה להיות גדולה מערך סף מסוים. ערך סף זה נקבע על ידי המתכנת.

יתר על כן, אלגוריתם אלטרנטיבי אשר נבחן במהלך העבודה על הפרויקט מתבסס על זיהוי אובייקטים בעזרת מסנן מובנה מראש של ספריית OpenCV. מסנן זה נקרא cv2.HOGDescriptor. היתרון העיקרי של אלגוריתם זה הוא פשטות המימוש אך כפי שיוסבר בפרק ניתוח תוצאות, אלגוריתם זה הביא לביצועים לא מספקים ולכן בחרנו שלא להשתמש בו.

נתייחס כעת ל**מאפיינים מרכזיים של מצלמות** המשפיעים בצורה ניכרת על הביצועים של רחפן.

גודל ומשקל המצלמה: באופן כללי, מרבית הכוח והסוללה מושקעים בהטסת הרחפן. לכן, כדי להקל על פעולת ההטסה, מצלמה אופטימלית היא מצלמה בעלת משקל מינימאלי, כלומר, עד מאות בודדות של גרמים. בנוסף, נדרש שהמצלמה תהיה מממדים (גובה, רוחב ואורך) המתאימים לגודל הרחפן, על מנת לא לפגוע בתפקודו. בין היתר יש צורך שרוחב המצלמה לא יסטה מעבר לסף תנועת הפרופלורים.

שדה ראייה (FOV): מונח זה מתאר את התחום הנראה של השטח הנפרש מנקודת המבט של עדשות המצלמה. כאשר המצלמה משמשת לצורך תעופת הרחפן, שדה ראייה מאוד קטן עלול ליצור קשיים במעקב אחרי מסלול הרחפן ואחר סביבתו. מצד שני, ככל ששדה הראיה גדל, התמונה הנדגמת על ידי המצלמה מתעוותת ונהרסת.

באופן כללי, מצלמות עם שדה ראייה של מעל $110°$ נחשבות בעלות שדה ראיה רחב. בפרויקט זה, על מנת לקלוט את מרבית הסביבה של הרחפן, נעדיף שהמצלמה תהיה בעלת שדה ראיה רחב. כתוצאה מכך, החיישנים על גבי הרחפן יוכלו לשמור על נקודות האינפורמציה החשובות לאורך זמן יחסית ארוך, בהשוואה למצלמה עם שדה ראיה צר.

קצב צילום פריימים – frame per second (fps): מונח זה מתייחס לתדירות הצילום של תמונות עוקבות על ידי המצלמה. כל תמונה כזו נקראת פריים (frame) וכמות הפריימים הנלקחים בשנייה מתוארת ביחידות של fps. ככל שקצב הצילום גבוה יותר, כך ביצועי המצלמה במעקב אחר מסלול הרחפן יהיו טובים יותר והתמונות שיילקחו פחות יימרחו. כיום מצלמות רבות בימנו בעלות קצב צילום של 24 fps.

רזולוציה (כושר הפרדה): מושג זה מתאר את כמות הפיקסלים בתמונה. באופן כללי, ככל שהרזולוציה תהיה גבוהה יותר כך ניתן יהיה לבחון פרטים בתמונה בצורה חדה יותר. יחד עם זאת, רזולוציה מאוד גבוהה עלולה להכניס השהיה רבה למערכת מכיוון שיש צורך לקלוט ולעבד כמות גדולה של אינפורמציה. מצד שני, רזולוציה נמוכה עלולה לגרום לתמונות מטושטשות אך ללא השהיה רבה.

נתאר כעת **מאפיינים מרכזיים של מקרנים** המשפיעים בצורה ניכרת על הביצועים של רחפן.

גודל ומשקל המקרן: השיקולים בבחירת מקרן מבחינת גודל ומשקל זהים לשיקולים שתוארו בנוגע למצלמות.

עוצמת הארה: מושג זה מתאר את בהירות התמונה ואת עושר הצבעים בה. ככל שעוצמת ההארה גדולה יותר כך התמונה תהיה בהירה יותר מכיוון שיגיע יותר אור למסך. במקרה זה, הצבעים של התמונה ייראו עשירים יותר ועמוקים יותר. יחידות אופייניות לפרמטר זה הוא ANSI. באופן כללי מקרנים עם ערך ANSI נמוך מ-1000 נחשבים בעלי עוצמת הארה יחסית נמוכה. יחד עם זאת, במקרה של רחפנים שיקול משמעותי יותר הוא גודל ומשקל המקרן ומקרנים זעירים הם לרוב בעלי עוצמת הארה נמוכה.

רזולוציה: השיקולים בבחירת מקרן מבחינת רזולוציה זהים לשיקולים שתוארו בנוגע למצלמות.

כניסות ודרכי תקשורת: שיקול עיקרי בבחירת המקרן הוא אופן התקשורת בינו לבין מערכת ההפעלה linux, שבה נעשה שימוש בפרויקט. ישנם מקרנים שניתן לחברם לרחפן דרך חיבור אלחוטי של Bluetooth או wi-fi. יחד עם זאת, מקרנים רבים אינם מובנים לעשות זאת. שיטות אחרות לחיבור הן דרך כניסות וחיבורים פיזיים. למשל, חיבור של HDMI ניתן לביצוע על linux וזו הדרך בה התבצע החיבור בפרויקט.

3 סימולציה

על מנת לבחון את נכונות האלגוריתמים בפרויקט נוצרו סימולציות שונות עבור חלקים שונים בפרויקט. נתאר כל שלב בתהליך על פי הסדר הכרונולוגי בו בוצע הפרויקט.

בקרת הרחפן:

מטרת הסימולציה בחלק זה היא לבחון האם הרחפן מצליח לשמור על מיקומו באוויר בצורה יציבה וללא תנודות רבות.

טרם הרצת הסימולציה הקמנו תקשורת בין ה super computer לרחפן על ידי נקודת wi-fi של הרחפן עצמו. המחשב השולט על מימוש הסימולציה (למשל, מחשב של סטודנט) מחובר ל super computer בעזרת פקודת ssh המעניקה גישה לכל קבצי הפרויקט. תהליך הרצת הסימולציה כלל שליחת "פקודת המראה" מהצורה:

$ rostopic pub --once /bebop/**takeoff** std_msgs/Empty

לאחר שהרחפן המריא בהצלחה לגובה ידוע שנקבע מראש, נשלחה פקודה נוספת שמטרתה התחלת תהליך הבקרה של הרחפן בעזרת טכנולוגית SLAM. לצורך ההמחשה, הפקודה שנשלחה היא:

$ rostopic pub --once /bebop/**state_change** std_msgs/Bool "data: true"

לאחר מכן, כדי לבחון האם ההתייצבות אכן מדויקת והרחפן מצליח להתייצב באוויר מבלי לסטות הרבה, מנחה הפרויקט יצר הפרעה חיצונית לרחפן. בתגובה לכך היה על הרחפן לתקן את טיסתו ולחזור חזרה למקום ההתייצבות מבלי לסטות. תהליך זה חזר על עצמו מספר פעמים בתנאי סביבה שונים על מנת לוודא שאכן חלק זה מומש כנדרש. ניתן לראות המחשה של הסימולציה בסרטונים "תצוגה עם ריחוף והפרעות" ו-"תצוגה עם ריחוף והפרעות ונחיתה".

ניווט הרחפן בין נקודות:

מטרת הסימולציה בחלק זה הייתה לבדוק מעבר יציב בין נקודות שונות במרחב המוגדרת מראש. לצורך הרצת הסימולציה בוצעו השלבים שתוארו בחלק "בקרת הרחפן". לאחר שהרחפן הצליח להתייצב בהצלחה באוויר נשלחה פקודה חדשה שתפקידה להתחיל את תעופת הרחפן על פי המסלול הנתון. הפקודה שנשלחה הינה:

$ rostopic pub --once /bebop/**path_change** std_msgs/Bool "data: true"

לאחר כל סימולציה שבוצעה ווידאנו שהמעבר בין נקודות שונות במסלול לא מהיר מדי ולא איטי מדי, כלומר, בקצב תזוזה אופייני לאדם. כמו כן, דאגנו למעבר חלק ורציף לאורך המסלול. תהליך זה חזר על עצמו עבור מסלולים שונים על מנת לוודא את נכונות האלגוריתם עבור המקרה הכללי. ניתן לראות המחשה של הסימולציה בסרטון "ריבוע" המציג את תעופת הרחפן על פני מסלול ריבועי.

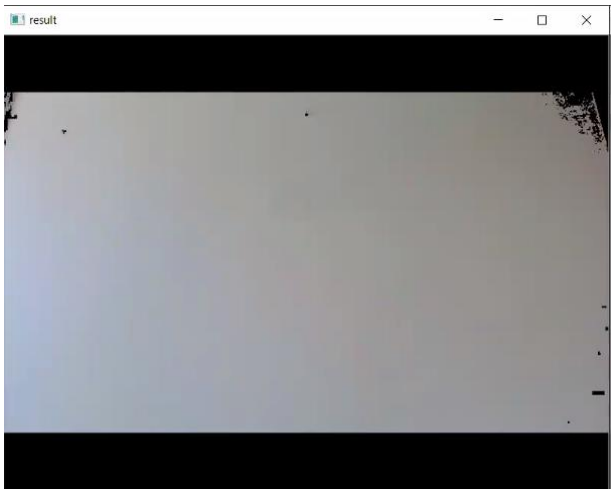
זיהוי עצמים מפריעים במרחב:

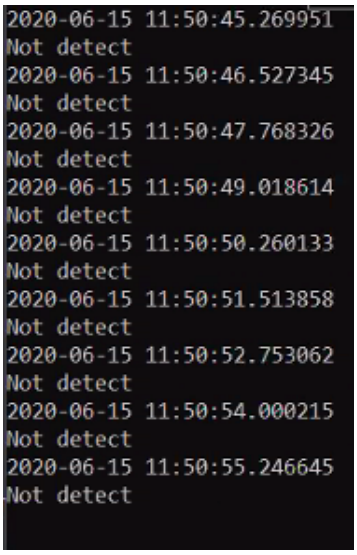
מטרת הסימולציה בחלק זה הינה בדיקת הביצועים של זיהוי עצמים "המפריעים" לתמונה מוקרנת על גבי הקרקע. לצורך בדיקת נכונות האלגוריתם הוחלט שהאלגוריתם יפלוט למסך הודעת "יש זיהוי" שמשמעותה שאדם או חפץ מסתירים "רקע" ואחרת תישלח הודעת "אין זיהוי". כמו כן, ב-"רקע" הכוונה לתמונה המוקרנת מהמקרן אך לצורך הסימולציה נעזרנו ברצפת המעבדה או קיר לבן. נמחיש את נכונות האלגוריתם על ידי צילומי המסך הבאים שנלקחו מאחת הסימולציות שבוצעו. בסימולציה זו נעזר במונחים ובהגדרות הבאות:

- "העצם המפריע" הינו דסקית שחורה
- ה-"רקע" הוא קיר לבן.
- חלון "mask2". חלון זה תמיד מופיע בצבעי שחור לבן ומתאר בלבן את "הרקע" ובשחור את העצמים המפריעים.
- חלון "result". חלון זה מופיע בצבע ומתאר את תוצאת הפעלת המסנן על הפריים הנקלט. התוצאה המופיעה בחלון זה היא למעשה תמונת הקלט לאלגוריתם אך עם אזורים מושחרים בהם מופיע העצם המפריע. כאשר כמות הפיקסלים המושחרים עוברת ערך סף מסוים הנקבע על ידי המתכנת, האלגוריתם מפרש זאת כזיהוי.
- חלון command line. בחלון זה ניתן לראות את הזמן בו בוצעה הסימולציה וכן את הודעת " Not detect" או "Detect".

תוצאות הסימולציה טרם הכנסת האובייקט המפריע:

ניתן לראות שבמצב ההתחלתי אין זיהוי של אובייקט ועל כן מודפס למסך רק הודעות "Not detect".

