- האוניברסיטה הפתוחה –

סמינר במערכות מידע (20373)

רובוטים עצמאיים בשדה הקרב: המעבר מלוחמה אנושית ללחימה רובוטית

מנחה: ד"ר מיה הרמן

מגישה: הינדה יעל פרומר

בס"ד

תוכן העניינים

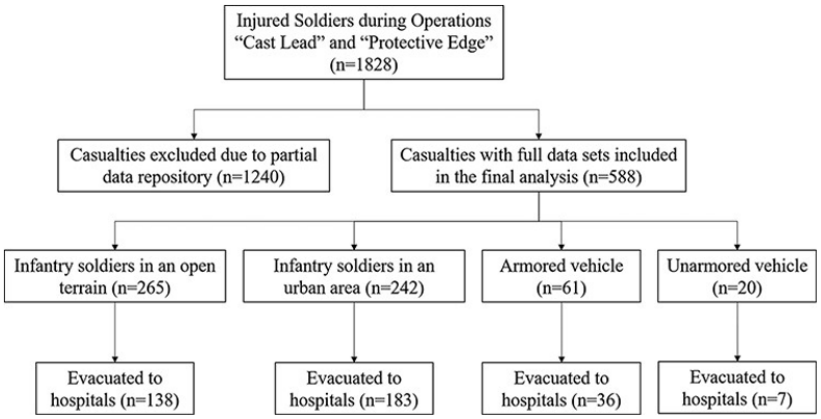
מבוא

הטרור שחווה מדינתנו הקטנה בחודשים האחרונים הביא רבים לחשב מסלול מחדש בכלל תחומי החיים. זו לא הפעם הראשונה בהיסטוריית המדינה שהיא מותקפת באכזריות ובעוצמות שלא תואמות את השטח הפעוט עליו היא פרושה. הטרגדיות הרודפות מוכרחות לגרום לנו להסתכל אחורה ולחשוב פעם נוספת על מה שנקרא ההשתדלות שאנו מחויבים בה כדי לשמור על חיי אדם. על חיי חיילינו ואולי אם אפשר לומר על שפיות מלחמתית גבוהה יותר. מוכרחים לשקול שוב את צורת הלחימה.

במאמר Combat Injury Profile in Urban Warfare מתוארות שתי מלחמות שהתרחשו בעשור האחרון - מלחמת צוק איתן ומלחמת עמוד ענן:

23 ימי חושך אש וקרבות בעופרת יצוקה ו-50 של כאב, דם ומוות בצוק איתן. לוחמים יוצאים לקרב עטויים ציוד מגן מכף רגל ועד ראש, חלקם בטנקים וחלקם ברכבים משוריינים ומהם מהלכים רגלית. בסופו של יום – כשנחתמו הפסקות האש נספרו החיילים ההרוגים והפצועים וכמו במלחמות שקדמו לאלו לא נפתרה מדינת ישראל מהכמות הכואבת של האבדות. כואב לחשוב שמאחורי המספרים עומדים יחידים שנקטפו בחטף ומשפחות משפחות שעוד יוסיפו להתגעגע לעולם שהיה ואיננו, לחיים היפים שנגוזו.

הנתונים המצביעים על כך שחיילים הנמצאים פחות בשדה הקרב הפיזי נפגעים פחות, קוראים לשכלול דרכי הלחימה כך שחיילים ימצאו פחות ופחות פיזית בשדה הקרב, עם שאיפה לצמצום מוחלט של לחימה פנים אל פנים. בדיאגרמה שלהלן מתוארים מספר המפונים לבתי החולים – הפצועים בעבור כל סוג הגנה – בעצם הרחקה משדה הקרב שעטו הלוחמים – המאמר מציין כי מבין 1828 החיילים ישנם חיילים שלא ידוע הסיבה המדויקת בגינה נפגעו ומהיתר נערך הניתוח להלן:



עפ"י הטבלה האחוז הגבוה ביותר של הלוחמים שנפגעו – יותר מ-75 אחוז מקבוצת הלוחמים בשטחים עירוניים. רוב הפצועים הם חיילים שנאלצו להילחם בעומק שדה הקרב, קרוב מאוד למחבלי החמאס וחיילי צבא האויב פונו לבתי חולים ורבים מהם לא יכלו לשוב ולתפקד כרגיל ימים רבים.

ההבנה שדרכי הלחימה הנוכחיות אינן יכולות להמשיך כמות שהן, מובילה למסקנה שיש צורך להשתמש ברובוטיקה ובינה מלאכותית כדי להגן על חיי חיילים ולשפר את יעילות ובטיחות הלחימה. רובוטים חכמים יכולים להחליף חיילים בשדה הקרב, לפעול במקומות מסוכנים ולהקטין את הצורך בכוח אדם במצבי סיכון. רובוטים אלו אינם מושפעים מרגשות כמו פחד או כאב, מה שמאפשר להם לפעול באופן מדויק ומכוון מטרה.

מטרת הסמינר על חמשת פרקיו היא לנתח צורת לחימה חדשה שמתפשטת בעולם. בסמינר נצטרף לעשרות שכבר טוענים שעל החברה המודרנית להחליף חייל בדמות חסרת חיות מברזל, רובה בזרוע מתקדמת בעלת ארבעה או חמישה מפרקים סיבוביים ולתת לרובוטיקה את המקום הראוי לה בשדה הקרב כדי להגן על חיי אנשים ולקדם יעילות ובטיחות. נכון הדבר שבמדינות מתקדמות כבר החלו להיעזר בכלי הרובוטיקה והבינה המלאכותית בתעשייה הצבאית אך בצורה מתונה. עדיין נשלחים חיילים צעירים המחרפים נפשם להילחם פנים אל פנים עם מחבל טרוריסט או חייל צבא אויב צמא דם. כמו בצבא האמריקאי שמשלב טכנולוגיות רובוטיות ומערכות אוטונומיות במבצעים שלו במידות שונות. זה כולל כלי טיס לא מאוישים למשימות סיור ולחימה, מערכות רובוטיות לפינוי מטענים ולסיורי היכרות, ורכבים אוטונומיים לתפקידי לוגיסטיקה ותמיכה.

בסמינר הזה אדבר על הנושא מכיוון חזק הרבה יותר. פה ארצה לנתח ולשאוף להחזיק ביכולות אחרות של הבינה המלאכותית והרובוטיקה. נדון ברובוטי הניווט האוטונומיים בתעשייה המלחמתית במטרה לנצל את הכוחות האמיתיים הגדולים וההכרחיים של הרובוטיקה – ובפשטות:

- טכנולוגיות אוטונומיות יכולות לבצע משימות מסוכנות ולהפחית את הצורך בכוח אדם במצבי סיכון, ובכך למזער פגיעה בלתי רצויה ולשמר את מוסר הלחימה.

- טכנולוגיות אוטונומיות מאפשרות פעולות לחימה מתקדמות ומדויקות, תוך הפעלת כוח ממוקד וחכם להשגת יעדים קרביים במינימום זמן ומשאבים. הן מספקות יכולת לבצע משימות קרביות בדיוק גבוה, מה שמביא להכרעות מהירות ויעילות בשטח הקרב, מבלי להסתמך על חישובים או הערכות אנושיות שעלולות להתארך ולהסתבך.

כפי שכבר צוין, לסמינר חמישה פרקים כאשר כל פרק מקדם את הקורא בעוד צעד לקראת מטרת הסמינר, החל מהצורך הגדול בהחלפת שיטת הלחימה הנהוגה היום ללחימה באמצעות רובוטי ניווט אוטונומיים עובר דרך סריקה מעמיקה על המושג "ניווט אוטונומי" – למה משמשים רובוטי הניווט האוטונומיים בעולם הרחב – תחבורה, רפואה, חקלאות, תעשיה, בית חכם וניקיון, מחקר ועוד. ממשיך דרך סריקה נרחבת של הלחימה הרובוטית והטכנולוגיות האוטונומיות שייקחו חלק בהתקדמות לקראת הצעד המשמעותי של רובוטיקת לחימה אוטונומית. ומתמקד בניתוח מעמיק של שלושת האלגוריתמים המרכזיים: SLAM, RRT וקינמטיקה הפוכה. הסמינר שואף לסקור את כל הנושא של רובוטיקת לחימה אוטונומית, אך היקפו הרחב מחייב התמקדות בתת-נושא ספציפי – הניווט האוטונומי בלוחמה רובוטית. בלב הסמינר עומד תהליך הניווט האוטונומי המורכב מהשלבים הבאים:

- **קבלת נתונים (מערכת התפיסה):** חיישנים אוספים נתונים מהסביבה.
- **מיפוי ומיקום (SLAM):** שימוש באלגוריתמים כמו EKF-SLAM, Graph SLAM ומונטה קארלו למיפוי הסביבה וקביעת מיקום הרובוט בתוכה.
- **מציאת מסלול (RRT):** מהנקודה שבה נמצא הרובוט ומהמפה שנוצרה בשלב הקודם, האלגוריתם RRT מתכנן את המסלול האופטימלי ליעד.
- **הקינמטיקה ההפוכה:** המרת המסלול המתוכנן לצעדים פיזיים.
- **אותות חשמליים:** הפיכת התוכנית לפעולות באמצעות אותות חשמליים המניעים את המפרקים.

כך מושלם התהליך ומאפשר לרובוט לפעול באופן עצמאי ומדויק בשדה הקרב.

רובוטי ניווט אוטונומי

הקדמה:

מבוסס על המאמר A one decade survey of autonomous mobile robot systems של Noor Abdul, K. Z., & Al-Araji, A.

בפרק זה נעסוק בעולם הרובוטיקה האוטונומית, תוך סקירת הטכנולוגיות המאפשרות לרובוטים לפעול באופן עצמאי ומתקדם. נתחיל בהגדרת מושג הרובוטיקה האוטונומית ונמשיך לפרט על מערכות התפיסה המאפשרות לרובוט לקלוט מידע מסביבתו ולעבדו לכדי נתונים בעלי משמעות.

מהי רובוטיקה אוטונומית?

רובוט הוא אוטונומי כאשר אין בו או כמעט אין התערבות אנושית, והוא מסוגל להחליט באופן עצמאי על הפעולות שעליו לעשות לצורך מימוש המטרות ועמידה במשימות. האוטונומיה מתאפשרת הודות לבינה המלאכותית, המאפשרת דמיון להתנהגות אנושית ומאפשרת לרובוט לפעול באופן עצמאי ומדויק.

מערכות התפיסה

מערכות התפיסה של הרובוט כוללות חיישנים כמו LIDAR, מצלמות, IMU ועוד, המספקים מידע חיוני לתהליך הניווט. מערכות אלו מאפשרות לרובוט לקלוט מידע מסביבתו ולעבדו לכדי נתונים בעלי משמעות, שמכינים את הבסיס לשלב הבא.

מיפוי ומיקום עצמי (SLAM)

שלב מרכזי במימוש האוטונומיה הוא יכולת המיפוי והמיקום העצמי של הרובוט. זה מתבצע על ידי אלגוריתם SLAM (Simultaneous Localization and Mapping), המאפשר לרובוט לסרוק את הסביבה ולבנות מפה מדויקת בזמן אמת, תוך כדי שהוא מזהה את מיקומו בתוך המפה. שלב זה קריטי בגינו הרובוט יודע "איפה הוא נמצא" בכל רגע נתון.

תכנון מסלול (RRT)

לאחר שהרובוט מיפה את הסביבה ומיקם את עצמו בתוכה, השלב הבא הוא תכנון המסלול. כאן נכנס לתמונה אלגוריתם RRT (Rapidly-exploring Random Tree), שמשמש לתכנון מסלול בסביבה המורכבת שהרובוט כעת מכיר. RRT פועל באופן יעיל למציאת מסלול בטוח ויעיל מנקודת המיקום הנוכחית ליעד המבוקש.

קינמטיקה הפוכה

עם המסלול המתוכנן, תת-מערכת הקינמטיקה של הרובוט נדרשת לבצע את המסלול. הקינמטיקה ההפוכה מאפשרת לרובוט לחשב את הזוויות הנדרשות לכל מפרק במערכת הרובוטית כדי להגיע מנקודה אחת לאחרת על פי המסלול המתוכנן. כלומר, היא ממירה את המסלול התיאורטי לרצף פעולות פיזית שמבוצעות על ידי הרובוט.