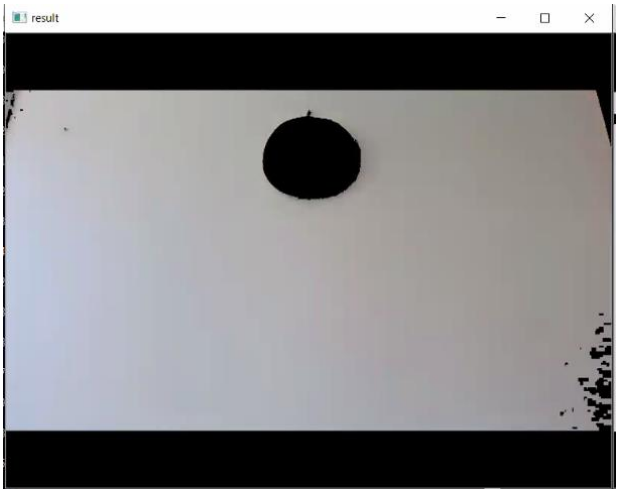


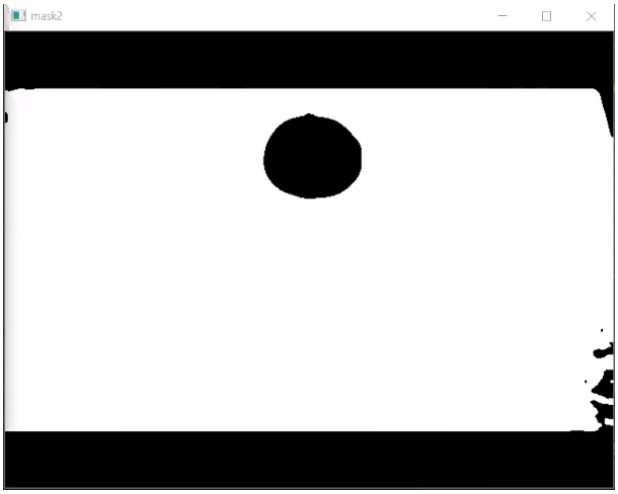


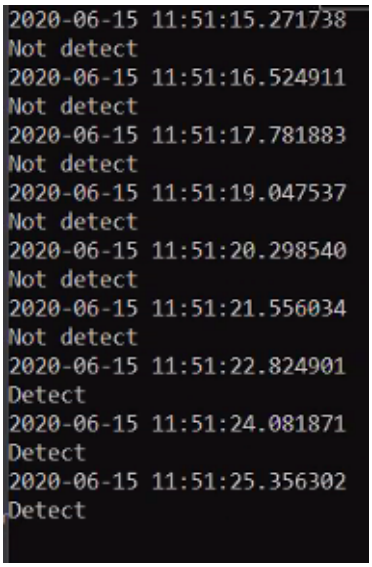
איור 2. סימולציית "זיהוי עצמים מפריעים במרחב" טרם הכנסת עצם מפריע. מימין מופיע חלון mask2, משמאלו מופיע חלון result ובשורה השנייה מופיע חלון ה command line המציג את הודעת הזיהוי.

תוצאות הסימולציה לאחר הכנסת האובייקט המפריע:

באמצעות צילום ה command line ניתן לראות שבזמן 11:51:22 נקלטה ההפרעה על ידי האלגוריתם והחל מאותו רגע הודפסה למסך הודעת "detect". נציין כי הזיהוי התבצע בזמן אמת ובזמן תגובה מהיר מזמן התגובה של העין האנושית.







איור 3. סימולציית "זיהוי עצמים מפריעים במרחב" לאחר הכנסת עצם מפריע. מימין מופיע חלון mask2, משמאלו מופיע חלון result ובשורה השנייה מופיע חלון ה command line המציג את הודעת הזיהוי.

<u>עיבוד תמונה במצלמת intel RealSense בשילוב אלגוריתם ההקרנה:</u>

מטרת הסימולציה הייתה לבדוק את נכונות הזיהוי של אלגוריתם עיבוד התמונה וכן את האלגוריתם "projector_ros". כפי שיפורט בפרק המימוש, אלגוריתם זה אחראי על מתן הוראות למקרן בהתאם למסלול התעופה ולמיקום הרחפן על גבי המסלול. לצורך כך הרצנו את האלגוריתם על ידי הפעלתו בשלבים שונים לאורך המסלול. נכונות האלגוריתם נבחנה במעבר בין הקרנת "תמרור עצור" לבין הקרנת "חץ ירוק" ולהפך וכן בסיום המסלול לצורך בדיקת הקרנת סימן סיום המסלול.

בסרטון "עיבוד תמונה" ניתן לראות המחשה לסימולציה זו. כפי שניתן לראות, האלגוריתם פולט כנדרש את סימני ההקרנה הנכונים בהתאם למסלול. כאשר המצלמה של הרחפן מזהה שיש "עצם מפריע" (הקופסה האדומה) המקרן יהיה מודע לכך ותינתן הוראה להקרנת סימן של חץ ירוק, שמשמעותו שניתן להמשיך במסלול התעופה.

<u>סימולציה מאוחדת המשלבת עיבוד תמונה עם הקרנה לאורך מסלול מוגדר מראש:</u>

בסרטון "עיבוד תמונה בשילוב עם הקרנה" ניתן לראות תוצאת סימולציה זו. בסימולציה ניתן לראות שהרחפן מנווט אדם לאורך מסלול ריבועי במרחב על ידי סימנים ניווט מתאימים. כאשר הרחפן מגיע לנקודה המהווה פינה של ריבוע הרחפן עוצר במקום ומחכה לאדם. כאשר הרחפן מזהה שהאדם הגיע לנקודות הביניים ניתן להמשיך במסלול אל נקודת הביניים הבאה. ניתן לראות שבסיום המסלול הרחפן מקרין סימן המעיד על סיומו המוצלח של המסלול. נציין כי בסימולציה זו הדגש היה על עיבוד התמונה וההקרנה ולא על תעופת הרחפן. לכן המסלול בין נקודות ביניים נמדד לפי משך זמן של מקטע ולא לפי קורדינטות מרחביות.

4 מימוש

השלב הראשון במימוש הפרויקט התמקד באלגוריתם הבקרה שמטרתו תפעול והטסת רחפן מסוג "Bebop". לשם כך נעזרנו ב supercomputer מסוג raspberry pi וכן במצלמת אינטל מסוג RealSense T265 המותקנת על גבי הרחפן. האלגוריתם מקבל מהמצלמה את המיקום המרחבי של הרחפן בזמן אמת ובעזרת קלט זה מבצע חישובי בקרה שיאפשרו לו לקבוע את המהירות האופטימלית לתעופה.

מימוש זה התבצע בהצלחה וביעילות. הרחפן ביצע ניווט אווירי לאורך מסלול מוגדר מראש על ידי בקרה עצמית ובאופן אוטונומי. אמנם המימוש התבצע בהצלחה אך במהלך ניסויים נוספים הרחפן החל לקרוס במהלך התעופה ובאופן שרירותי ולא תלוי באלגוריתם שהורץ. לאחר ניסויים רבים התברר כי הרחפן לא יהיה מסוגל לשאת במהלך מעופו גם את כובד משקלם של מצלמה נוספת ומקרן אשר הינם חלק מהפרויקט ולא היו מתוקנים עד לשלב זה. בעצת המנחה הוחלט להחליף את סוג הרחפן ומרגע זה העבודה הותאמה לרחפן חדש מסוג "DJI" הנעזר במקום ב raspberry pi ב- supercomputer בשם Jetson TX2.

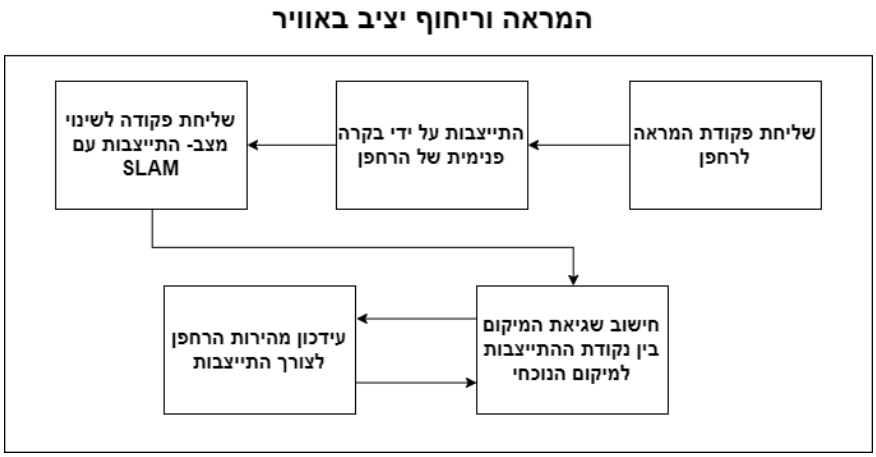
לצורך ההחלפה הותקנו תוכנות שונות על גבי כלי הפיתוח החדש: המערכת של Jetson TX2, Ubuntu 16, ROS kinetic, וכן סביבת העבודה של הרחפן. בנוסף, יובאה הפלטפורמה המתאימה לצורך הפעלת מצלמות Intel RealSense.

<u>נציין כי אלגוריתם הבקרה לא הותאם לרחפן ה- DJI, משימה זו הוטלה על צוות אחר במעבדת הרחפנים.</u>

מעתה והלאה השלב הראשון בפרויקט בו התבססנו על רחפן ה bebop יתואר כ-"חלק א'" ואילו השלבים הבאים בפרויקט, אשר התבצעו לאחר החלפת הרחפן לרחפן DJI יתוארו כ-"חלק ב'".

את המערכת הכוללת נפרק כעת למספר שלבים ונסביר על כל שלב בנפרד.

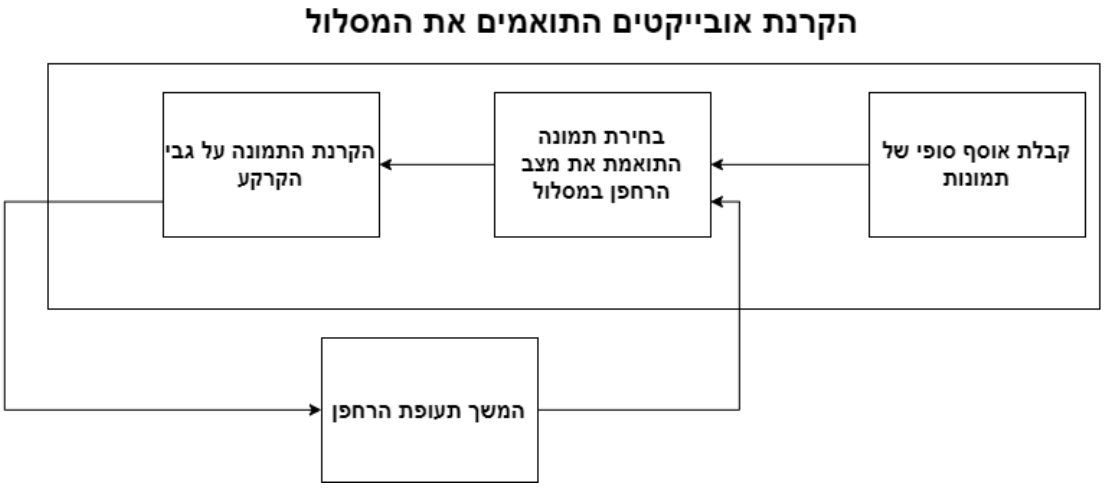
<u>המראה וריחוף יציב באוויר:</u>



איור 4. המראה וריחוף יציב באוויר

שלב ההמראה של הרחפן וריחופו באוויר מומשו בעזרת אלגוריתם הכתוב בשפת C++ הנקרא "drone_bebop_control.cpp". על ידי יישום אלגוריתם זה בפרויקט שלנו, ניתן "להפריע" לרחפן בצורה מכוונת והרחפן בתגובה מנסה לחזור למיקום היציב שלו. תכונה זו מסייעת רבות בהתייצבות הרחפן באוויר ובתנאי סביבה שונים. נציין כי אלגוריתם זה הותאם לרחפן ה- bebop.

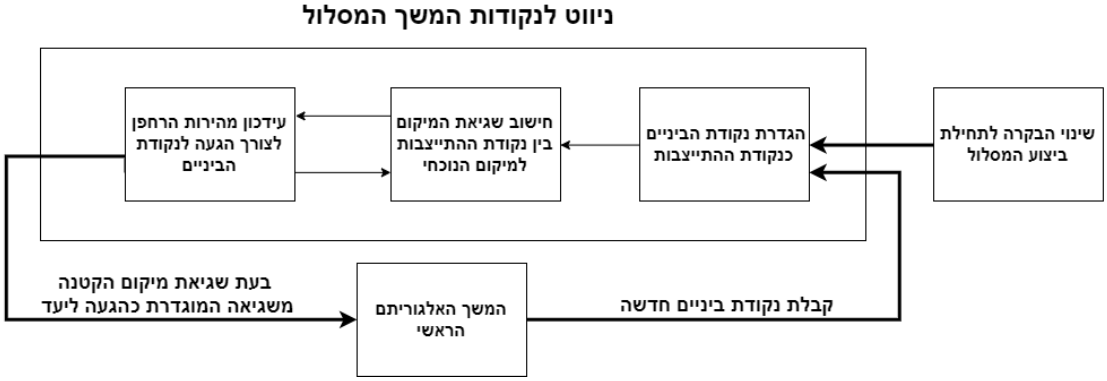
הקרנת אובייקטים התואמים את המסלול:



איור 5. הקרנת עצמים התואמים את המסלול

בשלב זה מומש אלגוריתם בשפת python הנקרא "projector_ros.py" המקבל כקלט אוסף סופי של תמונות ומקרין על הרצפה תמונה מסוימת. התמונה המוקרנת תלויה במיקום הרחפן על המסלול וכן בנקודת העצירה הבאה.

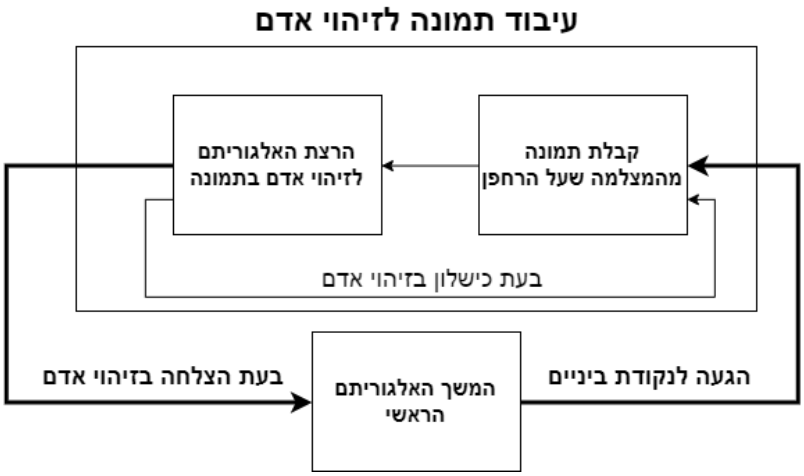
ניווט לנקודות המשך המסלול:



איור 6. ניווט לנקודות המשך במסלול

בלוק זה מומש גם הוא על ידי אלגוריתם הבקרה הנקרא "drone_bebop_contol.cpp" וזאת על מנת שהרחפן יצליח לבצע מעבר בין נקודות במרחב כחלק ממסלולים שונים. אלגוריתם זה מכונה בדיאגרמה כ-"האלגוריתם הראשי". המסלולים השונים נבחרים מראש טרם התעופה על ידי המשתמש. למעשה, באלגוריתם זה ניתן לממש כל מסלול מוגדר מראש בהינתן נקודות העצירה השונות. למשל, בפרק הסימולציה הוצג מסלול ריבועי ולצורך כך נקבע האורך של כל צלע וכן נקבע כיוון המסלול (עם כיוון השעון או נגדו). הקוד עצמו מתייחס לבחירה זו ומותאם לכך. תהליך בחירת מסלולים בצורה זו הוביל לכך שבעת העבודה על הפרויקט נוצרו מספר אלגוריתמים כך שכל אחד מייצג מסלול אחר.

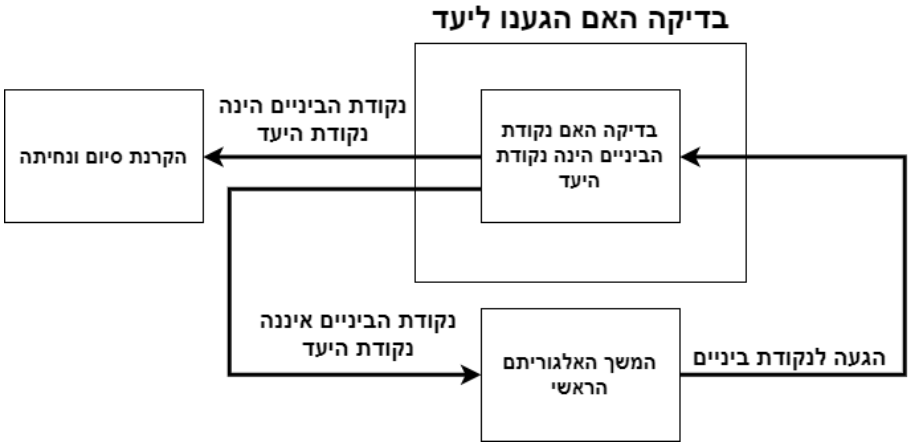
עיבוד תמונה לזיהוי אדם:



איור 7. עיבוד תמונה לזיהוי האדם

למימוש חלק זה נחקרו אלגוריתמים שונים לזיהוי עצמים ובפרט לזיהוי בני אדם. נשקלו שני סוגים עיקריים של מימושים. סוג ראשון הינו מימוש המתבסס על אלגוריתמים של למידת מכונה. מימוש זה נפסל בעקבות צורך בזיכרון רב מדי על ה-supercomputer. המימוש השני מתבסס על אלגוריתם הנעזר בספריית OpenCV וללא שימוש בלמידת מכונה כלל. בסופו של דבר הוחלט על דרך פתרון זו בזכות גמישותה ואחוזי ההצלחה הגבוהים. האלגוריתם שנכתב נקרא "object_detect.py" והידע התאורטי בעיבוד תמונות שעליו האלגוריתם מתבסס, מוצג ברקע תיאורטי.

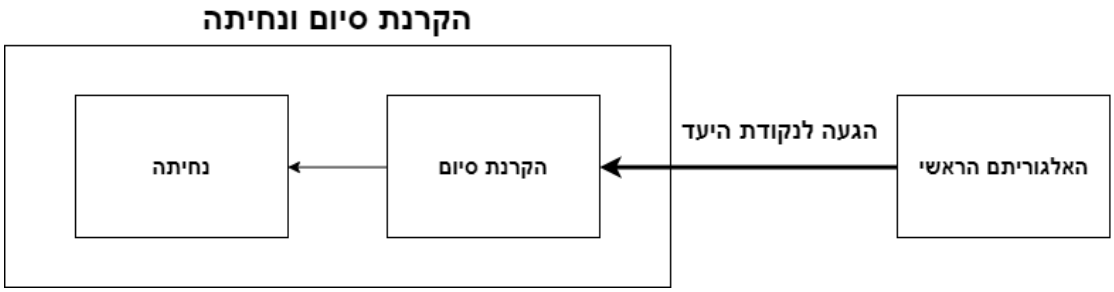
בדיקה האם הגענו ליעד:



איור 8. בדיקה האם הגענו ליעד

בשלב זה נבדק האם נקודת הביניים הנוכחית הינה נקודת היעד. מאחר ואלגוריתם הבקרה "drone_bebop_control.cpp" בעל גישה למסלול הניווט, מתבצעת בו הבדיקה ובהתאם לכך נקבע אופי המשך המסלול וכן סימני ההקרנה הבאים.

הקרנת סיום ונחיתה:

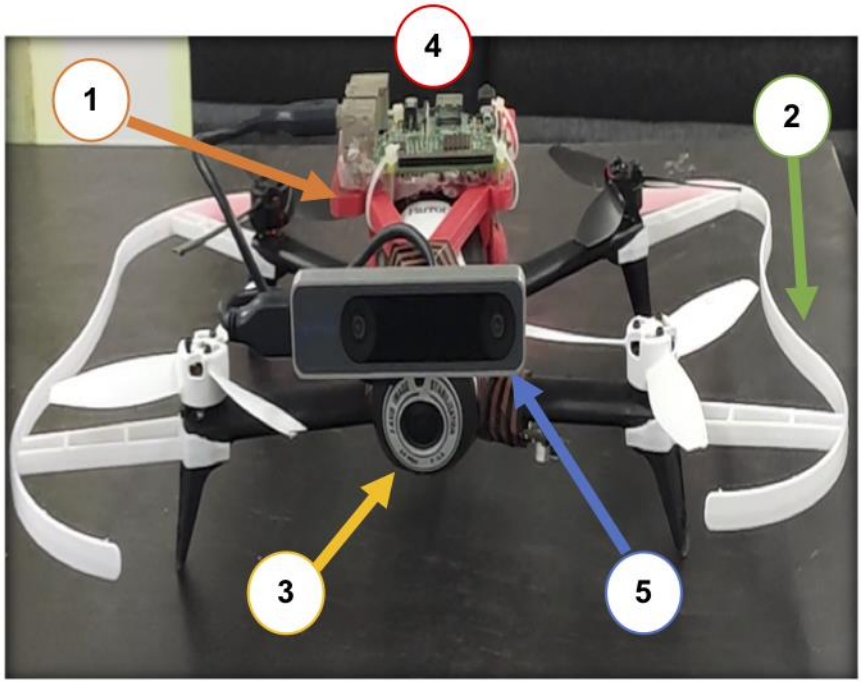


איור 9. הקרנת סיום ונחיתה

שלב זה ממומש בעזרת שני אלגוריתמים. באלגוריתם ההקרנה הנקרא "projector_ros.py" מתבצעת הקרנת על גבי הקרקע של אובייקט המעיד על סיום המסלול. לאחר מכן אלגוריתם הבקרה "drone_bebop_control.cpp" מבצע נחיתה ובכך התהליך הכולל מסתיים.

תיאור חמרה

<u>באיור הבא מוצג רחפן ה bebop שעליו רכיבים שונים בהם נעשה שימוש בחלק א' של הפרויקט:</u>



איור 10. רחפן ה- Parrot Bebop 2

1. **מנשא עבור ה supercomputer raspberry pi על גבי רחפן ה bebop.**
2. **מגני תעופה.**
3. **רחפן Parrot Bebop 2:**
   - זמן סוללה: 25 דקות רצופות.
   - משקל: 500 g.
   - כולל מעבד Dual core processor עם quadcore GPU.
   - תומך ב Wi-Fi 802.11a/b/g/n/ac.
   - בעל מהירות אופקית מקסימלית: $16\frac{m}{s}$.
   - בעל מהירות אנכית מקסימלית: $6\frac{m}{s}$.

תמונה זה לקוחה מהאתר המופיע ב [2]

4. **Supercomputer Raspberry pi:**
   - כולל מעבד Broadcom BCM2711, Quad core Cortex-A72 64-bit SoC @ 1.5GHz
   - בעל כניסות 2 × micro-HDMI ,2 USB 3.0 ports; 2 USB 2.0 ports (up to 4kp60 supported) ports.
   - מאפשר אספקת זרם של 2.5 A.

תמונה זה לקוחה מהאתר המופיע ב [1].