מערכת הבקרה

מערכת הבקרה היא הלב הפועם של הרובוט האוטונומי. היא אחראית על תיאום כל תת-המערכות שלו כדי להבטיח פעולה חלקה ויעילה. המערכת מקבלת נתונים ממערכת התפיסה, מנתחת אותם, יוצרת את המפה וממקמת את הרובוט בתוכה, משתמשת במערכת הניווט כדי ליצור תנועה חלקה ולהימנע ממכשולים. אח"כ, מערכת הבקרה מעבירה את המסלול שמצאה למערכת הקינמטיקה המחשבת את זוויות סיבובי המפרקים של הרובוט בכל צעד לפי המסלול המתוכנן ושולחת את הפקודות למנועים ולמפרקים.

סיכום

תהליך המיפוי, המיקום, תכנון המסלול והקינמטיקה ההפוכה מביא לידי ביטוי את האוטונומיה המושלמת של הרובוט, שבה הוא יכול לנווט ולפעול בסביבה מורכבת תוך יכולת שליטה והתאמה עצמית למשימות שונות. כל אלו יחד מאפשרים לרובוט לפעול באופן עצמאי ומדויק בשדה הקרב, תוך מינימום התערבות אנושית.

פרק ראשון: הבינה המלאכותית בניווט האוטונומי:

הבינה המלאכותית בתתי המערכת הרובוטית:

מערכות רובוטיות חכמות נעזרות באלגוריתמים חזקים של הבינה המלאכותית כדי לדמות לרובוט תכונות אנושיות כגון קבלת החלטות מושכלות בזמן אמת, והבנת סביבה פיזית סבוכה כדרך שבני אנוש יבינו אותה.

אלגוריתמים ידועים לדוגמא הם:

גישות הסתברותיות, Machine learning, אלגוריתמים לתכנון מסלול, רשתות נוירונים ועוד.

שיטות הבינה המלאכותית משולבות בתתי-מערכות שונות של הרובוט, כולל במערכות הניווט, הזיהוי והמיקום. הן מסייעות לרובוט ל"הבין" את המצב הנוכחי שלו, לזהות את מצב המטרה, המכשולים והאתגרים בדרך.

אלגוריתמי בינה מלאכותית הבאים לידי ביטוי ברובוטים אוטונומיים:

במהלך המאמר מצוינים מספר אלגוריתמי בינה מלאכותית המיושמים בהקשר של רובוטים ניידים אוטונומיים, במיוחד בתחום מערכות התפיסה, עיבוד נתוני חיישנים ותכנון מסלולים. אציג להלן אלגוריתמי בינה מלאכותית שהוצגו במאמר הללו משמשים בהקשר של בינה מלאכותית בתתי המערכת הרובוטית שהוזכרו קודם או בחלקי המערכת שלא הוזכרו. אלו אלגוריתמי בינה מלאכותית וישנם רבים אחרים הבאים לידי ביטוי ברובוטיקה ומסיעים לה להיות אוטונומית וחכמה.

לרגע אסביר את המושג "מערכות תפיסה של רובוטים" כדי להבין טוב יותר את אלגוריתמי הבינה המלאכותית שיוצגו בהמשך:

מערכות התפיסה של רובוטים הן חלק חשוב במערכת הכוללת של רובוט אוטונומי, והן בעצם העיניים והאוזניים של הרובוט. הן מאפשרות לרובוט לאסוף נתונים מהסביבה באמצעות חיישנים ולעבד את המידע כדי להבין ולהגיב למצבים שונים בסביבתו. מערכת התפיסה ברובוטים כוללת את כל התהליכים והטכנולוגיות שמעורבות בזיהוי, ניתוח והבנה של המידע הסביבתי.

חלקים במערכות התפיסה כוללים את הטכנולוגיות הבאות:

1. חיישנים: מכשירים אלקטרוניים שמודדים גודל פיזי כלשהו וממירים אותו לסיגנל שניתן לקרוא ולעבד. דוגמאות לחיישנים כוללות מצלמות, חיישני לייזר (LIDAR), חיישני אולטרסאונד, חיישני מרחק, ועוד.

2. עיבוד נתונים: תהליך שבו הנתונים שנאספים על ידי החיישנים מומרים למידע מועיל. זה כולל אלגוריתמים של עיבוד תמונה, זיהוי דפוסים, ולמידת מכונה כדי לזהות עצמים, מכשולים, ותכונות נוספות בסביבה.

3. אינטגרציה של נתונים: שילוב של נתונים ממקורות חיישנים שונים כדי לקבל תמונה כוללת ומדויקת של הסביבה. אינטגרציה זו מאפשרת לרובוט להבין את הסביבה בצורה טובה יותר ולפעול בה בהתאם.

מערכות התפיסה מאפשרות לרובוטים לקלוט ולהבין את הסביבה בצורה מדויקת ומפורטת. הן קריטיות לתפקודם האוטונומי של רובוטים, ומאפשרות להם לבצע משימות בסביבות מורכבות ומשתנות. מערכות אלו מהוות את הבסיס לכל פעולות הרובוטים האוטונומיים שנדון בהן בהמשך.

נמשיך,

כפי שאמרתי מוצגים להלן חלק מאלגוריתמי הבינה המלאכותית המוצגים במאמר:

1. אלגוריתמים של למידת מכונה (Machine Learning):

- משמשים לזיהוי דפוסים ולמידה מנתונים שנאספו.

- תומכים בעיבוד נתוני חיישנים, זיהוי עצמים, וזיהוי מכשולים.

2. גישות הסתברותיות בבינה מלאכותית:

- משמשות לניווט ומיקום, לניהול אי-ודאויות בנתוני חיישנים ותנועת הרובוט.

- כוללות טכניקות כמו ניווט מונטה קרלו ופילטרי קלמן להערכת מיקום הרובוט.

3. אלגוריתמי תכנון מסלול:

- משמשים לחישוב מסלולים אופטימליים לניווט רובוטים תוך התחשבות במכשולים.

- כוללים אלגוריתמים כמו A* ואלגוריתם דייקסטרה למציאת מסלולים קצרים ובטוחים.

4. אלגוריתמי התפתחות והמונים:

- משמשים לפתרון בעיות אופטימיזציה מורכבות בתכנון מסלול רובוטי.

- כוללים טכניקות כמו אופטימיזציה לפי חבורת חלקיקים ואופטימיזציה לפי קולוניית נמלים למציאת מסלולים אופטימליים.

5. רשתות נוירונים מלאכותיות (Artificial Neural Networks):

- מחקות תהליכי למידה של המוח האנושי, לזיהוי דפוסים ועיבוד נתונים מורכבים.

- משמשות לזיהוי עצמים, זיהוי דיבור, והפרדה של רעש משיחות.

6. רשתות נוירונים עם הפצה אחורית (Back Propagation Neural Networks):

- משמשות ללמידה מהירה ויעילה מנתוני חיישנים במהלך הניווט.

- מאפשרות לרובוטים ללמוד דרך ניסוי וטעיה ולהגיב למצבים חדשים.

7. פילטור חלקיקים (Rao-Blackwellized Particle Filter):

- משמשים לפיתוח מפות דו-ממדיות של סביבות עבודה לניווט אוטונומי.

- משלבים תכנון מסלול גלובלי ומקומי למציאת המסלול הטוב ביותר למטרה תוך עקיפת מכשולים.

8. שיטות חישוב רך (Soft Computing Methods):

- משמשות לפתרון בעיות ניווט של רובוטים והימנעות ממכשולים.

- כוללות אלגוריתמים דטרמיניסטיים לפתרון אופטימלי של בעיות ניווט.

במאמר מוצגים אלגוריתמים נוספים, אך אלו שהוצגו כאן הם בין החשובים שבינהם .

כל אחד מאלגוריתמים אלו משחק תפקיד חשוב באספקטים שונים של פונקציונליות הרובוט, מתנועה בסיסית והימנעות ממכשולים ועד לאינטראקציה מורכבת עם הסביבה והתאמה אליה. הם מאפשרים לרובוטים לבצע משימות בצורה יעילה ואמינה יותר במגוון רחב של הגדרות וסביבות בלתי צפויות.

אולי נתמקד באופן ספציפי יותר במספר אלגוריתמי בינה מלאכותית בהקשר של הרובוטיקה האוטונומית,

אציג לרגע את הרעיון הידוע למדי של "גישות הסתברותיות – היוריסטיקה"

פונקציה יוריסטית לרובוט בשדה קרב:

נתון גרף $G = (V, E)$ ורובוט הממוקם בצומת v_0 עם מטרה להגיע לצומת v_g.
הפונקציה היוריסטית $h: V \to \mathbb{R}$ מוערכת על ידי:

$$h(v) = d(v, v_g) + \lambda * r(v)$$

כאשר:

- $d(v, v_g)$ מייצג את המרחק האוקלידי בין v לבין v_g.

$r(v)$ - מייצגת פונקציית סיכון בהתאם לקרבה למכשולים או אזורי סיכון.

- λ הוא מקדם שמשקלל את החשיבות של הסיכון בהשוואה למרחק.

דוגמא:

נניח שהרובוט נמצא בצומת v_0 והיעד הוא v_g. המרחק האוקלידי $d(v_0, v_g) = 5$ והרובוט נמצא קרוב לאזור עם מוקשים שמגדיל את $r(v_0) = 3$. אם $\lambda = 0.5$, אז הערכת העלות לצעד הבא היא:

$$h(v_0) = 5 + 0.5 * 3 = 6.5$$

הרובוט יבחר להתקדם לצומת שממנה הערך $h(v)$ הכי נמוך – מכיוון שהערך $h(v)$ הוא ההשערה כמה צעדים עוד נותרו לרובוט אל המטרה v_g.

ע"י פונקציה זו נוכל להריץ אלגוריתם לחיפוש מסלול שיבחר בכל שלב את הצומת שהעלות המשוערת שלו היא הנמוכה ביותר – שההשערה כמה עוד צעדים נותרו אל המטרה היא הקטנה ביותר.

אלגוריתם בינה מלאכותית ספציפי נוסף בו נתמקד יהיה האלגוריתם החשוב בתת מערכת הניווט והמיקום – אלגוריתם מונטה קארלו (Monte Carlo Localization).

אלגוריתם מונטה קארלו הוא שיטה הסתברותית המיועדת למציאת מיקום לא ידוע של עצמים במרחב, כאשר המיקום המדויק אינו ידוע מראש. בתחום הרובוטיקה, השימוש העיקרי באלגוריתם זה הוא לזיהוי מיקום רובוט בסביבה מסוימת על ידי יצירת מגוון גדול של "ניחושים" או חלקיקים, שכל אחד מהם מייצג מצב אפשרי בו הרובוט יכול להימצא.

בפועל, אלגוריתם מונטה קארלו מגדיר מספר רב של חלקיקים עם מיקומים רנדומליים במרחב, ומשתמש בנתונים מהחיישנים של הרובוט כדי לעדכן ולתקן את מיקומם של החלקיקים בהתאם לתנועת הרובוט. בכל שלב, האלגוריתם מעריך את ההסתברות שחלקיק מסוים מייצג באופן מדויק את מיקום הרובוט ומשקלל את כל החלקיקים על פי מידת התאמתם לנתוני המדידה. בסופו של תהליך, החלקיקים שנותרים בעלי המשקל הגבוה ביותר מצביעים על ההערכה הטובה ביותר למיקום הרובוט.

האלגוריתם מחולק למספר שלבים:

1. אתחול:

יצירת קבוצת חלקיקים באופן רנדומלי, כאשר כל חלקיק מייצג אפשרות למצב הנוכחי של הרובוט (מיקום וכיוון).

2. שלב הניבוי:

האלגוריתם גורם לרובוט לזוז עם מהירות וכיוון מסוימים, ומעדכן את מיקום כל חלקיק בהתאם למודל התנועה של הרובוט. כל חלקיק מוזז לפי המהירות והכיוון של הרובוט, תוך הכנסת רעש רנדומלי לחיזוי כדי לשקף את האי-ודאות בנתוני התנועה.
הנוסחאות לעדכון מיקום החלקיקים הן:

$$X_{new\_place} = x + v * \cos(\theta) * \Delta t + \varepsilon x$$

$$Y_{new\_place} = y + v * \sin(\theta) * \Delta t + \varepsilon y$$

$$\theta_{new\_place} = \theta + \omega * \Delta t + \varepsilon\theta$$

כאשר:

v = מהירות ליניארית

θ = זווית פנייה

ω = מהירות זוויתית

Δt = זמן שחלף בין עדכונים

ε = רעש אקראי המוסף כדי לדמות אי-ודאות

3. שלב העדכון:

מתן ציון לכל חלקיק על פי המרחק ממני ועד אליו, לדוגמה, מהירות החזרה של קרן הלייזר ששולח החיישן לכל אחד מהחלקיקים. פונקציית ההסתברות לבדיקת התאמה(כלומר, עד כמה מיקום החלקיק תואם לערך שנמדד על ידי החיישנים, על פי מרחק ודיוק המדידה) היא:

$$(z_t|x_t) = e^{-(z_t - h\left(\frac{(x_t{}^2)}{2}\right) * \sigma^2)}$$

כאשר z_t הוא הערך הנמדד על ידי החיישנים, x_t הוא המיקום המשוער של החלקיק, h(x_t) היא פונקציית ההערכה שמחשבת מה היה הערך שהחיישנים היו מחזירים אילו החלקיק היה במקום זה, ו σ היא סטיית התקן של הרעש.