5. **Intel RealSense T265 camera:**

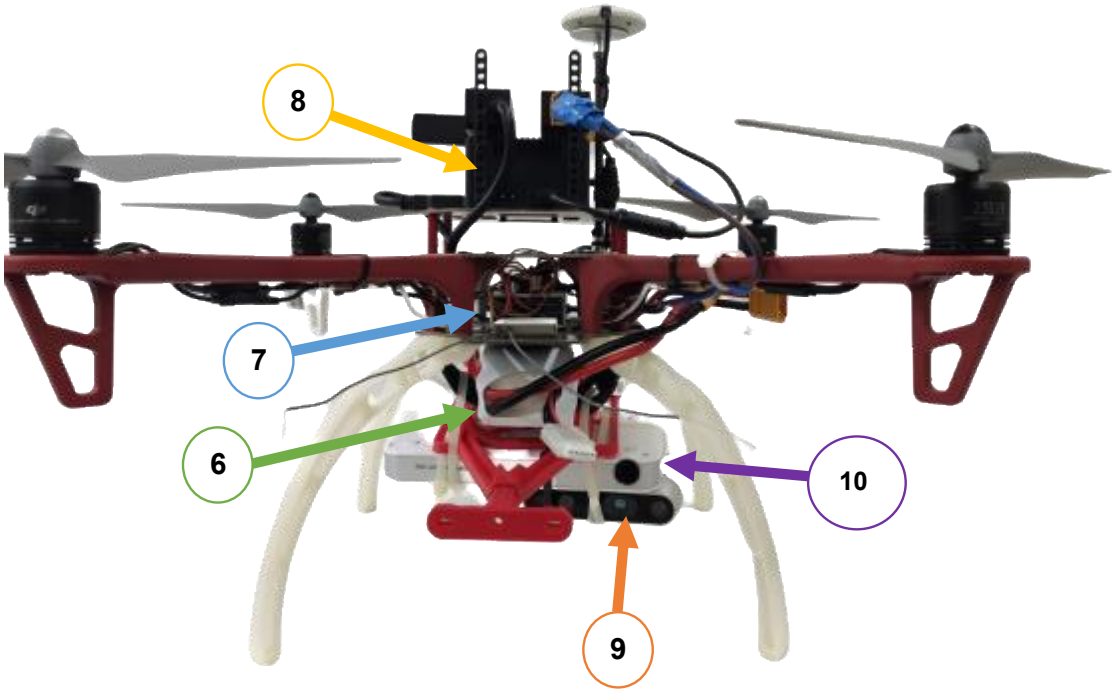
בעזרת שתי מצלמות OV9282 מובנות וג׳ירוסקופ רכיב זה מאפשר למצוא את המיקום המרחבי של הרחפן. תכונותיו העיקריות:

- שדה ראיה של $163 \pm 5°$.
- כניסת USB 3.1 Micro B.
- ממדים: $108 \times 24.5 \times 12.5\ mm$.
- רזולוציה: 848×800
- קצב צילום: $30\ fps$.
- משקל: $60\ gr$.
- פועל בהספק נמוך יחסית של $1.5\ W$.
- יכול לעבוד בתוך מבנה ומחוץ לו אך לא במצבי קיצון של אור או חושך מוחלט.

נציין כי בעזרת מצלמה זו ניתן לממש את טכניקת הפעולה של אלגוריתמי SLAM כפי שהוסבר בפרק רקע תיאורטי.

תמונה זה לקוחה מהאתר המופיע ב[3].

<u>באיור הבא מוצג רחפן ה DJI שעליו רכיבים שונים בהם נעשה שימוש בחלק ב׳ של הפרויקט:</u>



איור 11. רחפן ה- DJI.

6. **סוללה.**

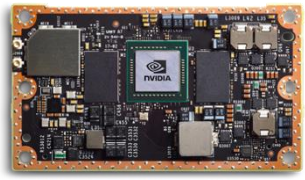
סוללת הרחפן הינה סוללת 5200 mAh של 4 תאים, כל תא 3.7 V, סה"כ 14.8 V.[2]

[2] מידע זה מופיע על גבי הסוללה.

7. רחפן מסוג DJI עם רכיבי Manifold 2 ו N3 flight controller:

- תומך ב Wi-Fi 802.11a/b/g/n/ac.
- הספק: $3 - 25\ W$.
- מהירות המראה: $5\frac{m}{s}$.
- מהירות נחיתה: $4\frac{m}{s}$.
- סטייה ממצב יציב: $\pm 0.5m$ בציר האנכי ו $\pm 1.5m$ בציר האופקי.

8. Supercomputer Jetson TX2:



- בעל מעבד Dual-Core NVIDIA Denver 2 64-Bit CPU
- בעל זיכרון 8GB 128-bit LPDDR4
- אחסון 32GB eMMC 5.1
- הספק 7.5W / 15W

תמונה זה לקוחה מהאתר המופיע ב [6].

9. Intel RealSense D435 camera:

- שדה ראיה $(H \times V \times D)$ $69.4° \times 42.5° \times 77°(\pm 3°)$.
- כניסת USB-C* 3.1.
- ממדים: $90 \times 25 \times 25\ mm$.
- רזולוציה: 1920×1080
- קצב צילום: $30\ fps$.
- משקל: $72\ gr$.

תמונה זה לקוחה מהאתר המופיע ב [7].

10. Laser beam pro projector:

- ממדים: $70.9 \times 140.5 \times 18.8\ mm$.
- כניסה: Micro HDMI.
- בהירות: $100\ ANSI - 200\ Lumens$.
- Throw distance: $0.61\ m - 4.6\ m$
- משקל: $260\ gr$.
- רזולוציה: 1366×768.

תמונה זה לקוחה מהאתר המופיע ב [8].

תיאור תוכנה

עיקר הפרויקט מתבצע במערכת ההפעלה של linux על פלטפורמה הנקראת Robot Operating System או בקיצור ROS. פלטפורמה זו משמשת בעיקר לתפעול רובוטים ובפרט רחפנים. בנוסף, רוס (ROS) היא סביבת עבודה פתוחה אשר מספקת שירותים שונים כגון אבסטרקציה בתהליך הפיתוח, בקרת שליטה בשפת תכנות low level, שליחת מסרים (messages) בין תהליכים שונים וניהול חבילות (packages).

לשימוש בפלטפורמת רוס בפרויקט זה יש מספר יתרונות. ראשית, כיום מימוש יעיל לתפעול רובוטים מסתמך על מספר רב של תהליכים (processes) אשר ממומשים על ידי מחשבים שונים. מכאן הצורך הרב בתקשורת טובה בין התהליכים שלא תהיה מוגבלת למכשיר פיזי אחד. רוס הוא פלטפורמה אשר מאפשרת קישוריות כזו בצורה פשוטה ונוחה למימוש. יתרון משמעותי נוסף לרוס הוא התמיכה הנרחבת שמשתמשי רוס יכולים לקבל על ידי הקהילה. במילים אחרות, רוס הפך לכלי סטנדרטי בקרב קהילת מתכנתי הרובוטים ובכך מהווה פלטפורמה עדכנית הן מבחינת עדכוני חומרה והן מבחינת אלגוריתמיקה. למשל, ישנן חבילות מובנות רבות (ROS packages) הניתנות לשימוש כמקורות מידע טובים ואמינים.

<u>מבנה כללי לפרויקט בפלטפורמת רוס</u>

ראשית נתאר מושגים שימושיים לצורך ניתוח המבנה הכללי:

- **Node**. Node הוא למעשה instance, כלומר, כל node הוא אובייקט המוגדר על ידי קובץ src יחיד וכל קובץ src מהווה קובץ הרצה בעבור אותו node. קובץ src יכול להיכתב בשפת python או c++.
- **Message**. Message הינו מבנה להעברת מידע בין ה nodes שונים. ה node השולח message נקרא publisher ואילו ה node המקבל את ה message נקרא subscriber.
  כל הודעה ניתנת לשליחה על גבי topic מסוים. למשל, publisher שולח messages מסוים על topic אחד או יותר ואילו subscriber הרוצה לקבל אינפורמציה מסוימת, מאזין ל topic אחד או יותר אשר הוא מעוניין בהם.

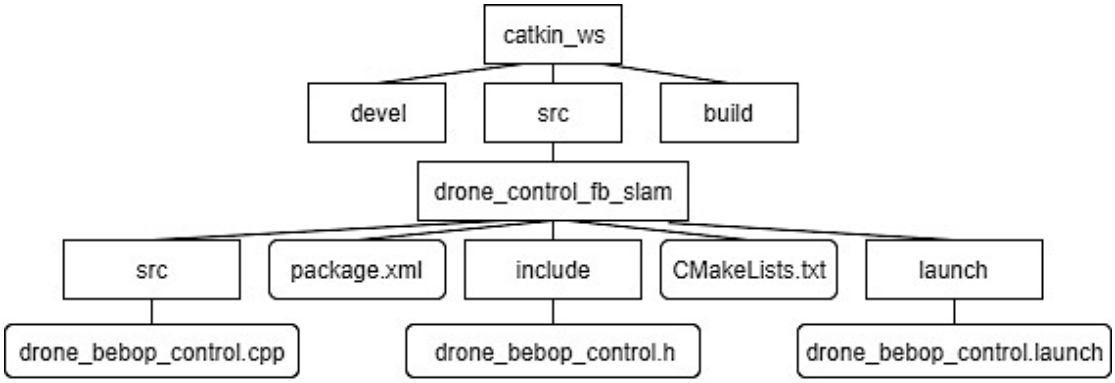
נתאר כעת את הכלים אשר באמצעותם מתבצעת התקשורת בין החלקים השונים בפרויקט ברוס. לאחר מכן נתמקד באופן פרטני יותר בתכנון שלנו. באופן כללי, רוס מאפשר למשתמש אחד או יותר לעבוד על פרויקטים שונים על ידי סביבות עבודה שונות. למשל, שם מקובל לסביבת עבודה הוא "catkin_ws" (catkin workspace). בכל סביבת עבודה ניתן לייצור packages שונים אשר יכולים לתקשר ביניהם כפי שנסביר בהמשך. בכל package ישנן תתי תיקיות אשר חיוניות לתקשורת תקינה:

- **תיקיית src**: זוהי תיקיית source files אשר מכילה את הקבצים עם האלגוריתמים השונים בהם נעשה שימוש בפרויקט. הקבצים יכולים להיכתב בשפת c++ או python.
- **תיקיית launch:** בתיקייה זו ישנם קבצי launch שהרצתם מאפשרת הפעלת nodes שונים בו זמנית.
- **תיקיית include:** תיקייה אופציונלית להוספת קבצי header files.
- **תיקיית msg:** תיקייה אופציונלית עבור קבצי קוד ליצירת messages חדשים.

בנוסף לתיקיות אלו, קיימים גם שני קבצים חשובים אשר מקשרים בין המידע העובר בין התיקיות:

- **CmakeLists.txt** – קובץ המכיל רשימה של הוראות בנייה וההרצה על מנת לאפשר את התפעול התקין של כל קבצי ה package.
- **Package.xml** – קובץ זה מכיל את המאפיינים של ה package כמו למשל שם ה package, גרסה ו- dependencies.

כפי שהוסבר בהקדמה לפרק זה, בשל אילוצים שונים הפרויקט מתפצל לשני חלקים. בחלק א׳ סביבת העבודה נקראת "catkin_ws" ואילו ה-package נקרא drone_control_fb_slam. בחלק זה ישנו קובץ src יחיד הכתוב בשפת c++ וכן קובץ launch יחיד המתאים לו. בסכמה הבאה מתואר מבנה הפרויקט בחלק א':



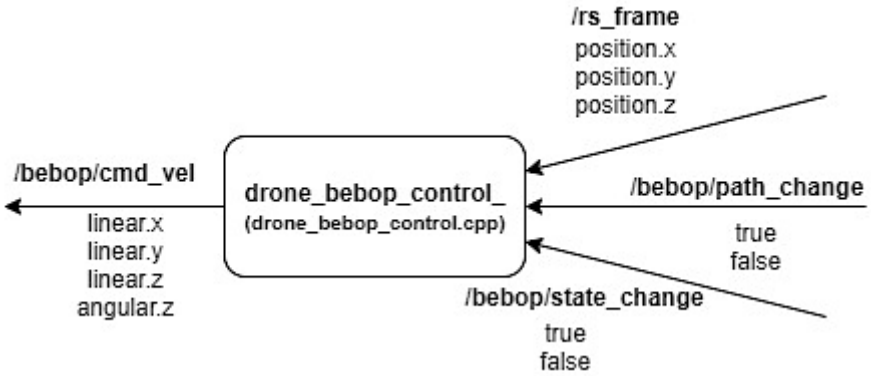
איור 12. מבנה חלק א׳ בפרויקט

בטבלה הבאה מוצג ה node הרלוונטי בחלק זה.

שם ה node: drone_bebop_control_	
שם קובץ src	Drone_bebop_control.cpp
שם קובץ launch	obj_detect_launch.launch
רשימת topics עליהם יתבצע publisher:	• /bebop/cmd_vel
רשימת topics עליהם יתבצע subscribe:	• /rs_frame • /bebop/path_change • /bebop/state_change

טבלה 1. מאפיינים עיקריים של ה node הנקרא drone_bebop_control_.

לצורך ההמחשה, באיור הבא מתואר הקשר בין ה node, drone_bebop_control_, ל topics שדרכם עוברת התקשורת.



איור 13. דרכי התקשורת עם ה node הנקרא drone_bebop_control_.

נסביר איור זה.

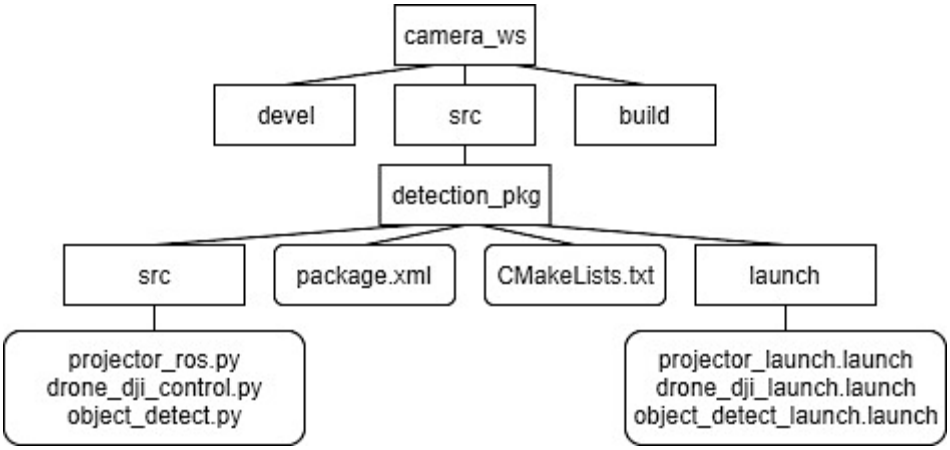
לאחר ההמראה ולמעשה במהלך כל זמן התעופה, node זה מקבל אינפורמציה דרך ההאזנה ל topic שנקרא /rs_frame. Topic זה מפורסם על ידי מצלמת RealSenseT265. כמו כן, ההודעה (message) המועברת דרך topic זה היא המיקום בזמן אמת של המצלמה במרחב, ביחס למיקום הרחפן טרם ההמראה.

בנוסף, במצב ברירת המחדל ההודעות המועברות דרך /bebop/path_change ו- /bebop/state_change הינן false. במצב זה הרחפן מתייצב באוויר בעזרת בקרה פנימית מובנת. כאשר המשתמש מפרסם true עבור topic ה- /bebop/state_change, ה node יבחין בכך וכתוצאה מכך הרחפן ייעזר באלגוריתם הבקרה, כפי שתואר בפרקים קודמים. לאחר מכן, כאשר המשתמש מפרסם true עבור topic ה- /bebop/path_change, ה- node יבחין בכך וכתוצאה מכך הרחפן יתחיל בניווט על פי המסלול שנקבע עבורו.

כמו כן, במקביל ה- node מפרסם על גבי topic ה- /bebop/cmd_vel את מהירות הטיסה של הרחפן עבור כל אחד מהצירים המרחביים וכן את המהירות הזוויות לאורך ציר z. מהירויות אלו נקבעות על פי האלגוריתם הכתוב בקובץ "drone_bebop_control.cpp" והן ייקבעו את אופן טיסת הרחפן. למשל, כאשר הרחפן יגיע לנקודת

עצירה מסוימת על גבי המסלול, האלגוריתם ידע לפרסם מהירויות השוות לאפס בכל אחד מהצירים. אלגוריתם בקרה זה מתואר ברקע תיאורטי ובפרקים קודמים.

בחלק ב׳ סביבת העבודה נקראת "camera_ws" ואילו ה package נקרא detection_pkg. package זה מכיל שלושה קבצי src הכתובים בשפת python וכן שלושה קבצי launch המתאימים לכל אחד מקבצי ה src. סכמה המתארת את מבנה הפרויקט שלנו בחלק ב׳ מוצגת באיור הבא:



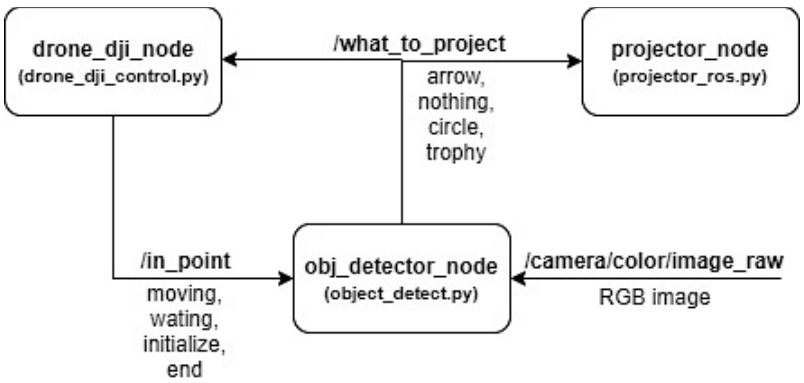
איור 14. מבנה חלק ב׳ בפרויקט.

הטבלה הבאה מציגה את מבנה הpackage באמצעות מושגי התקשורת כפי שהוצגו לעיל. כפי שניתן לראות, הטבלה המוצגת כוללת תתי טבלאות וכל תת טבלה מתייחסת ל node אחר.

שם הnode: obj_detector_node	
שם קובץ src	Object_detect.py
שם קובץ launch	obj_detect_launch.launch
רשימת topics עליהם יתבצע publisher:	• /what_to_project
רשימת topics עליהם יתבצע subscribe:	• /camera/color/image_raw • /in_point
שם הnode: projector_node	
שם קובץ src	projector_ros.py
שם קובץ launch	projector_launch.launch
רשימת topics עליהם יתבצע publisher:	• אין
רשימת topics עליהם יתבצע subscribe:	• /what_to_project
שם הnode: drone_dji_node	
שם קובץ src	Drone_dji_control.py
שם קובץ launch	drone_dji_launch.launch
רשימת topics עליהם יתבצע publisher:	• /in_point
רשימת topics עליהם יתבצע subscribe:	• /what_to_project

טבלה 2. מאפיינים עיקריים של ה nodes בחלק ב׳ של הפרויקט.

לצורך ההמחשה, באיור הבא מתואר הקשר בין ה node, drone_bebop_control_, ל topics שדרכם עוברת התקשורת.



איור 15. דרכי התקשורת עם ה nodes השונים בחלק ב׳.

נסביר כעת את איור זה. כפי שנאמר קודם לכן, בחלק א׳ של הפרויקט הרחפן טס באוויר על ידי המהירויות הנקבעות על פי אלגוריתם הבקרה. בחלק ב׳ של הפרויקט נניח ש drone_dji_node מייצג את אלגוריתם בקרה זה ולמעשה שקול לdrone_bebop_control node כפי שתואר לעיל. ההפרדה היא בשל אילוצי הפרויקט כפי שהוסבר בתחילת פרק המימוש. לכן, נוכל לתאר סכימה זאת בהמשך לתיאור שבוצע בעבור הסכימה המוצגת באיור 13.

לאחר ההמראה, ה- drone_dji_node מפרסם על גבי ה topic, in_point/, הודעה של initialize שמשמעותה תחילת המעבר על מסלול הניווט. מיד לאחר מכן, drone_dji_node מפרסם על גבי topic זה הודעה לפיה הרחפן נמצא במהלך תעופה, הודעה זו נקראת moving. כאשר obj_detector_node מבחין בהודעת ה moving, הוא יפרסם ל-projector_node הודעה על פיה יש להקרין חץ על גבי הקרקע. לכן, obj_detector_node יפרסם על גבי ה topic, what_to_project/ את ההודעה arrow ואילו projector_node יאזין לכך ובהתאם המקרן ידע להקרין חץ.

כאשר הרחפן יגיע לנקודת ביניים, drone_dji_node יפרסם הודעת waiting. עם קבלת message זה לראשונה, obj_detector_node מפרסם ל- projector_node הודעת circle לפיה יש להקרין תמרור עצור אדום על גבי הקרקע. תמרור זה מסמן לאדם המבצע את המסלול שזוהי נקודת עצירה ועליו להגיע אל מיקום זה. במקביל, אלגוריתם עיבוד התמונה שנמצא ב object_detect.py יתבצע במטרה לזהות האם האדם הגיע לאותה נקודת ביניים. אלגוריתם זה תואר ברקע תיאורטי ובפרקים קודמים.

נציין כי בעת הפעלת האלגוריתם, obj_detector_node נעזר במידע המתפרסם על גבי camera/color/image_raw/. מידע זה מגיע ממצלמת Intel RealSense D435 וכולל בתוכו פריים אחר פריים המכילים מידע "בזמן אמת". כל עוד אין זיהוי, obj_detector_node יפרסם הודעת nothing שמשמעותה התייצבות באוויר והמשך הקרנת תמרור העצור האדום. כאשר יש זיהוי obj_detector_node מפרסם הודעת arrow. לאחר ש projector_node מבחין בכך המקרן מקרין חץ על גבי הקרקע. בנוסף, כאשר drone_dji_node מבחין בהודעה, הרחפן מתחיל בתעופה אל נקודת הביניים הבאה ומפרסם הודעת moving.

תהליך זה חוזר על עצמו עד לנקודת היעד של המסלול, במקרה זה drone_dji_node מפרסם הודעת end. Obj_detector_node מזהה הודעה זו ומפרסם הודעת trophy. Projector_node מזהה הודעה זו וכתוצאה מכך המקרן מקרין על גבי הקרקע גביע המעיד על סיום המסלול. כמו כן, גם drone_dji_node מזהה את הודעת ה trophy ותהליך נחיתת הרחפן מתחיל.

נרצה להדגיש מהי משמעות הודעת nothing. באופן כללי, פרסום הודעות publish וקבלת הודעות subscribe מבוצעים ללא הפסקה לכל אורך ריצת האלגוריתמים. לכן כדי שלא לפרסם את אותה הודעה מספר רב של פעמים יצרנו את האופציה להודעת nothing אשר מסמלת שאין צורך לבצע שינויים נוספים הן מבחינת ההקרנה והן מבחינת מהירויות התעופה.

כמו כן, נרצה להתייחס לכוונה בהגדרה של "זיהוי מוצלח" ושל "זיהוי שגוי". כאשר האדם עומד על סימן העצירה ומסתירו, אלגוריתם עיבוד התמונה יזהה את הסתרת סימן העצור כ-"זיהוי מוצלח". כאשר ההסתרה היא חלקית או שאין הסתרה כלל, האלגוריתם יזהה זאת כ-"זיהוי שגוי".

הטבלה הבאה מכילה סיכום כל ה topics בהם נעשה שימוש:

שם ה topic	סוג (type)	ערכים אפשריים
	חלק א'	
/rs_frame	PoseStamped	x,y,z coordinates: - position.x - position.y - position.z
/bebop/state_change	Bool	- true - false
/bebop/path_change	Bool	- true - false
/bebop/takeoff	Empty	
/bebop/land	Empty	
/bebop/cmd_vel	Twist	Linear and angular velocity in x,y,z coordinates - linear.x - linear.y - linear.z - angular.z
	חלק ב'	
/what_to_project	String	- Nothing - Circle - Arrow - trophy
/camera/color/image_raw	Image	- RGB image
/in_point	String	- Initialize - Moving - Waiting - end

טבלה 3. טבלה המסכמת את ה topics בהם נעשה שימוש בפרויקט.