4. שלב השקלול:

1. שלב השקלול ומשמעותו:

בשלב השקלול, כל חלקיק מקבל ציון בהתאם למידת הקרבה שלו לרובוט. הציון הזה נמדד על פי המרחק בין המיקום של החלקיק לבין המדידה שנעשתה על ידי החיישנים של הרובוט. לדוגמה, אם הרובוט שולח קרן לייזר, הזמן שלוקח לקרן לחזור יכול לשמש כמדד למרחק. ככל שהמרחק קטן יותר, כך הציון של החלקיק נמוך יותר והוא מקבל משקל גבוה יותר, כלומר סביר להניח שהרובוט נמצא קרוב אליו.

2. תהליך השקלול:

חלקיקים עם ציון גבוה יותר (מרחק גדול מהרובוט) מקבלים משקל נמוך יותר ומשוקללים פחות בחישוב הסופי של המיקום. זאת מכיוון שהם פחות סבירים לייצג את מיקום הרובוט האמיתי. חלקיקים עם ציון נמוך (קרובים לרובוט) מקבלים משקל גבוה יותר, והם נחשבים יותר בחישוב הסופי של המיקום.

בפועל, תהליך השקלול כולל יצירת רשימה של חלקיקים שמשקלם גבוה יותר, והם נשארים ברשימת החלקיקים. חלקיקים שמשקלם נמוך מוחלפים בחלקיקים חדשים המבוססים על ההסתברויות שנמדדו. תהליך זה נקרא resampling.

3. צפיפות חלקיקים ומיקומו המשוער של הרובוט:

אזור שבו יש צפיפות גבוהה של חלקיקים הוא האזור שבו ככל הנראה הרובוט נמצא. ככל שיותר חלקיקים מתרכזים באזור מסוים, זהו סימן חזק לכך שהרובוט נמצא קרוב

לאותו אזור. הסיבה לכך היא שבאזור זה, המדידות מהחיישנים תואמות בצורה הטובה ביותר את המיקומים המשוערים של החלקיקים.

העיקרון המתמטי שעומד מאחורי שלב השקלול הוא שבכל איטרציה, האלגוריתם מחפש לשפר את הערכת המיקום של הרובוט על ידי שילוב של מידע חדש מהחיישנים עם ההערכות הקודמות של המיקום. התהליך הזה של שילוב המידע מבוסס על פילטרים הסתברותיים, שמטרתם לשפר את הדיוק של ההערכות על ידי שקלול ההסתברויות של כל אחד מהחלקיקים.

הפונקציה ההסתברותית שמשמשת לקביעת המשקל של כל חלקיק היא פונקציה אקספוננציאלית, מה שמבטיח שחלקיקים עם מרחקים גדולים מהמיקום האמיתי של הרובוט יקבלו משקל נמוך מאוד, בעוד שחלקיקים קרובים יקבלו משקל גבוה. בכך, האלגוריתם מבטיח שהתפלגות החלקיקים תתכנס למיקום האמיתי של הרובוט.

5. סיום ריצת האלגוריתם:

האלגוריתם עוצר כאשר כל החלקיקים מתרכזים במקום מסוים, שהוא כנראה המקום האמיתי של הרובוט. הריכוז הגבוה של החלקיקים במקום מסוים הוא אינדיקציה לכך שהמיקום הזה תואם בצורה הטובה ביותר את המדידות מהחיישנים ואת ההערכות הקודמות. בשלב זה, האלגוריתם יכול להפסיק את ריצתו ולספק הערכה מדויקת של מיקום הרובוט.

וכך נסיים את סריקת האלגוריתם, באמצעות השימוש בתהליכים הסתברותיים, תהליך resampling, ושילוב מידע מהחיישנים, אלגוריתם מונטה קארלו מספק שיטה אפקטיבית ומדויקת לזיהוי מיקום הרובוט בסביבה לא ידועה.

לאחר פירוט נרחב של אלגוריתמי בינה מלאכותית המתוארים במאמר ומשמשים רבות ברובוטיקה האוטונומית נעבור ללב הנושא – פירוט כיצד פועל הרובוט להשגת המטרה, כיצד ישולבו האלגוריתמים שתוארו או חלקם במערכת התפיסה של הרובוט ומיד לאחר מכן פרוט נרחב של שתי אלגוריתמים נבחרים מתוך מערכת התפיסה – אלגוריתם SLAM במערכת המיקום ואלגוריתם RRT במערכת תכנון מסלול. נצא לדרך.

מיקום (Localization)

המיקום מתייחס ליכולת של הרובוט לקבוע את מקומו בסביבה באופן מדויק. טכניקות מיקום מתקדמות כוללות שימוש ב-GPS בחוץ ובמערכות זיהוי ואיתור כמו לייזר ו-WLAN למיקום בפנים, מונטה קארלו ועוד. SLAM, שאינו נזכר במאמר אך חשוב

להבנת התחום, מאפשר לרובוט לבצע מיפוי ומיקום במקביל, תוך כדי יצירת מפת הסביבה בזמן אמת והגדרת מיקומו בה במדויק.

ניווט ותכנון מסלול

הניווט ברובוטים אוטונומיים כולל את היכולת לזהות ולהגיב לסביבה בזמן אמת תוך כדי תכנון וביצוע של מסלול ליעד מוגדר. מערכות ניווט מתקדמות משלבות חיישנים ואלגוריתמים שמתעדכנים על בסיס המידע שנאסף. במאמר נדון בשימוש ב-Dynamic Window Approach (DWA), אשר מאפשר לרובוט להתמודד עם מכשולים תוך שמירה על מהירות וזוויות פנייה מתאימות. לצורך תכנון מסלול בסביבות יותר מורכבות, שימוש באלגוריתם כמו RRT יכול להועיל מאוד. RRT מתאים במיוחד לסביבות עם מכשולים רבים ומשתנים, שכן הוא מייצר במהירות עץ חיפוש שמתרחב לאזורים חדשים ולא מוכרים, מה שמאפשר לרובוט למצוא נתיב חדש ובטוח במהירות – יפורט בהמשך.

קינמטיקה

בתחום הקינמטיקה, המאמר מתאר כיצד מערכת הניווט של הרובוט משתמשת בנתונים קינמטיים לתכנון וביצוע מסלול התנועה. הקינמטיקה עוסקת בחישובים של מיקום, מהירות ותאוצה של הרובוט בלי להתחשב בכוחות שגורמים לתנועה. ניתן למצוא במאמר שימוש בחיישנים כמו מקודדים על הגלגלים(חיישנים הממירים אותות מכניים לאותות חשמליים), המדווחים על סיבובי הגלגלים שמהם ניתן להסיק על המהירות והמרחק שעבר הרובוט. נתונים אלו משמשים לחישוב ותיקון מסלול הרובוט בזמן אמת, כדי להבטיח שהוא עקבי במסלול המדויק שתוכנן.

דינמיקה

בפן הדינמי של התנועה, המאמר מתאר כיצד הרובוט מתמודד עם כוחות חיצוניים והתנגדויות שעלולות להשפיע על התנועה. הדינמיקה עוסקת בהשפעות של כוחות כמו חיכוך, כבידה והתנגדות האוויר, וכיצד הם משפיעים על הקצב והיכולת של הרובוט לבצע מסלול מסוים. כדי לשלוט על תגובות אלה, רובוטים משתמשים במערכות בקרה שמתאימות את הפעולה בהתאם לדרישות המסלול ולמגבלות הפיזיקליות של הרובוט.

אחת השיטות שמוזכרת במאמר לניהול דינמיקה זו היא שימוש ב-PID Controller, שהוא רגולטור משוב קלאסי המשמש לשליטה במהירות ובמיקום של הרובוט. הרגולטור מתאים את פלט המנוע בהתאם לשגיאה בין המיקום הרצוי למיקום הנוכחי, מה שמאפשר לרובוט לשמור על תזוזה יציבה וחלקה לכיוון היעד.

באמצעות שילוב של קינמטיקה ודינמיקה, הרובוט יכול לנווט ולהגיע ליעדו ביעילות תוך שהוא מתמודד עם אתגרי הסביבה והמשטח שעליו הוא נע.

פרק שני: האלגוריתם SLAM: מיפוי ומיקום בו-זמנית

בואו ונצא למסע מרתק בעולם של טכנולוגיות מתקדמות ברובוטיקה, נחקור את האלגוריתם שהפך את הרובוטים לכלים קרביים רבי עוצמה: SLAM - Simultaneous Localization and Mapping.

שלב ראשון: איסוף נתונים

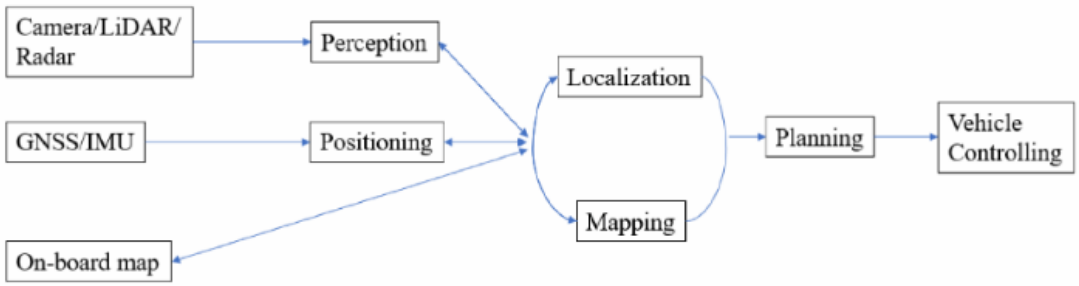
המסע מתחיל באיסוף נתונים. דמיינו רובוט צבאי הנע דרך שדה קרב, מצויד בחיישנים מתקדמים כמו LIDAR (Light Detection and Ranging), מצלמות ו-IMU (Inertial Measurement Unit). חיישני LIDAR מאפשרים לרובוט למפות את השטח על ידי שליחת קרני לייזר ומדידת הזמן שלוקח להן לחזור אליו. מצלמות מספקות תמונות וידאו, ו-IMU מספק נתונים על התנועה והזווית של הרובוט. חיישנים אלו מעניקים לרובוט את היכולת "לראות" את העולם סביבו, בדיוק כפי שהחיישנים שלנו - העיניים והאוזניים - מסייעים לנו להבין את הסביבה. לדוגמה, חיישן LIDAR יכול לזהות אויבים ומכשולים בשדה הקרב במרחק של עשרות מטרים ולסייע לרובוט להימנע מהם.

שלב שני: תהליך ראשוני ואינטגרציה של נתונים

בשלב זה, המידע המתקבל מהחיישנים עובר סינון ועיבוד ראשוני. תמונות ממצלמה עשויות להיות מטושטשות או עם רעש, ולכן יש צורך בטכניקות עיבוד תמונה כמו Gaussian Blur לסינון רעשים ולהבהרת התמונה. הרובוט משתמש באלגוריתמים מתקדמים כמו Kalman Filter או Particle Filter כדי לשלב את כל המידע מהחיישנים השונים ולהפיק תמונה כוללת ומדויקת של הסביבה. כל חיישן מספק חלק קטן מהתמונה, והרובוט מאחד את כל החלקים לכדי תמונה שלמה.

שלב שלישי: אמידת מיקום ועדכון מפה

הגענו לשלב הקריטי: אמידת המיקום של הרובוט ועדכון המפה. כאן הרובוט משתמש בטכניקות מתמטיות להערכת מיקומו במרחב וליצירת עדכון מפה של הסביבה. רובוט לחימה, למשל, עשוי להשתמש בטכניקות כמו Kalman Filter לאמידת מצב, המשלבות נתונים קודמים (כגון המיקום המשוער מה-IMU) עם מדידות חדשות מה-LIDAR ומהמצלמות כדי לקבל הערכה מדויקת יותר של מיקומו.



שלב רביעי: אופטימיזציה ויישומים ברובוטיקה צבאית

ברובוטיקה צבאית, חשוב שהמפות יהיו מדויקות ככל האפשר. האלגוריתם משתמש בשיטות אופטימיזציה מתקדמות כמו Graph SLAM, שבו נבנה גרף של מיקומים והקשרים ביניהם. האופטימיזציה מבוצעת כדי למזער שגיאות ולהשיג מפה מדויקת יותר. דמיינו רובוט שפועל במתחם עירוני מורכב, עם מבנים גבוהים ומכשולים רבים. האלגוריתם מסייע לרובוט להבין את סביבתו במדויק, למנוע התנגשויות ולספק נתונים טקטיים חשובים לכוחות האנושיים.