5 ניתוח תוצאות

5.1. השוואות בין תוצאות הסימולציה לאלגוריתמים חליפיים

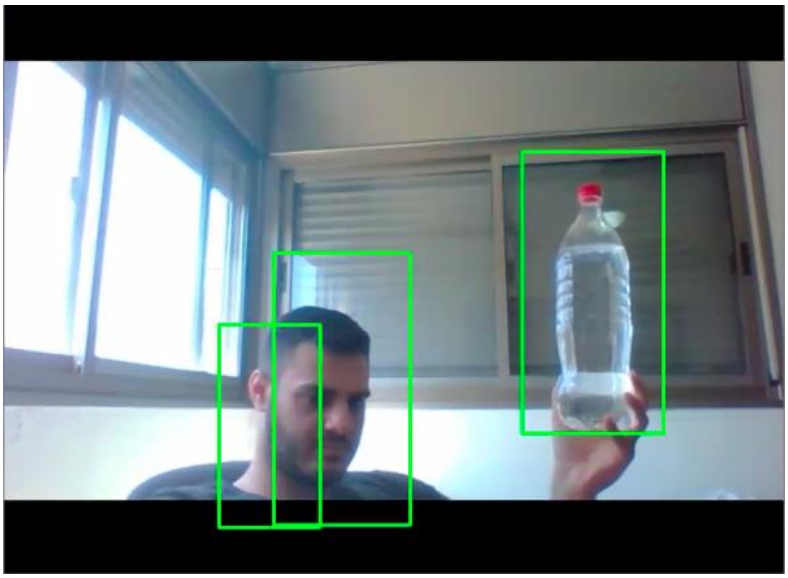
בפרק זה נשווה בין אלגוריתמי הפרויקט לבין אלגוריתמים אלטרנטיביים. ראשית נשווה בין אלגוריתם הבקרה הסופי לבין אלגוריתם בקרה ראשוני שעליו עבדנו. האלגוריתם הראשוני עליו עבדנו היה אלגוריתם בסיסי שקיבלנו ממנחה הפרויקט. לאחר התאמה ראשונית אל סוג הרחפן של פרויקט זה ואל ה topics המתאימים, בוצעו סימולציות בעבור האלגוריתם. הביצועים של האלגוריתם לא היו מספקים בעיקר בשל העובדה שתנועת הרחפן באוויר הייתה לא רציפה ואף מקוטעת במהלך התעופה.

נמחיש זאת בסרטון "ניווט בין נקודות".

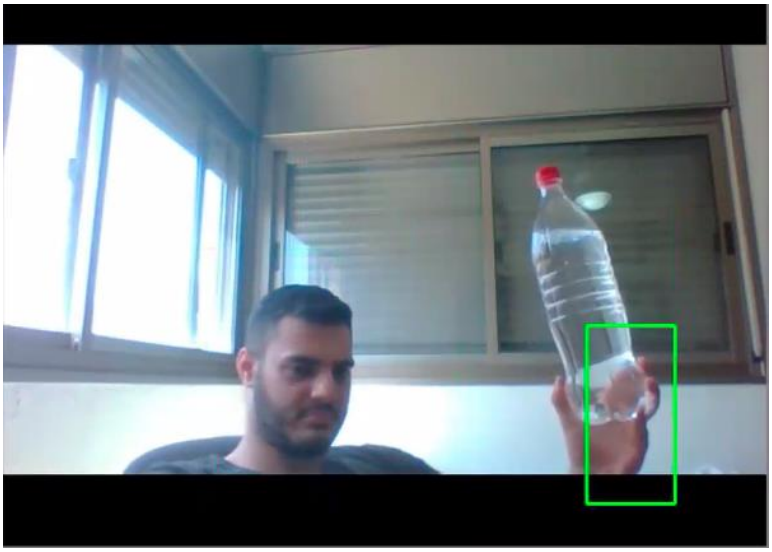
על מנת לשפר את הביצועים, כל מקטע המהווה תת מסלול בין שתי נקודות ביניים, חולק למספר רב של תתי מקטעים. בכל תת מקטע המהירות היא קבועה ולכן המהירות המתקבלת על פני מקטע שלם נראית חלקה הרבה יותר מאשר באלגוריתם הראשוני. תוצאות אלגוריתם משופר זה מוצגות בפרק הסימולציה.

נשווה כעת בין האלגוריתם הנבחר עבור עיבוד התמונה לבין אלגוריתם חליפי. כפי שנאמר ברקע תיאורטי, אלגוריתם נוסף שנלקח בחשבון הוא אלגוריתם המתבסס על מסנן מובנה מראש של ספריית OpenCV. לעומת זאת, האלגוריתם בו בחרנו להשתמש בפרויקט, מבוסס על זיהוי לפי מרחב הצבע. לצורך ההשוואה נבחן את ביצועי האלגוריתמים כאשר מטרתם היא לזהות בצורה מיטבית רק את האדם.

האלגוריתם הראשון שנציג הוא האלגוריתם החליפי. בתמונות הבאות מוצגות תוצאות הרצתו.



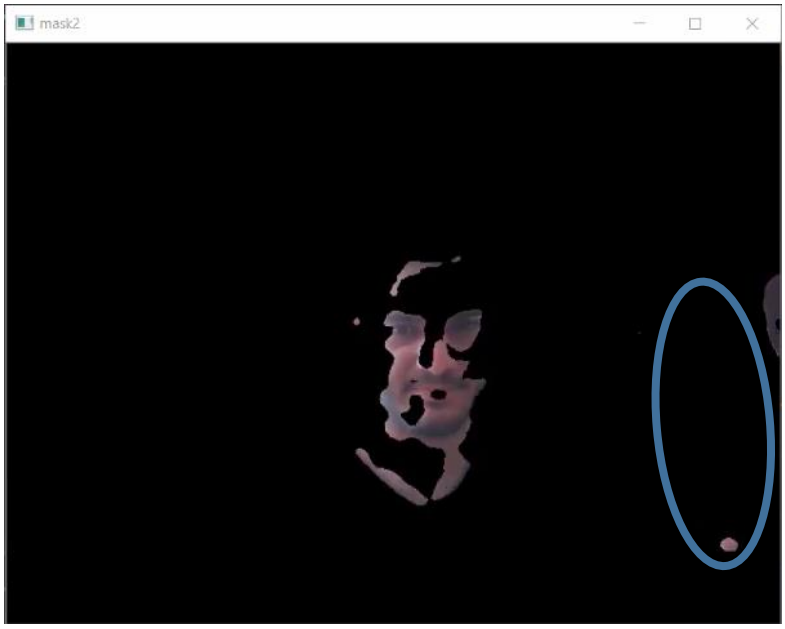
איור 16. תוצאה ראשונה של הרצת האלגוריתם החליפי לזיהוי אדם.



איור 17. תוצאה שנייה של הרצת האלגוריתם החליפי לזיהוי אדם.

בתמונות אלו המלבנים הירוקים מסמנים זיהוי של אובייקט. כפי שניתן לראות מהתמונה הראשונה, האלגוריתם אכן מוצא את האדם בצורה נכונה אך מסמן גם אובייקטים נוספים אשר מהווים זיהוי שגוי. למעשה, בתמונה השנייה, האלגוריתם לא מזהה כלל את האדם אלא רק אובייקט משני ולכן התקבל זיהוי שגוי לחלוטין.

נדגים בתמונה הבא את תוצאות הרצת האלגוריתם אשר נעשה בו שימוש בפרויקט



איור 18. תוצאת הרצת האלגוריתם המבוסס על מרחב הצבע לזיהוי אדם.

בתמונה זו האליפסה הכחולה מייצגת את מיקום הבקבוק כפי שהוחזק על ידי האדם. בקבוק זה הוא למעשה אובייקט מפריע שנשאף שהאלגוריתם לא יזהה. כפי שניתן לראות מהתמונה, האלגוריתם אכן הצליח לבצע זיהוי מוצלח של האדם. אמנם האלגוריתם כלל באופן מינימלי את היד המחזיקה את הבקבוק ולא כל פרצוף האדם נכלל בזיהוי, אך כפי שהוסבר ברקע תיאורטי, לאלגוריתם הוכנסו מנגנוני הגנה אשר מוודאים שגם במקרה של התמונה לעיל, יבוצע זיהוי נכון.

5.2. ביצועי המערכת מבחינת זמן אמת

נבחן את ביצועי המערכת מבחינת זמן אמת על פי הפרמטרים הבאים[3]:

- רמת השפעה מצד תנאי הסביבה שונים: הרחפן bebop, אשר עליו מתבצע אלגוריתם הבקרה, מסוגל לטוס הן בתוך מבנה והן בסביבת חוץ. כמו כן, כפי שהוצג בפרק הסימולציה, הרחפן שומר על יציבותו בעת התעופה באוויר גם לאחר הפרעות מצד גורמים חיצוניים.

- מהירות התעופה של הרחפן: מהירות התעופה ניתנת לשליטה על ידי המתכנת. ניתן להתרשם ממהירויות וזמן התגובה של הרחפן מהנתונים הבאים הלקוחים מהסרטונים המופיעים בפרק הסימולציה.
מתוך הסרטון "ריבוע" נובעים הדברים הבאים:
זמן ביצוע לתעופה לאורך צלע אחת באורך 2 מטר הוא 7 שניות (מהירות ממוצעת של $0.28\frac{m}{sec} \cong \frac{2}{7}\frac{m}{sec}$).
זמן ביצוע מסלול ריבועי במרחב הוא 41 שניות.

  מתוך הסרטון "תצוגה עם ריחוף והפרעות בנחיתה" נובעים הדברים הבאים:
  זמן המראה: 6 שניות.
  זמן נחיתה: 1.5 שניות.

- התייצבות באוויר: בפרויקט זה השאיפה היא שהרחפן יוכל להתייצב באוויר גם לאחר הפרעות מצד גורמים חיצוניים המשפיעים על תנועתו. ניתן לראות את ביצועי האלגוריתם מהבחינה הזו על פי סרטון "תצוגה עם ריחוף והפרעות בנחיתה". בסרטון זה ניתן לראות שזמן ההתייצבות הממוצע שבמהלכו הרחפן מסוגל לחזור לנקודת היעד הוא 1 שניות.

- זמן ביצוע לאיטרציה באלגוריתם עיבוד התמונה: זהו הזמן מרגע קבלת קלט לאלגוריתם ועד לפלט המורה האם יש זיהוי של אובייקט או שאין. הזמן הממוצע שהתקבל לאחר ביצוע של מספר סימולציות, הוא 1.1 שניות. כפי שמוסבר בפרק רקע תיאורטי, באלגוריתם לא כל פריים הנקלט על ידי מצלמת RealSense D435 מהווה קלט לעיבוד התמונה. הסיבה לכך הייתה רצון להקטין את העומס על האלגוריתם. לכן, מבחינה תאורטית ניתן היה להקטין את זמן הביצוע לאיטרציה (בקוד עצמו נבדק רק פריים אחד מכל 13 פריימים עוקבים, לכן ניתן להקטין זמן זה לכל היותר פי 13). אך הקטנה זו תבוא על חשבון הפעלת האלגוריתם גם על פריימים שלא תורמים מידע חדש.

- זמן התגובה לזיהוי: זהו הזמן מרגע הגעת האובייקט ועד לזיהוי נכון. משך זמן זה הוא עשירית השנייה.

- זמן התגובה לזיהוי מוצלח: זהו הזמן מרגע הגעת האובייקט ועד להחלפת סימן ההקרנה בהתאם לנקודה על פני המסלול בה נמצא הרחפן. משך זמן זה הוא 4 שניות.

- איכות התמונה המוקרנת: יש צורך בהתאמת הפרמטרים של הקוד כדי לעבור ממצבי תאורה שונים. במילים אחרות, איכות התמונה מושפעת מגורמים שונים כמו גובה הטיסה, זווית הצבת המקרן והמצלמה על הרחפן וכן רזולוציית התמונות המוקרנות.

[3] בחלק זה כל הזמנים המתוארים הם גדלים ממוצעים של כל המדידות שבוצעו.

6 סיכום, מסקנות והצעות להמשך

תחילה נשווה בין תוצרי הפרויקט שהושגו לבין המטרות שהוגדרו בתחילתו.

כפי שהוסבר בתחילת פרק המימוש, בפרויקט זה נתקלנו בבעיה שרחפן ה bebop, עליו התבססנו בפרויקט, לא היה מסוגל לשאת במשקל הנדרש במהלך תעופתו. לכן, בעצת המנחה הוחלט לעבור לשימוש ברחפן ה DJI. על רחפן זה לא היה ניתן לממש את אלגוריתם הבקרה שנוצר ולכן נכתב אלגוריתם המדמה את תעופת רחפן ה DJI. לאחר מכן, אלגוריתמי עיבוד תמונה והקרנה נכתבו ונוצרה התקשורת בינם לבין אלגוריתם הבקרה המדומה של רחפן ה DJI. לכן, בשל אילוצים אלו ובהמשך לסימונים בפרק המימוש, גם בפרק זה נפריד בין שני חלקי הפרויקט. חלק א׳ מתייחס לרחפן ה bebop וחלק ב׳ מתייחס לעבודה על רחפן ה DJI.

ראשית תוצג השוואה העוסקת בנושאי חלק א׳, כלומר, בתעופת הרחפן, בתהליך הניווט וכן באלגוריתם הבקרה באופן כללי. בחלק השני תוצג השוואה העוסקת במטרות שהוצבו הקשורות לחלק ב׳, כלומר, בעיקר בעיבוד התמונה ובפעולת ההקרנה.

נציין כי בשל המגבלות שצוינו והפיצול שנגרם מכך, לא הושג מימוש המשלב את תהליך התעופה עם אלגוריתמי עיבוד התמונה וההקרנה.

חלק א'

א. **יציבות הרחפן באוויר**

בחלקו הראשון של הפרויקט מטרה עיקרית והכרחית הייתה המראה ונחיתה של הרחפן בצורה בטיחותית ומבוקרת. כמו כן, על תעופת הרחפן להיות יציבה ולא מושפעת מהפרעות חיצוניות אשר עלולות להשפיע על התעופה. נטען כעת כי מטרה זו הושגה במלואה כפי שניתן לראות מהסרטון "תצוגה עם ריחוף והמראות ונחיתה".

בתחילת הסרטון הרחפן מבצע המראה בטוחה עד להגעה לגובה הטיסה. מעט לאחר מכן נשלחת פקודה המשנה את בקרת הרחפן מבקרה מובנית פנימית לבקרה אשר פועלת בעזרת אלגוריתם הבקרה וטכניקת SLAM. במקביל לכך, הרחפן ממשיך במעופו סביב הנקודה המוגדרת.

בהמשך הסרטון ניתן לראות את מנחה הפרויקט מפעיל כוח חיצוני המפריע לתעופתו של הרחפן. במהלך ההפרעה ניתן לשמוע את מנועי הרחפן אשר מגבירים עוצמה ומתנגדים להפרעה החיצונית. לאחר פרק זמן קצר הרחפן חוזר למקומו המקורי, בכך ניתן לראות את הגשמת המטרה שהוצבה בכל הנוגע להגנה מפני הפרעות חיצונית. לבסוף ניתנת לרחפן פקודת נחיתה. ניתן לראות שהנחיתה מבוצעת בצורה בטוחה ואיטית על גבי הקרקע. לסיכום, חלק זה מומש בהצלחה בהתאם למטרה.

ב. **יכולות ניווט**

נבחן את המטרה לפיה על הרחפן לבצע תעופה בין מספר נקודות בצורה חלקה ורציפה ובמגוון מסלולים שונים. נטען כי מטרה זו מומשה בהצלחה ונמחיש זאת בעזרת הסרטון "ריבוע" שהוצגו בפרק הסימולציה. בסרטון זה מבצע הרחפן טיסה במסלול ריבועי לאורך צלע שאורכה 2 מטרים. ניתן לראות בסרטון כי התעופה אינה מקוטעת בין נקודות עצירה שונות לאורך המסלול וניתן להחשיבה כ-"חלקה ורציפה". בנוסף, בכל נקודת עצירה ניתן לראות שהרחפן מתייצב באוויר. המעבר לנקודת הביניים הבאה מתבצעת רק לאחר התייצבות מוחלטת ובמהירות אפסית. מטרת כל נקודת עצירה כזו הינה התחלת הפעלת האלגוריתם לעיבוד תמונה לצורך זיהוי האדם. נתייחס למטרה זו בהמשך כאשר נדבר על חלק ב׳.

בנוסף, נציין כי מהירות הרחפן ניתנת לשליטה על ידי המתכנת לפי פרמטר אשר נקבע מראש, ומאפשר מנעד רחב של מהירויות. כמו כן, מסלול הטיסה אינו מוגבל להיות ריבועי ולמעשה ניתן לבחור כל מסלול המוגדר מראש ובהינתן נקודות עצירה שונות. התאמת הקוד למסלול אחר היא פשוטה ולא משפיעה על יציבות הרחפן באוויר או על מהירותו. לאור כל זאת ניתן לומר שחלק זה מומש בהצלחה ובהתאם למטרה.

חלק ב'

ג. **ניווט בשילוב הקרנה**

בחלקו השני של הפרויקט מומש מנגנון התקשורת בין הרחפן לבין המקרן המוצב עליו. מטרת חלק זה הייתה ליצור תיאום בין אלגוריתם ההקרנה לאלגוריתם עיבוד התמונה וכך לאפשר הקרנת סימונים שונים על גבי הקרקע. כפי שהוסבר בפרק תיאור תוכנה התיאום עצמו נוצר על ידי שימוש ב topics שנוצרו על ידנו ושליחת messages בהתאם. נמחיש שמטרה זו מומשה בהצלחה על ידי הסרטון "עיבוד תמונה בשילוב הקרנה" המופיע בפרק סימולציה. בסרטון זה נראה שבעת הימצאות הרחפן בשלב התעופה, כלומר, בין נקודות

עצירה, מוקרן חץ ירוק על גבי הקרקע אשר מסמל את כיוון הניווט. בעת הגעה לנקודת עצירה, המקרן מחליף את התמונה המוקרנת לסימן עצירה אדום, סימן זה מסמל את שלב עצירת הרחפן עד להגעת האדם לנקודה המסומנת על גבי הקרקע. עם סיום מסלול הניווט והגעת הרחפן לנקודת היעד, הרחפן מקרין גביע על הקרקע, סימן המסמל את סיום המסלול.
מהדגמה זו ניתן לראות שהמטרה מומשה בשלמותה אך באופן בסיסי שכן סט הסימנים המוקרנים היה דל. יחד עם זאת, תוספת של סימני הקרנה נוספים על גבי הקרקע היא פשוטה ולא משפיעה על בקרת הרחפן, על הביצועים של האלגוריתם לזיהוי האדם או על תהליך ההקרנה.

ד. **עיבוד תמונה**

חלק זה מתייחס לביצועים של האלגוריתם לזיהוי נוכחות האדם בנקודות עצירה. על פי הסרטון שהוצג בסעיף הקודם, חלק זה בוצע בהצלחה אך בצורה בסיסית מהסיבות הבאות. ראשית, זמן התגובה מרגע הגעת האדם לנקודות העצירה ועד להמשך התעופה הוא מסדר גודל של 4 שניות. השאיפה היא להקטין זמן זה ככל הניתן ועדיין לשמור על אותה רמת ביצועים. לכן, אלגוריתם משופר יותר יאפשר הקטנה של זמן תגובה זה. שנית, אלגוריתם זה מבוסס על זיהוי על פי מרחב הצבע. מימוש זה מושפע מגווני תאורה שונים ומגוון סימן העצירה, שבמקרה של ההדגמה הוא מוצג בגווני אדום. לכן, ניתן לומר שמימוש מדויק ויעיל יותר יהיה מבוסס על למידה עמוקה וכך לא יושפע מבעיות אלו.

ה. **הימנעות ממכשולים**

לבסוף, המטרה על פיה הרחפן יהיה מסוגל להתחמק ממכשולים לא מומשה בפרויקט זה עקב קוצר בזמן.

לסיכום, נראה שעמדנו במרבית המטרות שהוצבו. יחד עם זאת, ניתן היה לשפר את ביצועי אלגוריתם הזיהוי ולקבל מימוש יעיל יותר וכן להוסיף אלגוריתם שמטרתו לזהות מכשולים ולהימנע מהם בהתאם. כמו כן, בשל האילוצים לעבודה עם שני רחפנים, לא ניתן היה להציג תוצאה סופית הכוללת תעופה בשילוב הקרנה ועיבוד תמונה. למרות זאת, בוצעו ההכנות הנדרשות לשילוב כל האלגוריתמים על רחפן ה DJI (כפי שצוין, לא ניתן לעבוד על רחפן ה bebop).

<u>נתאר כעת הצעות שונות לשיפור ביצועי המערכת:</u>

א. על מנת להביא להשלמתו המלאה של פרויקט זה ניתן להתאים את אלגוריתם הבקרה המוצג בחלק א׳ של הפרויקט כך שיתאים לרחפן ה DJI המתואר בחלק ב׳. ניתן לבצע זאת על ידי יצירת קובץ בשפת python הממשש את כל הפעולות והפונקציות המוצגות בקובץ בשפת c++ של חלק א׳. בנוסף, יש לבצע התאמה גם מבחינת ה topics המופיעים בחלק א׳ כך שיותאמו להגדרת ה topics בחלק ב׳. במילים אחרות, יש לאחד את קובץ הבקרה מחלק א׳ הכתוב בשפת c++ עם הקובץ המדמה את קובץ בקרה זה, אשר כתוב בשפת python. לדעתנו יש לעבוד בדרך פעולה זו מכיוון שעבודה במקביל עם קובץ בשפת c++ וקבצים בשפת python עלולה להיות מסובכת יותר.

ב. על ידי התקנת רכיב זיכרון גדול יותר על רחפן ה DJI ניתן לשכלל את אלגוריתם עיבוד התמונה ולהשתמש באלגוריתמים המבוססים על למידת מכונה. במידה ומגבלה זו איננה פתירה ניתן להשתמש באלגוריתמים מורכבים יותר המבצעים עיבוד וידאו (שימוש במספר פריימים מן המצלמה) במקום באלגוריתם המבצע עיבוד תמונה (פיענוח פריים בודד בכל איטרציה).

ג. בפרויקט זה על המתכנת להריץ בחלונית ה terminal מספר שורות הרצה כדי שהרחפן ימריא ויחל תהליך הניווט. ניתן לשפר את הפרויקט על ידי מתן אופציה להרצת פקודות אלו בצורה פשוטה ונוחה למימוש, כך שגם אדם ללא ידע רב בתחום ידע להפעיל את המערכת. אופציה למימוש זה הינה בניית קובץ הרצה אשר מפעיל כל קובץ לפי הסדר הנדרש לתפעול הקבצים. בקובץ זה ניתן לכלול את פקודות ההמראה וכן את פקודות העברת הבקרה מבקרה פנימית של הרחפן אל בקרה הנשלטת באלגוריתם אשר בקובץ התעופה. בעזרת ביצוע פעולות אלו נוכל ליצור ממשק נוח יותר למשתמש.

נתאר כעת אפשרויות להמשך הפעילות המחקרית בתחום הרחפנים בשילוב הקרנה.

חסרון עיקרי בפרויקט זה הוא שקביעת מסלולי הניווט נקבעת על ידי מיקומן המרחבי של נקודות העצירה השונות ולא על ידי נקודת יעד בלבד, כפי שמתבצע כיום באפליקציות ניווט שונות. כלומר, בהינתן נקודות יעד, נרצה שקובץ הבקרה והתעופה יכלול אלגוריתם למציאת המסלול הטוב ביותר המסתיים בנקודת היעד, תוך מציאת נקודות העצירה השונות המתאימות למסלול זה. כמו כן, נרצה שבחירת המסלול תהיה מושפעת ממכשולים שונים לאורך הדרך כמו קירות בתוך מבנה, עצים ועוד. כמוסבר בפרק ההקדמה, אפליקציות ניווט המבצעות זאת מתבססות לרוב על טכנולוגית ה GPS ואילו בפרויקט זה אופציה זו לא רלוונטית. לכן, לשם כתיבת אלגוריתם המבצע משימה זו יש לבצע מחקר נוסף בבניית מסלול טיסה אל עבר נקודת יעד תוך התחשבות במכשולים לאורך הדרך, זאת בעזרת טכנולוגית ה SLAM. נציין כי זוהי משימה מורכבת שכן לטכניקת הפתרון בעזרת SLAM יש מגבלה ביחס לטכנולוגית GPS. באופן כללי, המימוש לבעיה החישובית SLAM כפי שמתבצע בעזרת מצלמת Intel RealSense מתבסס על התמונות הנדגמות מהמצלמה. כתוצאה מכך, כאשר המצלמה מוצבת מול מכשול או קיר ומכיוון ששדה הראיה של המצלמה הוא מוגבל, הנתונים המגיעים למצלמה לא יספקו מידע רלוונטי לצורך מציאת המסלול הטוב ביותר. רק לאחר הימצאות הרחפן בדרך עוקפת של המכשול ניתן לקבל אינפורמציה רלוונטית מהמצלמות שתשמש לבחירת המסלול. לכן, פתרון למשימה זו צריך לאפשר את עדכון המסלול הטוב ביותר בעת תעופת הרחפן ולא רק טרם תהליך ההמראה. כתוצאה מכך המשימה הופכת למורכבת יותר ולכן מצריכה מחקר מעמיק יותר, ייתכן בעזרת טכנולוגיה שונה מטכניקת SLAM.