אתגרים ופתרונות

האלגוריתם SLAM עומד בפני אתגרים רבים בשדה הקרב: שינויים מהירים בסביבה, תנאים קשים כמו אבק ועשן, וצורך בהתמודדות עם תקלות חיישנים. דמיינו רובוט שנמצא בשטח פתוח שבו פתאום מתחוללת סופת חול. האלגוריתם חייב להיות חזק וגמיש מספיק כדי להמשיך לפעול ביעילות גם בתנאים אלו. שימוש בטכנולוגיות מתקדמות כמו Machine Learning מאפשר לרובוט ללמוד ולהתאים את עצמו למצבים משתנים, לשפר את דיוק המפות ולהגביר את יעילות המשימות.

באמצעות אלגוריתם SLAM, רובוטים צבאיים יכולים להבין בדיוק היכן הם נמצאים ביחס למכשולים, לכוחות האויב וליעדים שלהם, ולהתקדם בצורה מדויקת ואפקטיבית לעבר המטרה.

הבדל בין סביבות קבועות לסביבות משתנות באלגוריתם SLAM:

ההבדל בין סביבות משתנות לקבועות משפיע באופן משמעותי על האפקטיביות והמהירות של אלגוריתם SLAM.

סביבות קבועות

בסביבה קבועה, כל העצמים נשארים במקומם, וזה מאפשר ל-SLAM לייעל את תהליך המיפוי והלוקליזציה. במצב זה, הרובוט משתמש במערכת חיישנים כדי לזהות נקודות ייחודיות (landmarks) בסביבה ולמדוד את המרחק (d_i) והזווית (θ_i) אליהם על מנת לשפר את דיוק המיקום שלו על המפה.

נניח שהרובוט מזהה נקודה ייחודית L_i במיקום (x_i, y_i) במרחב. המרחק והזווית מהנקודה הזו ימדדו בצורה הבאה:

$$d_i = \sqrt{(y_r - y_i)^2 - (x_r - x_i)^2}$$

$$\theta_i = tan^{-1}(\frac{y_r - y_i}{x_r - x_i})$$

כאשר (x_r, y_r) הוא מיקום הרובוט.

הקביעות של העצמים מאפשרת לאלגוריתם להיות מדויק יותר ולהסתמך על המידע שכבר אסף בעבר. המשמעות היא שהרובוט יכול להשתמש באלגוריתמים מתקדמים כמו Kalman Filter כדי לשלב מדידות חוזרות ונשנות, ולהקטין את השגיאות המצטברות:

$$X_{k|k} = K_k\left(z_k - h\left(X_{k|k-1}\right)\right) + X_{k|k-1}$$

$$P_{k|k} = P_{k|k-1}(I_k - K_k H)$$

כאן K היא מטריצת Kalman Gain z היא המדידה החדשה ו-H היא מטריצת המדידה. יפורט בהמשך.

סביבות משתנות

בסביבה משתנה, עצמים כמו רהיטים או כלים נעים יכולים להתמקם מחדש, מה שמציב אתגרים משמעותיים לפני האלגוריתם. במצב כזה, הרובוט צריך להיות מסוגל להבחין בין עצמים קבועים לעצמים משתנים ולעדכן את המפה בזמן אמת. נניח כי קיימת פונקציית סיכון r(v) המתארת את הקרבה למכשולים או אזורי סיכון:

$$h(v) = d\left(v, v_g\right) + \lambda * \mathrm{r(v)}$$

כאשר h(v) היא הפונקציה היוריסטית המעריכה את עלות התנועה לצומת v_g , d(v,v_g) הוא המרחק האוקלידי לצומת היעד, ו-λ הוא מקדם שמשקלל את החשיבות של הסיכון.

זיהוי של עצמים משתנים מחייב שימוש באלגוריתמים מתקדמים של למידת מכונה ועיבוד תמונה. לדוגמה, שימוש ברשתות נוירונים לזיהוי אובייקטים ושימוש באלגוריתם Particle Filter המאפשר לאלגוריתם לשמור על מספר השערות במקביל ולבצע resampling על סמך ההסתברויות:

$$t^{[m]}{}_w = \frac{p(\mathbf{z}_t|\mathbf{x}_t^{[m]})p(\mathbf{x}_t^{[m]}|\mathbf{x}_{t-1}, \mathbf{u}_t)}{p(\mathbf{z}_t|\mathbf{x}_t^{[i]})p(\mathbf{x}_t^{[i]}|\mathbf{x}_{t-1}, \mathbf{u}_t) \sum_{i=1}^{M}}$$

כאשר $t^{[m]w}$ הוא המשקל של חלקיק m בזמן t, ו- M הוא מספר החלקיקים.

מדוע זה משנה?

היכולת להבדיל בין עצמים קבועים למשתנים היא קריטית מכיוון שהיא משפיעה ישירות על יעילות ודיוק המיפוי והלוקליזציה של הרובוט. במיוחד ברובוטיקה צבאית או רובוטים המופעלים בסביבות דינמיות כמו בתי חולים או מפעלים, חשוב מאוד שהמערכת תוכל להגיב במהירות לשינויים ולעדכן את הנתונים בהתאם. זיהוי נכון של עצמים קבועים משפר את היכולת לנווט בסביבה מוכרת, בעוד שהתמודדות עם עצמים משתנים דורשת מהרובוט להיות ערני ומוכן להתאים את תכנון המסלול בהתאם למציאות החדשה שהתגלתה.

סיכום

בזאת, נסכם את המסע התאורטי המרתק שלנו בנבכי אלגוריתם SLAM. זהו אלגוריתם שמהווה את הבסיס לטכנולוגיות מתקדמות ביותר ברובוטיקה הצבאית, ומאפשר לרובוטים להבין את הסביבה שלהם ולפעול ביעילות רבה יותר בשדה הקרב. נעבור לסקור באופן פרטי ואלגוריתמי טכנולוגיות שונות של SLAM.

אלגוריתם EKF-SLAM: טכנולוגיה חדשנית לניווט עצמאי של רובוטים

ארצה להציג בפניכם את אלגוריתם EKF-SLAM, טכנולוגיה מדהימה מעולם ה-SLAM. כיצד אלגוריתם EKF-SLAM מאפשר לרובוטים לנווט באופן עצמאי בסביבות מורכבות?

מבוא: בעיית הניווט העצמאי

דמיינו רובוט קטן, חסר מושג היכן הוא נמצא. סביבו, עולם זר ומלא אתגרים. איך יוכל הרובוט למצוא את דרכו? התשובה טמונה באלגוריתם EKF-SLAM, כלי רב עוצמה המאפשר לו ליצור מפה מדויקת של סביבתו ולנווט בתוכה בביטחון.

המסע מתחיל במפה קיימת, נתונה מראש, המהווה עבור הרובוט נקודת התייחסות. כעת, מצויד ב-LiDAR, סורק לייזר מתקדם, הרובוט יוצא למסע גילוי. קרני לייזר עדינות נורות לכל עבר, ומהן מתקבלת תמונה תלת-ממדית מפורטת של הסביבה.

תהליך העבודה של EKF-SLAM

המסע ממשיך כאשר הרובוט משווה את תמונת הסריקה הנוכחית למפה הקיימת, תוך התחשבות במיקומו המשוער. זה כמו משחק פאזל מתוחכם, שבו הרובוט מנסה להתאים את החלקים יחדיו וליצור תמונה שלמה.

לאחר מכן, מגיע תורם של חיישנים נוספים. חיישני IMU מודדים את תנועת הרובוט, בעוד שמערכת GNSS (Global Navigation Satellite System) מספקת מידע על

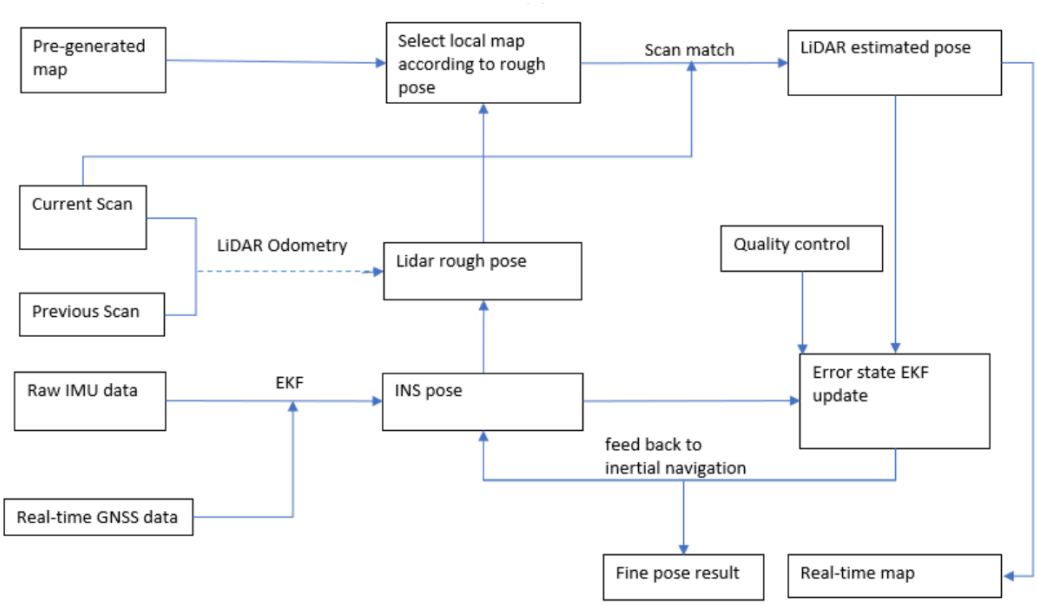
מיקומו הגיאוגרפי. כל אלה משולבים יחד כדי להבטיח שהרובוט תמיד ידע היכן הוא נמצא, גם בסביבות סגורות או מאתגרות.

שימוש באלגוריתם EKF לתיקון טעויות

אך הדרך לא נטולת מכשולים. טעויות עלולות להתרחש, מידע עשוי להיות לא מדויק. כאן נכנס לתמונה אלגוריתם EKF (Extended Kalman Filter), כלי חכם המסייע לרובוט להתגבר על קשיים אלו. EKF מנתח את כל המידע הזמין, מתקן טעויות ומעדכן את מיקום הרובוט באופן רציף.

התהליך הזה חוזר על עצמו שוב ושוב. הרובוט סורק, משווה, מתאים, מחשב, מתקן ומעדכן. לאט לאט, המפה הולכת ומתמלאת, הופכת מדויקת יותר ויותר. הרובוט לומד את סביבתו, צעד אחר צעד, ומתקרב ליעדו.

אלגוריתם EKF-SLAM:



השלבים בתהליך EKF-SLAM לפי התרשים

Pre-generated map (מפה שנוצרה מראש)

תחילת המסע עם מפה קיימת של הסביבה. זו נקודת ההתחלה של הרובוט.

Select local map according to rough pose (בחירת מפה מקומית לפי מיקום משוער)

הרובוט בוחר חלק מהמפה הקיימת על פי המיקום המשוער שלו.

Current Scan (סריקה נוכחית)

הרובוט סורק את הסביבה עם סורק LiDAR כדי לקבל תמונה עדכנית של הסביבה.

LiDAR rough pose (מיקום ראשוני על פי סריקות LiDAR)

מיקום ראשוני של הרובוט מחושב על פי הסריקות הנוכחיות.

LiDAR estimated pose (מיקום משוער על פי LiDAR)

מיקום משוער יותר מדויק של הרובוט מחושב על פי התאמה לסריקות קודמות ולמפה הקיימת.

Quality control (בקרת איכות)

בקרת איכות של המידע שנאסף לוודא שהוא מדויק.

Previous Scan (סריקה קודמת)

השוואה לסריקות קודמות כדי לאתר שינויים בסביבה.

INS pose (מיקום על פי מערכת ניווט אינרציאלית)

שימוש בחיישנים אינרציאליים IMU למדידת תנועות וזוויות של הרובוט.

Raw IMU data (נתונים גולמיים מחיישני IMU)

נתונים גולמיים מהחיישנים האינרציאליים שמודדים את תאוצת וזוויות הרובוט.

Error state EKF update (עדכון מצב שגיאה באמצעות EKF)

אלגוריתם EKF מנתח את הנתונים, מתקן טעויות ומעדכן את מיקום הרובוט.