7 נספחים

פרק הנספחים מחולק לפי הנושאים הבאים:

א. קישור לסרטוני הסימולציה לפרויקט המופיעים בדרייב

ב. סימני הקרנה בהם נעשה שימוש לצורך ניווט

ג. הוראות התקנה שיצרנו לצורך הורדת ROS עבור Ubuntu 16 ויצירת סביבת עבודה חדשה.

ד. הוראות התקנה שיצרנו לצורך התקנת חבילת RealSense של אינטל

ה. הוראות התקנה שיצרנו לצורך התקנת bebop drivers

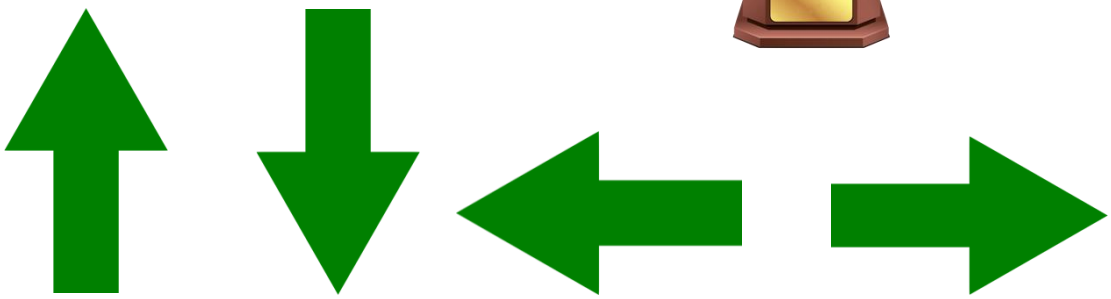
ו. הדגמת קובץ launch

ז. סדר הרצת הקבצים בחלק ב של הפרויקט.

א. קישור לסרטוני הסימולציה לפרויקט המופיעים בדרייב.

סרטון "ניווט בין נקודות"
סרטון "ניווט בין נקודות2"
סרטון "עיבוד תמונה בשילוב עם הקרנה"
סרטון "עיבוד תמונה"
סרטון "ריבוע"
סרטון "ריבוע עם בעיה"
סרטון "ריחוף באוויר 2"
סרטון "ריחוף באוויר עם בעיה"
סרטון "תצוגה עם ריחוף והפרעות"
סרטון "תצוגה עם ריחוף והפרעות ונחיתה".

ב. סימני הקרנה בהם נעשה שימוש לצורך ניווט.



ג. הוראות התקנה שיצרנו לצורך הורדת ROS עבור Ubuntu 16 ויצירת סביבת עבודה חדשה.

1. שלב 1: התקנת ubunto לפי הקישור הבא:

https://www.youtube.com/watch?v=ehtUb55Rmmg&list=PLk51HrKSBQ8-jTgD0qgRp1vmQeVSJ5SQC

חשוב מאוד לבחור סיסמה קלה ולשמור אותה. סיסמה זו משמשת בכל התקנה נוספת ובכל פעם שמתחילים לעבוד על ה supercomputer. עדיף סיסמה קלה ללא מספרים.

2. התקנת ROS חלק ראשון לפי הקישור הבא:

http://wiki.ros.org/kinetic/Installation/Ubuntu

השלבים שצריך לבצע:

- 1.2
- 1.3
- 1.4 את שורות הקוד הבאה:

```
sudo apt-get update
sudo apt-get install ros-kinetic-desktop-full
```

- 1.5
- 1.6 רק את שורות הקוד הראשונות:

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

- 1.7

3. שלב 3: התקנת ROS חלק שני:

כאן מתקינים את החלק של ה ROS build על פי הקישור הבא:
https://catkin-tools.readthedocs.io/en/latest/installing.html#installing-on-ubuntu-with-apt-get

רק את ארבע השורות הראשונות:

Installing on Ubuntu with apt-get

First you must have the ROS repositories which contain the `.deb` for `catkin_tools`:

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu `lsb_release -sc` main" >
/etc/apt/sources.list.d/ros-latest.list'
$ wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
```

Once you have added that repository, run these commands to install `catkin_tools`:

```
$ sudo apt-get update
$ sudo apt-get install python-catkin-tools
```

4. שלב 4: בדיקה שהכל תקין עד לנקודה הזו. יש לפעול על פי הקישור הבא:

http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

רק השורה הראשונה מחלק (2):

```
$ printenv | grep ROS
```

5. שלב 5 – יצירת סביבת עבודה workspace: חשוב לבחור בשם משמעותי. למשל catkin_ws.

ליצור workspace על פי הקישור הבא:
http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/
$ catkin build
$ source devel/setup.bash
```

חשוב! יש לעדכן את קובץ ה bashrc באופן הבא. בחלון ה terminal יש לרשום:

```
$ cd catkin_ws/
~/catkin_ws$ sudo nano ~/.bashrc
```

כאשר bashrc זהו הקובץ הרצה אשר מופעל כל פעם שפותחים את הterminal. יש להוסיף לקובץ זה את שורת הקוד הבאה:

```
$ source ~/catkin_ws/devel/setup.bash
```

6. שלב 6: יש לפעול על פי הקישור הבא:

https://www.youtube.com/watch?v=IcCUyybZmAE

ד. הוראות התקנה שיצרנו לצורך התקנת חבילת RealSense של אינטל.

התקנת RealSense wrapper. על מנת ליצור את ה dependencies יש לרשום את שורות הקוד הבאות:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu `lsb_release -sc` main" > /etc/apt/sources.list.d/ros-latest.list'
wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
sudo apt-get update -qq
sudo apt-get install ros-kinetic-cv-bridge -y
sudo apt-get install ros-kinetic-image-transport
sudo apt-get install ros-kinetic-tf -y
sudo apt-get install ros-kinetic-diagnostic-updater -y
sudo apt-get install ros-kinetic-ddynamic-reconfigure -y
source ~/.bashrc
```

ה. הוראות התקנה שיצרנו לצורך התקנת bebop drivers.

על פי הוראות אלו יכולנו להתקין על ה supercomputer raspberry pi את חבילת bebop autonomy אשר שימשו אותנו לצורת הטסת הרחפן בחלק ב׳ של הפרויקט.

```
sudo apt-get install build-essential python-rosdep python-catkin-tools
cd ~/catkin_ws
catkin init
git clone https://github.com/AutonomyLab/bebop_autonomy.git src/bebop_autonomy
rosdep update
rosdep install --from-paths src -i
catkin build -j1
```

לאחר מכן ניתן להריץ ולוודא שההתקנה הצליחה על ידי שורות הפקודה הבאות

```
source ~/.bashrc
roslaunch rs_pub publish_rs.launch
```

ו. הדגמת קובץ launch.

נדגים את תוכן קובץ ה launch הנקרא obj_detect_launch.launch.
באופן כללי, כל קובץ launch יכול לאפשר את הרצתם של כמה nodes במקביל אך במקרה של הפרויקט שלנו בחרנו ליצור שלושה קבצי launch, אחד עבור כל node. תוכן הקובץ:

```
<launch>
<include file="$(find realsense2_camera)/launch/rs_camera.launch"/>
<node pkg="detection_pkg" name="obj_detector_node" type="object_detect.py" output="screen">
</node>
</launch>
```

קובץ launch המוצג מתייחס ל node הנקרא "obj_detector_node". כפי שהוסבר בפרק תיאור תוכנה node זה אחראי על הרצת האלגוריתם לעיבוד התמונה. כל קובץ launch מתחיל בשורה של

```
<launch>
```

ונגמר בשורה של

```
</launch>
```

"הלב" של קובץ ה launch המכיל את המידע הרלוונטי על node זה נמצא בשורה הבאה:

```
<node pkg="detection_pkg" name="obj_detector_node" type="object_detect.py" output="screen">
```

ב "pkg" הכוונה לשם ה package המכילה את הקוד של קובץ ההרצה (של ROS הכתוב בשפת python או c++). ב- "type" הכוונה לשם קובץ ההרצה. ב "name" הכוונה לשם ה- node. שם זה מוגדר לראשונה בקובץ ההרצה. ב "output" הכוונה לסוג הערוץ שלתוכו ייפלט מוצא התכנית.

נתייחס כעת לשורה הבאה:

```
<include file="$(find realsense2_camera)/launch/rs_camera.launch"/>
```

מטרת שורה זו הינה להכיל בקובץ launch זה את תוכן קובץ ה launch הנקרא "rs_camera.launch". על ידי פעולה זו ניתן להריץ את כל ה nodes הרלוונטיים מקובץ ה launch של "rs_camera". Nodes אלו חיוניים לצורך התיאום והתקשורת בין אלגוריתם עיבוד התמונה לבין פעולת מצלמת ה- realSense. למעשה, ללא שורה זו יש צורך להריץ בצורה ידנית את קובץ ה launch של המצלמה. אם לא נעשה זאת, התמונות הנדגמות במצלמה לא יוכלו להגיע לאלגוריתם לעיבוד תמונה והאלגוריתם לא ירוץ.

ז. סדר הרצת הקבצים בחלק ב של הפרויקט.

לצורך הרצת חלק ב' של הפרוייקט יש צורך להריץ 3 קבצי launch בסדר מסוים, מומלץ לפתוח שלושה חלונות terminals שונים.
ראשית יש להריץ את קובץ ה launch- projector_launch.launch.
לאחר מכן להמתין 2 שניות ולהריץ את קובץ ה drone_dji_launch.launch.
רק לאחר הופעת השורה " " ב terminal השני יש להריץ את קובץ ה obj_detect_launch.launch. מומלץ לבצע שורה זו ב-5 השניות הקרובות.
*במידה ומופיע error בהקשר למצלמת ה RealSense D435 יש לנתק ולחבר את המצלמה ולחזור על השלבים מההתחלה.

8 מקורות

דף נתונים של רכיב:

[1] "Raspberry Pi 4 Tech Specs", https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

[2] "Parrot bebop 2", https://www.parrot.com/en/

[3] "Intel® RealSense™ Tracking Camera T265, Intel® RealSense™ Tracking Module T261", September 2019 https://www.intelrealsense.com/wp-content/uploads/2019/09/Intel_RealSense_Tracking_Camera_Datasheet_Rev004_release.pdf

[4] "Dji, MANIFOLD 2, SPECS - Manifold 2-G (128GB) General", https://www.dji.com/manifold-2/specs

[5] Dji, N3, "New flight control algorithm, dual-IMUs and vibration damping system.", https://www.dji.com/n3/info#specs

[6] "NVIDIA. AUTONOMOUS MACHINES, Jetson TX2 Module", https://developer.nvidia.com/embedded/jetson-tx2

[7] "Intel RealSense Depth Camera D435", Datasheet, June 2020, https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf

[8] "Laser beam pro," http://www.laserbeampro.com/

User's Guide:

[9] "Laser beam pro", IEC 60825-1 Edition 3: 2014, User Guide, http://laserbeampro.com/download/C200_User_Manual.pdf

קישורים למקורות באינטרנט:

[10] "Real-time Human Detection with OpenCV", https://thedatafrog.com/en/articles/human-detection-video/

[11] "What Is Simultaneous Localization and Mapping?", https://blogs.nvidia.com/blog/2019/07/25/what-is-simultaneous-localization-and-mapping-nvidia-jetson-isaac-sdk/

[12] "bebop_autonomy - ROS Driver for Parrot Bebop Drone (quadrocopter) 1.0 & 2.0" https://bebop-autonomy.readthedocs.io/en/latest/#bebop-autonomy-ros-driver-for-parrot-bebop-drone-quadrocopter-1-0-2-0

[13] "Intel RealSense Tracking Camera T265", https://www.intelrealsense.com/visual-inertial-tracking-case-study/

[14] "HOW TO... What frame rate to use for shooting video" https://camerajabber.com/what-frame-rate-to-use-for-shooting-video/

[15] "All About Multirotor Drone FPV Cameras", https://www.getfpv.com/learn/new-to-fpv/all-about-multirotor-drone-fpv-camera/

[16] "HOW TO CHOOSE FPV CAMERA FOR QUADCOPTERS AND DRONES", https://oscarliang.com/best-fpv-camera-quadcopter/#fov