שלב זה יורחב בהמשך שכן הוא מהווה לב האלגוריתם EKF-SLAM.

Real-time GNSS data (נתוני GNSS בזמן אמת)

נתוני GNSS בזמן אמת מספקים מידע גיאוגרפי מדויק על מיקום הרובוט.

LiDAR Odometry (מדידת תנועה מבוססת LiDAR)

מערכת למדידת מרחקים ותנועות הרובוט באמצעות סריקות LiDAR.

Feedback to inertial navigation (משוב לניווט אינרציאלי)

תוצאות העדכונים נשלחות חזרה למערכת ה-INS כדי לשפר את דיוק הניווט.

Fine pose result (תוצאה מדויקת של מיקום)

לאחר כל החישובים והתיקונים, הרובוט מקבל מיקום מאוד מדויק שלו.

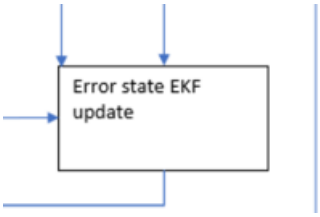
Real-time map (מפה בזמן אמת)

המפה מתעדכנת בזמן אמת עם המיקום המדויק של הרובוט ופרטי הסביבה.

סיכום התהליך

הרובוט מתחיל עם מפה קיימת ומשתמש בסריקות LiDAR, חיישנים אינרציאליים ונתוני GNSS כדי לעדכן את מיקומו ואת המפה בזמן אמת. הוא סורק את הסביבה, משווה את המידע למפה הקיימת, ומחשב את מיקומו בצורה מדויקת תוך כדי תיקון טעויות. כל הנתונים משולבים יחד כדי להבטיח שהרובוט תמיד יודע היכן הוא נמצא ובונה מפה מדויקת של הסביבה.

השלב החשוב: Error state EKF update



לאחר הבנת סדר הפעולה של האלגוריתם עפ"י התרשים נעבור בעז"ה להתמקדות בשלב אחד חשוב במיוחד הוא שלב Error state EKF update, השלב בו אלגוריתם EKF מנתח את הנתונים, מתקן טעויות ומעדכן את מיקום הרובוט.

פירוט השלבים בעדכון מצב שגיאה באמצעות EKF

חישוב מטריצת קלמן Kk = (Kalman Gain)

מטריצה זו עוזרת להחליט כמה לסמוך על המדידות החדשות לעומת מה שכבר ניחשנו. מטריצת הקלמן מאזנת בין אי-הוודאות בתחזית הקודמת ובין המידע החדש שמתקבל מהמדידות, ומספקת ערך אופטימלי לעדכון.

עדכון מצב הרובוט Xk|k = (State Update)

כאן אנו מעדכנים את המיקום של הרובוט בעזרת המדידות החדשות. מטריצת הקלמן מחושבת על פי המידע החדש ונוספת להערכת המיקום הקודמת, ובכך מבטיחה דיוק מרבי של המיקום.

עדכון מטריצת הקוריאציה Pk|k = (Covariance Update)

כאן אנו מעדכנים את הידע שלנו על כמה אנחנו בטוחים במיקום של הרובוט. עדכון זה כולל את הפחתת האי-ודאות הכוללת על בסיס המדידות החדשות והערכת המיקום המעודכנת.

פירוט:

חישוב מטריצת קלמן K:

$$K_k = P_{k|k-1}{}^{-1} H_K^T (H_k P_{k|k-1} H_K^T + R_k)$$

הנוסחה מתחילה במכפלה של המטריצה P שאומרת לנו כמה אנחנו בטוחים במיקום הנוכחי של הרובוט עם המטריצה הטרנספורמציה של H. מכפלה זו עוזרת לנו להעריך את התחזית של המדידה על בסיס המיקום הנוכחי.

לאחר מכן, בתוך הסוגריים, אנחנו מחברים את המטריצה H שמחברת בין המיקום הנוכחי למדידה החדשה, שוב עם המטריצה P שמייצגת את מידת הביטחון שלנו, ועוד המטריצה R שמייצגת את האי-וודאות במדידות החדשות.

לבסוף, אנחנו לוקחים את המטריצה ההופכית של הסכום הזה, כדי לקבל את המטריצה קלמן K. מטריצת קלמן משקללת את המידע מהמדידות החדשות ומעדכנת את התחזיות שלנו בהתאם.

אז אם נסתכל על המטריצות שבנוסחא:

- המטריצה P אומרת לנו כמה אנחנו בטוחים במיקום הנוכחי של הרובוט לפני המדידה החדשה. זו הערכת הביטחון שלנו.

- המטריצה H מחברת בין המיקום הנוכחי למדידה החדשה. היא עוזרת לנו להבין איך המדידות שלנו צריכות לשנות את התחזיות הקודמות.

- המטריצה R מייצגת את האי-וודאות במדידות החדשות. אם המדידות פחות מדויקות, המטריצה הזו תהיה גדולה יותר.

- המטריצה K זוהי המטריצה קלמן עצמה. היא קובעת כמה להאמין למדידות החדשות לעומת התחזיות הקודמות שלנו.

סיכום

מטריצה קלמן היא חלק חשוב באלגוריתם EKF-SLAM, כי היא קובעת **כמה להאמין למדידות החדשות לעומת התחזיות הקודמות שלנו.** המטריצות P, H, ו-R כולן תורמות לחישוב הזה, כדי לעדכן את מיקום הרובוט בצורה מדויקת ויעילה יותר.

נוסחת עדכון מצב הרובוט X

$$X_{k|k} = (K_k \left(z_k - h(X_{k|k-1})\right) + X_{k|k-1}$$

המיקום החדש של הרובוט מתקבל על ידי הוספת התיקון למיקום המשוער הקודם. התיקון הזה מחושב באמצעות מכפלה של מטריצת קלמן K בהפרש בין המדידה החדשה לבין התחזית שלנו למדידה החדשה.

- המיקום המשוער הקודם X זהו המיקום של הרובוט כפי שהוערך לפני קבלת המדידה החדשה.

- המדידות החדשות Z אלו המדידות החדשות שהתקבלו מהחיישנים. המדידות מספקות מידע חדש על מיקום הרובוט והסביבה.

- התחזית שלנו למדידה החדשה H התחזית שלנו לגבי מה שהמדידות היו אמורות להיות על פי המיקום המשוער הקודם.

התוצאה היא המיקום המעודכן של הרובוט, אשר משלב את המידע מהמדידה החדשה עם התחזיות הקודמות שלנו.

סיכום

תהליך זה מבטיח שהמיקום של הרובוט יעודכן בצורה אופטימלית ומדויקת, על בסיס המדידות החדשות שהתקבלו מהחיישנים. זה מאפשר לרובוט לשפר את ההבנה שלו לגבי מיקומו בסביבה ולהתאים את התחזיות שלו בהתאם.

עדכון מטריצת הקוואריאציה P

$$P_{K|k} = P_{k|k-1}(I_k - K_k H)$$

המטריצה P מייצגת את הביטחון שלנו במיקום המעודכן של הרובוט. מטריצת הקוואריאציה המעודכנת מתקבלת על ידי הכפלת ההפרש בין מטריצת היחידה I למכפלה של מטריצת קלמן K והמטריצה H במטריצת הקוואריאציה הקודמת P.

הסבר על מרכיבי הנוסחה:

- מטריצת היחידה I מייצגת את שמירת המצב הנוכחי.

- מטריצת קלמן K קובעת כמה להאמין למדידות החדשות.

- המטריצה H מחברת בין המיקום הנוכחי למדידה החדשה.

- המטריצה P הערכת הביטחון שלנו במיקום לפני המדידה החדשה.

התוצאה היא מטריצת הקוואריאציה המעודכנת, שמייצגת את הביטחון שלנו במיקום המעודכן של הרובוט לאחר המדידה החדשה.

סיכום

תהליך זה מבטיח שהערכת אי-הוודאות שלנו תתעדכן בצורה אופטימלית לאחר קבלת המדידות החדשות, ובכך תשפר את הדיוק הכולל של מערכת ה-SLAM. זה מאפשר לרובוט לשמור על מיקום מדויק יותר ולשפר את התחזיות שלו בהתאם.

כיצד המטריצות מאפשרות את מימוש המטרה באלגוריתם EKF-SLAM

בואו נבין בצורה פשוטה איך אלגוריתם EKF-SLAM משתמש במטריצות כדי להשיג את המטרה של מיפוי ומיקום הרובוט בצורה מדויקת.

תחילת התהליך: ניבוי המיקום

דמיינו שהרובוט שלנו נמצא במקום כלשהו והוא צריך לנבא את המיקום הבא שלו. בשביל זה, הוא משתמש במודל מתמטי שנקרא "מטריצת מעבר" שמנבאת איך התנועה שלו תשפיע על המיקום. זו כמו מפת דרכים שאומרת לרובוט לאן ללכת בהתאם לתנועותיו.

ניבוי האי-ודאות: מטריצת הקוריאציה

במקביל לניבוי המיקום, הרובוט גם מנבא כמה הוא בטוח במיקום הנוכחי שלו. זה נעשה באמצעות "מטריצת הקוריאציה". אם המטריצה הזו מראה מספרים גדולים, זה אומר שיש הרבה אי-ודאות במיקום הנוכחי. אם המספרים קטנים, זה אומר שאנחנו די בטוחים במיקום.

קבלת מדידות חדשות:

הרובוט ממשיך לנוע ומקבל מדידות חדשות מהחיישנים שלו. המדידות האלה מושוות לתחזיות הקודמות. ההבדל ביניהן נקרא "חידוש המדידה". אם ההבדל גדול, זה אומר שיש שגיאה גדולה בתחזית, ואם הוא קטן, זה אומר שהתחזית שלנו הייתה מדויקת.

עדכון המיקום: מטריצת קלמן

עכשיו, הרובוט צריך לעדכן את המיקום שלו בהתבסס על המדידות החדשות. לשם כך, הוא משתמש במטריצה שנקראת "מטריצת קלמן". המטריצה הזו קובעת כמה להאמין למדידות החדשות ביחס לתחזיות הקודמות. אם המדידות מאוד מדויקות, המטריצה הזו תיתן להן משקל גבוה יותר בעדכון המיקום.

עדכון הביטחון: מטריצת הקוריאציה

לבסוף, הרובוט מעדכן את מידת הביטחון שלו במיקום החדש. זה נעשה על ידי עדכון "מטריצת הקוריאציה" בהתבסס על המדידות החדשות והתחזיות הקודמות. התהליך הזה עוזר לרובוט להיות בטוח יותר במיקום שלו בכל שלב של הדרך.

איך כל זה עובד יחד?

השלבים האלה חוזרים על עצמם בכל רגע שהרובוט זז ומקבל מדידות חדשות. בכל פעם שהרובוט מנבא את המיקום הבא, משווה אותו למדידות, ומעדכן את המיקום

והביטחון שלו. כך, בעזרת המטריצות, הרובוט מצליח להבין את מיקומו בסביבה בצורה מדויקת ולפעול בהתאם.

באופן זה, השימוש במטריצות באלגוריתם EKF-SLAM מאפשר לרובוט לנווט בצורה בטוחה ואמינה, ללמוד מהמידע החדש ולהגיב לשינויים בסביבה בצורה מתמדת.

לאחר שסקרנו את התהליך המפורט והמתמטי של EKF-SLAM והבנו כיצד האלגוריתם הזה מאפשר לרובוטים למפות את סביבתם ולנווט בה בצורה מדויקת, ניתן להמשיך ולחקור אלגוריתמים נוספים שמאפשרים לרובוטים להתמודד עם משימות מורכבות יותר במצבים משתנים. אחד האלגוריתמים המתקדמים ביותר בתחום זה הוא Graph SLAM.

בעוד EKF-SLAM מתמקד בשילוב מידע חיישנים ותיקון שגיאות בזמן אמת, Graph SLAM מציע גישה רחבה ומורכבת יותר. Graph SLAM לא רק מבצע מיפוי ומיקום בו-זמנית, אלא גם עוסק בבניית גרף המתאר את המיקומים והקשרים בין הנקודות שהרובוט עבר בהן. הדבר מאפשר לרובוט להתמודד עם בעיות מורכבות של ניהול מדידות מרובות לאורך זמן, תוך שמירה על דיוק גבוה.

Graph SLAM מצטיין בכך שהוא מאפשר לא רק לתקן שגיאות מדידה מצטברות, אלא גם לזהות ולסגור לולאות – תהליך שבו הרובוט מזהה שהוא חזר לנקודה שבה כבר היה בעבר ומעדכן את המפה והמסלול בהתאם. תכונה זו חשובה במיוחד בסביבות דינמיות שבהן דיוק המיקום והמפה קריטיים להצלחת המשימה.

כעת, נעבור לדיון מעמיק יותר על Graph SLAM, נשווה בין שתי השיטות ונדגיש כיצד כל אחת מהן תורמת לשיפור הניווט וההתמצאות של הרובוט בסביבות מורכבות.

Graph-SLAM:

הבנה מתמטית ויישום

מבוא ל-Graph SLAM

Graph SLAM היא שיטה מתקדמת בתחום של SLAM, שמטרתה לאפשר לרובוטים לאתר את מיקומם ולמפות את הסביבה שבה הם נמצאים באותו הזמן. תהליך זה מערב שילוב של מיקום (Localization) ויצירת מפה (Mapping) בזמן אמת. ב-Graph SLAM, מיקומים שונים שהרובוט מבקר בהם ותכונות שהוא מזהה בסביבה מיוצגים כנקודות, או "קודקודים", בתוך מודל גרפי. קישורים, או "קשתות", בין הקודקודים הללו מציינים את הקשרים המרחביים והתנועתיים בין הנקודות, ונוצרים בעזרת מדידות יחסיות, נתונים ממערכות הניווט של הרובוט (אודומטריה),

והתיקונים שמתבצעים כאשר הרובוט זיהה שהוא חזר למקום שכבר נמצא בו בעבר (תהליך המכונה "סגירת לולאה").

עקרונות הפעולה של Graph SLAM

אלגוריתם Graph SLAM מחולק לשני שלבים מרכזיים: בניית הגרף ואופטימיזציה של הגרף.

בניית הגרף ב-Graph SLAM

בניית הגרף ב-Graph SLAM היא שלב מכריע שבו מבנה הנתונים המרכזי של המערכת, הגרף, מורכב ונבנה. הגרף משמש לתיאור מסלול הרובוט ולקשר בין נקודות שונות שהרובוט עבר בהן. ניתן לחלק את התהליך לשני חלקים עיקריים:

1. זיהוי קודקודים:

כל פעם שהרובוט מגיע למיקום חדש או משנה את תנוחתו במרחב, נוצר קודקוד חדש בגרף. קודקוד זה מייצג את המיקום המדויק של הרובוט באותו רגע. דמיינו זאת כאילו הרובוט מצייר נקודה על מפה כל פעם שהוא עוצר או משנה את מיקומו.

2. יצירת קשתות:

לאחר יצירת הקודקודים, השלב הבא הוא לחבר בין הנקודות הללו על ידי קשתות. הקשתות מייצגות את הנתיב שהרובוט עבר מנקודה אחת לשנייה ומשקפות את המרחק והזווית בין הנקודות. הן נוצרות על בסיס מדידות יחסיות שמתקבלות מהאודומטריה של הרובוט, כלומר מעקב אחר המרחק שהרובוט נסע ובאיזו כיוון. בנוסף, הקשתות יכולות להיות מבוססות גם על זיהוי תכונות בסביבה, כמו עצמים מסוימים או סימנים קבועים שהרובוט מזהה שוב ושוב, מה שמאפשר לו לאמת ולעדכן את מיקומו ביחס לנקודות אלו.

לסיכום, בניית הגרף ב-Graph SLAM היא תהליך שבו הרובוט מפתח מפת דרכים של נקודות ומסלולים שהוא עבר, וזה מהווה את הבסיס להבנת המרחב סביבו וניווט מדויק יותר בהמשך.

אופטימיזציה של הגרף

האופטימיזציה מחפשת את ההערכה הסבירה ביותר למצב הרובוט ולמפת העולם, על ידי מיזוג והקטנת שגיאות המדידה המצטברות.

הצגה מתמטית של בעיית האופטימיזציה:

נניח שיש לנו n קודקודים וm קשתות. כל קודקוד i מיצג מצב x_i וכן קשת (i,j) מייצגת מדידה יחסית $z_{i\ j}$ עם מטריצת הקוואריאנס $\pi_{i\ j}$ פונקציית המטרה היא:

$$E(x) = \Omega_{ij} 2 \parallel h(x_i, x_j) - z_{ij} \parallel \sum_{(i,j)\in C}$$

כאשר

C היא קבוצת כל הקשתות בגרף

$\mathrm{h(xi, xj)}$ היא פונקציה לא לינארית המתארת את התחזית של המדידה היחסית בין המצבים.

$\underset{\Omega}{2}||e|| = e^T \Omega \mathrm{e}$ מייצגת את הנורמה עפ"י מטריצת הקוואריאציה.

פתרון הבעיה באמצעות אופטימיזציה לא לינארית

גזירה ראשונית: הגרדיאנט של פונקציית המטרה נתון על ידי:

$$\nabla \mathrm{E(x)} = \nabla \mathrm{h(xi, xj)} \cdot (\mathrm{h(xi, xj)} - \mathrm{zij}) \cdot 2 \cdot \Omega\mathrm{ij} \cdot \sum_{\mathrm{C} \in (\mathrm{i,j})}$$

איפוס הגרדיאנט

נבצע איפוס של הגרדיאנט באמצעות שיטות כמו *Gauss-Newton* או *Levenberg-Marquardt* למציאת הערכה האופטימלית של המצבים:

$$\mathrm{E(xk)} \nabla \cdot \lambda - \mathrm{kx} = \mathrm{k} + 1\mathrm{x}$$

כאשר λ הוא פרמטר הלמידה המתאים.

הסבר לתהליך המתמטי:

בבעיית האופטימיזציה הזו, אנו מנסים למצוא את המיקום המדויק של כל קודקוד (מצב) בגרף כך שיתאים בצורה הטובה ביותר למדידות שנאספו.

1. מדידות יחסיות ומטריצת קוואריאציה: כל קשת בגרף מייצגת מדידה יחסית בין שני מצבים, עם מידע על חוסר הוודאות במדידה הזו (מטריצת הקוואריאציה). המדידה היחסית יכולה להיות, למשל, המרחק בין שני רובוטים או זווית בין שתי נקודות.

2. פונקציית מטרה: פונקציית המטרה *E(x)* נועדה למדוד את השגיאה בין התחזיות שלנו באמצעות הפונקציה *h(x_i, x_j)* לבין המדידות בפועל z_{ij}. הפונקציה כוללת את הנורמה של השגיאה, משוקללת לפי מטריצת הקוואריאציה, כדי להבטיח שמדידות עם אי ודאות גדולה יותר יקבלו משקל מתאים.

3. חישוב הגרדיאנט: הגרדיאנט של פונקציית המטרה מראה לנו את הכיוון שבו השגיאה גדלה או קטנה. על ידי חישוב הגרדיאנט, אנחנו יודעים איך לשנות את המצבים x_i כדי להקטין את השגיאה.

4. איפוס הגרדיאנט: כדי למצוא את המצב האופטימלי, עלינו למצוא נקודה שבה הגרדיאנט הוא אפס (כלומר, השגיאה המינימלית). זה נעשה באמצעות שיטות איטרטיביות כמו *Gauss-Newton* או *Levenberg-Marquardt*, שמעדכנות את המצבים x_i בכל איטרציה, כך שהשגיאה תקטן.

5. איטרציות לשיפור: בתהליך זה, בכל איטרציה, אנו מחשבים את הגרדיאנט ומעדכנים את המיקום של כל קודקוד בהתאם. כך אנו מתקרבים בהדרגה לפתרון שבו השגיאה היא מינימלית.

באמצעות תהליך זה, אנו מצליחים למצוא את המיקומים המדויקים ביותר של הקודקודים, תוך התאמה מרבית למדידות ולמידע הזמין לנו. השיטה מאפשרת לרובוטים להבין את מיקומם בסביבה ולפעול בצורה אופטימלית יותר.

3. עדכון חוזר: נחזור על התהליך עד להתכנסות.

יתרונות ואתגרים ב-Graph SLAM

- יתרונות: עמידות בפני טעויות מצטברות, גישה מדויקת לניהול נתונים מורכבים, ויכולת התאמה לסביבות גדולות.

- אתגרים: דרישות חישוביות גבוהות וצורך בזיהוי ותיקון שגיאות מדידה.

Graph SLAM מהווה אבן פינה בתחום הניווט האוטונומי, מציעה גמישות ודיוק גבוה במיפוי וניהול סביבות מורכבות, תוך ניצול הידע המתמטי והאלגוריתמי.

לאחר שסקרנו את האלגוריתמים EKF-SLAM ו-Graph SLAM, ניתן להצביע על מספר הבדלים מרכזיים בין השיטות הללו:

1. מודל מתמטי ותהליך אופטימיזציה:

EKF-SLAM משתמש במסננים כמו Kalman Filter (EKF) כדי לשלב מדידות בזמן אמת ולעדכן את המיקום והקוואריאציה של הרובוט בצורה רציפה. הוא מבוסס על מודל ליניארי או קרוב לליניארי ומתאים לסביבות שבהן ניתן לבצע עדכונים רציפים ומהירים. מנגד, Graph SLAM בונה גרף המורכב מקודקודים וקשתות המתארים את מסלול הרובוט והמדידות היחסיות בינו לבין עצמים בסביבה. האופטימיזציה של Graph SLAM כוללת פתרון בעיות אופטימיזציה לא ליניאריות ומשתמשת בשיטות כמו Gauss-Newton או Levenberg-Marquardt כדי למזער שגיאות מצטברות ולהשיג דיוק מרבי.

2. ניהול מדידות ותיקון שגיאות:

ב-EKF-SLAM, כל מדידה חדשה מעדכנת את מצב הרובוט ומצמצמת את אי-הוודאות במיקומו. התהליך חוזר על עצמו בצורה רציפה ומידית. לעומת זאת, Graph SLAM מנהל את כל המדידות שנאספו על פני זמן ומבצע תיקון שגיאות מצטברות על ידי זיהוי וסגירת לולאות – כלומר, זיהוי מצבים שבהם הרובוט חזר

לנקודה שבה כבר היה בעבר. תיקון השגיאות ב-Graph SLAM מאפשר לו להתמודד בצורה יעילה יותר עם שגיאות מדידה מצטברות לאורך זמן.

3. דיוק וביצועים:

EKF-SLAM מתאים לסביבות שבהן נדרשים עדכונים מהירים ורציפים, והוא מספק פתרונות די מדויקים במקרים שבהם המודל הליניארי הוא מספיק טוב. עם זאת, הוא עלול להיתקל בקשיים בסביבות מורכבות או לא ליניאריות. Graph SLAM, לעומת זאת, מציע דיוק גבוה יותר במקרים שבהם נדרשת אופטימיזציה על פני מדידות רבות ולאורך זמן. השימוש בגרף מאפשר ל-Graph SLAM לטפל בסביבות מורכבות ולבצע תיקוני שגיאות מצטברות בצורה יעילה יותר.

4. יישומים וסביבות עבודה:

EKF-SLAM מתאים יותר ליישומים בזמן אמת שבהם נדרש עדכון מיקום מהיר ורציף, כמו רובוטים נעים בסביבות פשוטות או חצי-מובנות. Graph SLAM מתאים יותר ליישומים שבהם נדרשת אופטימיזציה מורכבת ומדויקת יותר, כמו רובוטים הפועלים בסביבות משתנות ומורכבות או במקרים שבהם יש צורך בניהול מדידות על פני זמן רב.

לסיכום, שתי השיטות – EKF-SLAM ו-Graph SLAM – מציעות פתרונות שונים לנושא המיפוי והלוקליזציה בו-זמנית של רובוטים. הבחירה בשיטה המתאימה תלויה בסוג המשימה, דרישות הדיוק, והמורכבות של הסביבה שבה פועל הרובוט. בעוד EKF-SLAM מציע פתרון מהיר ורציף, Graph SLAM מספק אופטימיזציה מדויקת ומורכבת יותר, המאפשרת לרובוטים לפעול בסביבות דינמיות ומשתנות בצורה יעילה ובטוחה.

מבוסס על המאמר:

Simultaneous Localization and Mapping (SLAM) for Autonomous Driving: Concept and Analysis

פרק שלישי: RRT

לאחר שחקרנו את האלגוריתם SLAM והבנו כיצד הוא מאפשר לרובוטים למפות ולהתמקם בסביבתם בו-זמנית, נעבור לדון באלגוריתם שממשיך את שרשרת התכנון והניווט האוטונומי - ה-RRT, Rapidly-exploring Random Trees. זהו אלגוריתם שמשמש לתכנון מסלול בסביבות מורכבות ומשתנות, ומבטיח שהרובוט ימצא דרך בטוחה ויעילה להגיע ליעדו.

לרגע נבין היכן בעצם בכל מערכת תכנון הרובוט נמצאים שני אלגוריתמים מרתקים אלו:

בעולם הרובוטיקה האוטונומית, עולם שבו מכונות חכמות מתמודדות עם אתגרי שדה הקרב המודרני. בשדה הקרב הדיגיטלי, שבו חיילי הרובוט משתמשים בבינה מלאכותית כדי לפתור בעיות מורכבות בזירה שאינה נחשבת למשחק ילדים.

הנה עומד לפנינו רובוט שמהווה שיא טכנולוגי, מוכן לבצע משימות בסביבה לא ידועה ולא ניתנת לחיזוי. זו לא רק מכונה - זו מערכת מורכבת שמשלבת חיישנים, עיבוד נתונים ותוכניות פעולה על מנת להבטיח שהוא יעמוד בכל משימה שתוטל עליו.

שלב ראשון:

הרובוט מתחיל את מסעו בזיהוי המרחב שסביבו באמצעות SLAM, טכניקה שמאפשרת לו לפתח מפה בזמן אמת תוך כדי מיקום עצמו במפה שיצר. כל זה נעשה במהירות רבה ובדיוק גבוה, כאשר מכל פינה ופינה יכול לצוץ מכשול חדש או אויב לא צפוי.

ברגע שהמפה מוכנה והרובוט יודע את מיקומו, הוא עובר לשלב הבא של תכנון המסלול – מהי הדרך הטובה ביותר מהנקודה הנוכחית לנקודת היעד. כאן נכנס לתמונה אלגוריתם ה-RRT, שמשמש ככלי לחיפוש מהיר ויעיל של מסלול בשטח המורכב שנפרש לפניו.

דמיינו שהרובוט ניצב במרכז שדה קרב שוקק, כאשר מכשולים ומטרות משתנות כל הזמן. כיצד הוא יכול לנווט במציאות כזו? ה-RRT פועל בתור חוקר, דוחף ענפים וירטואליים לכל עבר, מתרחב באופן אקראי לכיוון המטרות. המסלולים שהוא יוצר מתפתחים כאילו הם חלק ממוח מחשב ענק, כל נתיב מתווה תוך שיקול דעת מלא והתחשבות בכל המשתנים.

אך זיהוי המסלול הינו רק חלק מהאתגר. ברגע שהמסלול נקבע, על הרובוט להתמודד עם דינמיקה של הסביבה - שינויים פתאומיים בתוואי השטח, הופעת מכשולים חדשים והצורך לעדכן בזמן אמת את התנועה שלו כדי להתגבר עליהם. שילוב של עקרונות קינמטיים ודינמיים מאפשר לרובוט לשמר תנועה חלקה ומדויקת, מתמקד במטרה תוך שהוא מבצע תיקונים קלים על המסלול בהתאם למציאות השוטפת. – שלא לבצע קפיצות מכאן ולכאן וצעדים שבן אנוש לא היה בוחר לבצע –

ע"י הקינמטיקה והדינמיקה הרובוט בוחר את הצעד הבא בצורה אנושית וחכמה – מסלול חלק ומדויק.

הרובוטים האוטונומיים שלנו אינם רק מכונות - הם חוקרים, לוחמים, מתכננים ומבצעים, מפעילים מערכות מורכבות בשליטה עצמית כדי להשיג את המטרות הקשות ביותר או המסוכנות.

אלגוריתם RRT - פירוט:

אלגוריתם RRT פותח למטרת תכנון מסלול במרחבים גדולים ומורכבים עם מכשולים. הוא מאוד יעיל בגישה לאזורים שלא נחקרו קודם לכן ומצליח למצוא מסלול במהירות גבוהה גם במרחבים גדולים.

כיצד עובד ה-RRT?

1. אתחול: התהליך מתחיל מנקודת התחלה במרחב החיפוש - שתיכלל בעץ (המיוצג כגרף). זוהי נקודת המוצא שממנה ה"עץ יתחיל לצמוח".

2. איטרציה:

- בכל צעד, נבחרת נקודה אקראית במרחב - X.
- מוצאים את הנקודה הקרובה ביותר לX שכבר נמצאת בעץ.
- מנסים להרחיב את העץ מהנקודה הקרובה ביותר לעבר הנקודה האקראית, בדרך כלל במרחק מוגדר (כלומר, מרחק קטן בכיוון הנקודה האקראית).
- אם המרחק החדש אינו נתקל במכשולים, הוא מתווסף לעץ.

3. לולאה: התהליך נמשך עד שהעץ מגיע לנקודת הסיום או עד שנגמר הזמן/משאבים המוקצים לחיפוש.

4. מסלול סופי: בסיום התהליך את המסלול מנקודת ההתחלה לנקודת הסיום ניתן למצוא על ידי צעדים אחורה מנקודת הסיום בחזרה לנקודת ההתחלה בעץ – דרך כל קשת שהוספנו לנקודה הקיימת בעץ לנקודה אקראית חדשה.

דוגמת קוד לאלגוריתם RRT בפייתון:

```
import numpy as np
import matplotlib.pyplot as plt

class RRT:
```

```
    def __init__(self, start, goal, bounds,
obstacle_list, step_size=1.0, max_iter=500):
        # אתחול המשתנים לכל האובייקט
        self.start = start  # נקודת התחלה
        self.goal = goal  # נקודת סיום
        self.bounds = bounds  # גבולות המרחב בו מתבצע החיפוש
        self.obstacle_list = obstacle_list  # רשימת
מכשולים, כל מכשול מיוצג כטופל (x, y, radius)
        self.step_size = step_size  # גודל הצעד המרבי בכל
איטרציה
        self.max_iter = max_iter  # מספר האיטרציות המרבי
        self.node_list = [start]  # רשימה של נקודות שבהן
כבר ביקרו עד כה בעץ

    def plan(self):
        # פונקציה לתכנון המסלול
        for i in range(self.max_iter):
            random_point = self.get_random_point()
            nearest_node =
self.get_nearest_node(random_point)
            new_node = self.steer(nearest_node,
random_point) # יוצר נקודה חדשה בכיוון הנקודה הרנדומלית שנמצאה - ב
לא בדיוק אותה נקודה רנדומלית אלא נקודה חדשה במקסימום המרחק המותר לצעד

            if self.check_collision(new_node): # אם
הנקודה החדשה אינה מתנגשת במכשול כלשהו.
                self.node_list.append(new_node)
                if self.reached_goal(new_node):
                    return
                    self.generate_final_course(len(self
                    .node_list) - 1) # הנקודה החדשה שווה
                    לנקודת המטרה.

        return None  # אם לא נמצא מסלול None החזרת

    def get_random_point(self):
        # יצירת נקודה אקראית בתוך גבולות המרחב
        return (np.random.uniform(self.bounds[0],
self.bounds[1]),
                np.random.uniform(self.bounds[2],
self.bounds[3])) # בחירת נקודה אקראית בגבולות השטח המותר
לרובוט. ב השדה.

    def get_nearest_node(self, random_point):
        # מציאת הנקודה הקרובה ביותר בעץ לנקודה האקראית
```

```
        distances = [(node[0] - random_point[0])**2 +
(node[1] - random_point[1])**2 for node in
self.node_list]
        nearest_index =
distances.index(min(distances))
        return self.node_list[nearest_index]

    def steer(self, nearest_node, random_point):
        # יצירת נקודה חדשה לכיוון הנקודה האקראית מהנקודה הקרובה ביותר
        theta = np.arctan2(random_point[1] -
nearest_node[1], random_point[0] - nearest_node[0])
        new_node = (nearest_node[0] + self.step_size *
np.cos(theta),
                    nearest_node[1] + self.step_size *
np.sin(theta))
        return new_node

    def check_collision(self, node):
        # בדיקת התנגשות עם מכשולים
        for (ox, oy, size) in self.obstacle_list:
            if (node[0] - ox)**2 + (node[1] - oy)**2
<= size**2:
                return False  # יש התנגשות
        return True  # אין התנגשות

    def reached_goal(self, node):
        # בדיקה האם הנקודה החדשה היא נקודת הסיום
        if (node[0] - self.goal[0])**2 + (node[1] -
self.goal[1])**2 < self.step_size**2: #במרחק הקטן מצעד מ
אל היעד
            return True
        return False

    def generate_final_course(self, goal_index):
        # יצירת המסלול הסופי לאחר שהגענו ליעד
        path = [self.goal]
        while self.node_list[goal_index] !=
self.start:
            node = self.node_list[goal_index]
            path.append(node)
            goal_index = self.node_list.index(node) -
1
        path.append(self.start)
        return path

# דוגמת שימוש
```

```
bounds = (0, 100, 0, 100)  # x_min, x_max, y_min,
y_max - 100,100 ל0,0מ
start = (0, 0)
goal = (100, 100)
obstacles = [(50, 50, 10), (80, 80, 5)]  # מכשולים במרחב
rrt = RRT(start, goal, bounds,
obstacle_list=obstacles, step_size=2.0, max_iter=1000)
path = rrt.plan()

# ציור המסלולעִם המכשולים
if path is not None:
    plt.plot([x for (x, y) in path], [y for (x, y) in
path], '-o')
    for (ox, oy, size) in obstacles:
        circle = plt.Circle((ox, oy), size, color='r',
fill=True)
        plt.gca().add_artist(circle)
    plt.plot(start[0], start[1], 'ro')  # נקודת התחלה
    plt.plot(goal[0], goal[1], 'go')  # נקודתעִיום
    plt.grid(True)
    plt.show()
```

ניתן לקרוא את האלגוריתם וההערות ולהבין את סדר הפעולה של האלגוריתם כדי למצוא מסלול מנקודת היציאה אל נקודת המטרה תוך הימנעות מכניסה לשטחים אסורים – למכשולים. כאשר RRT מציע יעילות נדירה במשימת תכנון המסלול בסביבות דינמיות ומורכבות, בהן נדרשת גמישות רבה ותגובה מהירה לשינויים בסביבה. יתרונו המובהק של RRT בולט במיוחד בהקשר של רובוטיקה אוטונומית, שם המכונות נתקלות במכשולים בלתי צפויים. באמצעות פיתוח מהיר ויעיל של עץ חיפוש המתרחב כלפי כל כיווני המרחב, RRT מסוגל לחקור אזורים חדשים במהירות רבה, תוך כדי שמירה על מציאת מסלולים אופטימליים ובטוחים לרובוט. לא בכל איטרציה בה מוסיפים נקודה חדשה לעץ, הרובוט פוסע את הצעד הזה בפועל. הנקודה החדשה נבדקת תחילה אם היא נקיה מהתנגשויות. רק אם היא נמצאת חופשית ממכשולים ותואמת לקריטריונים נוספים (כגון קירבה למסלול המתוכנן – לא נותח בקוד שהוצג), היא תתווסף למסלול הסופי.

האלגוריתם מוכיח יעילות גבוהה ביותר בתרחישים בהם זמן התגובה הוא קריטי. למשל, ברכבים אוטונומיים הנוסעים בדרכים ציבוריות, יש צורך במערכת שמסוגלת לעבד מידע ולהגיב במהירות להתפתחויות בסביבה כמו שינויים בתנועה, הופעת מכשולים או תנאי דרך משתנים. RRT מספק פתרון בעל יכולת חיפוש מהירה שמתאימה לדינמיקה המתמדת של סביבות אלו, מה שהופך אותו לכלי חיוני מאוד בתחום הרובוטיקה האוטונומית בכלל וברובוטיקה מלחמתית בפרט.

שינויים דינאמיים בשדה הקרב:

האלגוריתם נחשב לאחד הכלים הבסיסיים בתחום תכנון מסלולים בסביבות דינמיות. נבחן איך הוא עובד בפועל בסביבה שבה יש שינויים דינמיים ומה התהליך שמתרחש כשמופיע מכשול חדש.

תהליך העבודה של RRT בסביבה דינמית

1. איתור והתמודדות עם שינויים:

בסביבה דינמית, שבה מכשולים עלולים להופיע בפתאומיות, RRT נדרש להתמודד עם השינויים בזמן אמת. כשנתקל במכשול חדש או שינוי בתצורת המכשולים, האלגוריתם יעצור את הפעולה ויחשב מחדש נתיב מהנקודה הנוכחית של הרובוט אל היעד.

2. טיפול במכשולים חדשים:

כאשר מתווסף מכשול חדש שמשנה את המסלול המתוכנן, RRT יכול להתחיל בפעולת חיפוש חדשה מהנקודה הנוכחית של הרובוט או לבצע תיקון למסלול על ידי מציאת נתיב עקיף שמתחמק מהמכשול החדש.

3. גילוי מכשולים חדשים:

כיצד בכלל מזהה הרובוט שנוספו מכשולים חדשים?

השימוש בחיישנים כמו LIDAR מאפשר לרובוט לזהות מכשולים חדשים בזמן אמת. LIDAR שולח קרני לייזר שחוזרות חזרה אל החיישן כאשר הן פוגעות במכשול. זמן ההחזרה מאפשר למערכת להעריך את המרחק למכשול ולזהות את מיקומו בסביבה. בעזרת המידע הזה, הרובוט יכול לעדכן את מפת המכשולים ולבצע התאמות במסלולו.

סיכום RRT (Rapidly-exploring Random Trees):

אלגוריתם RRT הוא כלי חזק בתחום תכנון המסלול, שנועד להתמודד עם סביבות גדולות ומורכבות שבהן קיימים מכשולים ואתגרים דינמיים. האלגוריתם מאופיין ביעילותו בחיפוש וגילוי של אזורים חדשים במרחב, תוך כדי בניית עץ שמתפתח מהר מאוד ומרחיב את הידע על הסביבה.

נקודות עיקריות שנדונו:

1. **תהליך האלגוריתם:** RRT מתחיל מנקודת התחלה ומתפשט באופן אקראי במרחב, תוך כדי שמירה על חיבור צעד לנקודה הקרובה ביותר בעץ. בכל איטרציה, הוא בוחן אפשרות להוספת נקודה חדשה שלא מתנגשת במכשולים.

2. **התמודדות עם סביבה דינמית:** במקרה של שינויים בסביבה, כמו הופעת מכשולים חדשים, RRT מסוגל להגיב במהירות ולעדכן את המסלול על ידי חישוב מחדש של העץ מהנקודה הנוכחית.

3. **גילוי מכשולים:** חיישנים כמו LIDAR משמשים לזיהוי מכשולים חדשים בזמן אמת, מה שמאפשר ל-RRT להתאים את עצמו לשינויים בסביבה.

מתוך מאמר ניתוח והצגת אלגוריתם RRT ע"י ממציאו Steven M. LaValle:
.The Rapidly-Exploring Random Tree (RRT) Page by Steven M. LaValle

אעצור לרגע כדי לבחון את השאלה המתבקשת. כיצד לאחר גילוי המערכת מהו מיקום הרובוט ולאחר קביעת המסלול. הרובוט ידע למעשה מה הצומת הבאה שהוא צריך להיות בה ובאיזה כיוון. אך כיצד הוא ידע מה עליו לשנות בכל אחד מהמפרקים שלו מהמקום בו הוא נמצא עכשיו כדי שהוא יגיע לנקודה שאליה אמור להגיע?

התשובה לכך היא **הקינמטיקה ההפוכה:**

פרק רביעי: קינמטיקה הפוכה:

הרובוט צריך למצוא את סט הזוויות טטה בו הוא מסובב את המפרקים על מנת שהחלק הפועל יגיע למקומו. כלומר למצוא את הקונפיגורציה המתאימה כך ש y(teta) = Y כאשר y היא פונקציה למציאת קינמטיקה ישירה. – בהינתן ווקטור הזוויות טטה – זווית הסיבוב המדויקת של כל מפרק במערכת הרובוטית מחזירה את מיקום הEE – מיקום החלק הפועל של הרובוט. טטה היא וקטור הזוויות שנמצא בקינמטיקה ההפוכה וY הוא מיקום הEE הרצוי. זהו למעשה אלגוריתם שניתן לביצוע ע"י חישובים אנליטיים – באיזו זווית יש לסובב כל מפרק על מנת שהחלק הפועל ימצא במיקום שקבעה לו מערכת התנועה. או לחילופין – מסתבר שבעבור מערכות רובוטיות סבוכות משמעותית כמו המערכות שנציג בהמשך - פתרונות אנליטיים למציאת הקינמטיקה הישרה סבוכים מידי ולעיתים לא אפשריים בכלל. ולכן ישנם אלגוריתמים אחרים – נומריים למציאת הקינמטיקה ההפוכה כגון אלגוריתם הקירוב ניוטון רפסון המקבל משוב מהמערכת הרובוטית לצורך מציאת וקטור הזוויות של המפרקים. – הקונפיגורציה המתאימה למיקום החלק הפועל מתוך מרחב הקונפיגורציות של הרובוט – תוך דיוק עד לקבוע אפסילון מזערי שנקבע מראש ע"י מתכנתי הרובוט. – בד"כ גודלו עפ"י רמת הדיוק של המערכת הרובוטית. שכן אם נקבע את אפסילון להיות קטן מרמת הדיוק של הרובוט החיפוש אחר הקונפיגורציה המתאימה יהיה

אינסופי מכיוון שהEE לעולם לא יגיע למרחק אפסילון מהמטרה שנקבעה לו ותמיד יפספס אותה.

אלגוריתם ניוטון רפסון ר Modern Robotics של Lynch, Kevin Cambridge.

ניחוש קונפיגורציה אפשרית ראשונית - זוויות סיבוב אפשריות לכל המפרקים.

המיקום הרצוי של הEE.

Equation (6.6) suggests u e Newton–Raphson iterative algorithm for finding θ_d:

(a) **Initialization**: Given $x_d \in \mathbb{R}^m$ and an initial guess $\theta^0 \in \mathbb{R}^n$, set $i = 0$.

(b) Set $e = x_d - f(\theta^i)$. While $\|e\| > \epsilon$ for some small ϵ:

- Set $\theta^{i+1} = \theta^i + J^\dagger(\theta^i)e$.
- Increment i.

השגיאה בין המקום אליו יגיע הEE בעבור קונפיגורצית הניחוש הנוכחית לבין המקום הרצוי Xd.

שגיאה עד לגודל קבוע אפסילון שנקבע.

יעקוביאן המערכת.

חישוב הקונפיגורציה המדויקת יותר

הניחוש הקודם.

סיום פרק רובוטי ניווט אוטונומי:

כאשר אנו מסכמים את התקדמות הרובוטיקה האוטונומית, אנו עדים לפריצות דרך רבות המבטיחות עתיד בהיר. רובוטים אוטונומיים כבר משפיעים באופן משמעותי על תחומים רבים, והם משפרים את אופן ביצוע המשימות תוך יצירת חדשנות טכנולוגית. התפתחות זו מתבטאת ביכולת לתכנן ולבצע משימות בצורה יעילה וחכמה, הודות לשילוב של בינה מלאכותית ואלגוריתמים מתקדמים כמו SLAM ו-RRT. היכולת להתמודד עם מכשולים ולתכנן מסלולים בסביבות משתנות פותחת את הדרך לשימושים חדשים וחשובים בעתיד.

פרק חמישי: פרק סיום:

במהלך הסמינר, התעמקנו בטכנולוגיות מתקדמות ברובוטיקה אוטונומית, עם דגש על שילוב אלגוריתמי SLAM ו-RRT לתכנון ניווט מדויק ויעיל בשדה הקרב.
האלגוריתמים הללו מהווים את הבסיס ליכולות האוטונומיות של הרובוטים, המאפשרים להם לבצע משימות מורכבות במצבים משתנים ודינמיים.

תהליך הניווט האוטונומי: שילוב של מיפוי, מיקום ותכנון מסלול

1. מיפוי ומיקום - SLAM:

- האלגוריתם SLAM (Simultaneous Localization and Mapping) הוא כלי מרכזי המאפשר לרובוט למפות את סביבתו ולמקם את עצמו במפה זו בו-זמנית. באמצעות חיישנים כמו LiDAR, מצלמות ו-IMU, הרובוט אוסף נתונים מהסביבה, מזהה נקודות ייחוס, ומעדכן את מיקומו והמפה באופן רציף. השימוש באלגוריתמים כמו EKF-SLAM ו-Graph SLAM מבטיח שהרובוט ידע בכל רגע נתון היכן הוא נמצא ביחס לסביבתו.

2. מציאת מסלול – RRT:

- לאחר שהרובוט ממפה את סביבתו וממקם את עצמו במפה, מגיע שלב תכנון המסלול. האלגוריתם RRT (Rapidly-exploring Random Tree) משמש לתכנון מסלול יעיל ובטוח מהנקודה הנוכחית של הרובוט ליעדו. האלגוריתם פועל על ידי יצירת עץ של נקודות אקראיות במרחב והרחבתו לעבר נקודות יעד, תוך הימנעות ממכשולים ומציאת מסלול אופטימלי.

3. קינמטיקה הפוכה:

- לאחר תכנון המסלול, תת-מערכת הקינמטיקה ההפוכה ממירה את המסלול לצעדים פיזיים. האלגוריתם מחשב את הזוויות הנדרשות לכל מפרק במערכת הרובוטית כדי לבצע את המסלול המתוכנן בדיוק מרבי. כך, הרובוט מבצע את התנועות הנדרשות בצורה חלקה ומדויקת.

4. אותות חשמליים למפרקים:

- לבסוף, מערכת הבקרה של הרובוט שולחת אותות חשמליים למנועים ולמפרקים כדי לבצע את התנועות הנדרשות. תהליך זה מבטיח שהרובוט פועל בהתאם לתכנון המסלול וההוראות המדויקות שהתקבלו מהאלגוריתמים הקודמים.

הפוטנציאל המהפכני של הרובוטיקה האוטונומית

הסמינר חשף את הפוטנציאל הגדול של רובוטיקה אוטונומית בלחימה, לצד האתגרים המוסריים והטכנולוגיים הכרוכים בה. למרות שמאמרים מסוימים מצביעים על מגבלות הטכנולוגיה, אחרים רואים בה פוטנציאל מהפכני שישנה את אופי הלחימה בצורה דרמטית. הבנה מעמיקה ושילוב חכם של אלגוריתמים כמו SLAM ו-RRT הם המפתח לפיתוח מערכות רובוטיות אוטונומיות מתקדמות ואמינות בעתיד.

סיכום

העבודה הסמינריונית הזו מדגימה כיצד שילוב של טכנולוגיות מתקדמות ברובוטיקה אוטונומית יכול לשפר את היכולות הלחימה בשדה הקרב המודרני. האלגוריתמים SLAM ו-RRT, בשילוב עם טכניקות קינמטיקה הפוכה ובקרת תנועה מתקדמת, מאפשרים לרובוטים לפעול בצורה עצמאית ומדויקת בסביבות מורכבות. התהליך המפורט שנחקר במאמר מציב את הבסיס להבנה כיצד רובוטים אוטונומיים יכולים להפוך לכלי עיקרי בלחימה העתידית, תוך שמירה על חיי אדם ושיפור היעילות הקרבית.

Bibliography

.1

Combat Injury Profile in Urban:

Satanovsky, A., Gilor, Y., Benov, A., Chen, J., Shlaifer, A., Talmy, T., Radomislensky, I., Siman-Tov, M., Peleg, K., Weil, Y. A., & Eisenkraft, A. (2022). Combat Injury Profile in Urban Warfare. *Military Medicine*, usac366. https://doi.org/10.1093/milmed/usac366

.2

A one decade survey of autonomous mobile robot systems:

Noor Abdul, K. Z., & Al-Araji, A. (2021). A one decade survey of autonomous mobile robot systems. *International Journal of Electrical and Computer Engineering*, 11(6), 4891-4906. https://doi.org/10.11591/ijece.v11i6.pp4891-4906. Available at: https://www.proquest.com/docview/2661969448?accountid=12994

.3

Simultaneous Localization and Mapping (SLAM) for Autonomous Driving: Concept and Analysis:

Zheng, S., Wang, J., Rizos, C., Ding, W., & El-Mowafy, A. (2023, April 20). *Simultaneous Localization and Mapping (SLAM) for Autonomous Driving: Concept and Analysis.* Remote Sens. 2023, 15, 1156. https://doi.org/10.3390/rs15041156

.4

The Rapidly-Exploring Random Tree (RRT) Page: LaValle, S. M. (n.d.). The Rapidly-Exploring Random Tree (RRT) Page. Retrieved from https://lavalle.pl/rrt/

.5

No Man's War: Will Robots Fight the Battles of the Future?:

Conrad, J. (2019). No Man's War: Will Robots Fight the Battles of the Future? *SAIS Review of International Affairs*, 39(1), 103-106. https://doi.org/10.1353/sais.2019.0009. Available at the Open University library.

.6

Robot wars: Autonomous drone swarms and the battlefield of the future

King, A. (2024). Robot wars: Autonomous drone swarms and the battlefield of the future. *Journal of Strategic Studies*, 47(2), 185-213. https://doi.org/10.1080/01402390.2024.2302585. Available at the Open University library.

.7

Modern Robotics

Lynch, K. M., & Park, F. C. (2019). Modern Robotics: Mechanics, Planning, and Control. Cambridge University. Available at The Open University Of Israel Library.