האוניברסיטה הפתוחה
המחלקה למתמטיקה ומדעי המחשב

סימולטור למחקר תנועת רכב אוטונומי – ניהול התנגשות תוך שמירת נתיב קצר

פרויקט גמר מוגש כחלק מדרישות התואר
לקראת תואר שני M.Sc במדעי המחשב
האוניברסיטה הפתוחה
החטיבה למדעי המחשב

נכתב על ידי
אמנון יעקב

הוכן תחת ההנחיה של דר׳ ליאוניד ברנבוים

יולי 2018

הקדמה

תחום הרכב האוטונומי נחשב כתחום חדשני ומרכזי בתעשייה ובאקדמיה בימים אלו. תחום זה הינו רחב ובעל אתגרי מחקר שונים ורחבים החל מחקירת סנסורים שונים לבניית תמונת מציאות, אלגוריתמים לניהול מסלול הרכב ועד ניהול תנועת הרכב ברמה הנמוכה ביותר.

בפרויקט זה אנו נציג פיתוח סימולטור שיהווה תשתית סימולטיבית לבדיקת אלגוריתם למניעת התנגשות תוך שמירת הנתיב הקצר עבור רכב אוטונומי. הסימולטור פותח כחלק הנדסי משלים לעבודת גמר (שמתבצעת במקביל על ידי מפתחי הפרויקט) החוקרת את תחום האלגוריתמיקה למניעת התנגשות בהתייחסות לשמירת נתיב קצר אל מול יעד מוגדר. הסימולטור מאפשר ממשק נוח לכתיבת אלגוריתם מניעת התנגשות למשתמש המעוניין לחקור התנהגות של סימולטור למניעת התנגשות תחת תרחישים מסוימים.

הסימולטור פותח בשפת ג׳אווה על גבי מנוע גרפי (Sinalgo, תוכנת קוד פתוח [1]). פיתוח הסימולטור דרש פיתוח בשני וקטורים מרכזיים – התאמת תשתית המנוע הגרפי לתמיכה ביכולות רכב אוטונומי ויכולות תחקור ובמקביל פיתוח מעטפת שלמה להטמעה של אלגוריתם מניעת התנגשות לרכב תוך התייחסות לתרחישים שונים יחד עם הטמעה של אלגוריתם בסיסי להוכחת הסימולטור.

תוכן

הקדמה 2

מבוא 5

רקע 6

ארכיטקטורת הרכב האוטונומי 6

אלגוריתמי תכנון נתיב ומניעת תאונה 8

קונפיגורציית מרחב הבעיה 8

רקע לסימולטור 9

יעוד הסימולטור 9

מוטיבציה 9

תשתית הסימולטור 9

תיאור הפרויקט 9

תיאור טכני 9

System view – רכיבי התשתית 10

System view – רכב האוטונומי - UGV 12

Operation view – תהליכי המערכת 15

כלים/ספריות שהשתמשנו בבניית הסימולטור 18

פעולות ותצוגת משתמש 18

כללי 18

תפריטים 19

בקרת תרחיש ומסך תמונת מצב 20

הרצת תרחיש 21

ביצועים 23

כללי 23

דוגמת הרצה 23

חישוב הנתיב קצר 26

גודל תמונה 27

מספר כלי הרכב 28

סיכום 29

מה למדתי מהתהליך 29

האתגרים בפרויקט 29
אפשרויות להמשך פיתוח 29
ביבליוגרפיה 31
נספח א' - אלגוריתם A* 32

מבוא

אם נבקר ב"עמק הסיליקון" (או באריזונה, פלורידה ועוד) בימים אלו נוכל לראות מספר לא קטן של רכבים ללא נהג נעים בכבישים בתהליכי ניסוי ואבלואציה כאלה ואחרים. המרוץ התעשייתי יחד עם ההבשלה של המחקר האקדמי נמצא בשיאו תחת הבנה שרכב אוטונומי יהווה את אחת מפריצות הדרך המרכזיות בשנים הקרובות. כמעט כל חברות הטכנולוגיה הגדולות יחד עם חברות הרכב עוסקות בתחום ומנהלות מחקר מתקדם בנושא (google, uber, GM, etc) כאשר לא בהכרח כל המחקרים זהים ולעיתים מוגדר יעד מחקרי שונה בהתאם לצורך . מטרת העל ברורה – רכב ללא נהג המסיע נוסעים ליעדם בצורה בטוחה, יעילה ובאיכות נסיעה לא פחותה מזו של נהג אנושי.

איור 1 פיזור חברות הרכב בתהליך הפיתוח

המוטיבציה לפיתוח רכב אוטונומי נובעת מכמה וקטורים ובראשם שיפור הבטיחות ושיפור ביעילות ניהול הזמן של הנהגים. כמעט 1.3 מיליון אנשים נהרגים כל שנה מתאונות דרכים ועוד רבים נפצעים "בדרך". אנו מצפים מרכב אוטונומי להוריד נתונים אלו לאפס ע"י נטרול הגורם האנושי (עייפות, נהיגה בשכרות ועוד) ובנוסף ע"י שיפור ניהוג הרכב ותגובה מהירה להפתעות שונות וטעויות נהיגה של בן אנוש. מעבר לאספקט הבטיחותי (שהוא לעצמו מהווה מוטיבציה מספקת לרכב האוטונומי) אנו מצפים פשוט

לשיפור באורחות החיים, ניצול זמן יותר טוב ע״י עבודה או מנוחה ברכב. ניהול דרך אפקטיבי יותר ומבוסס אוטומציה ועוד.

תחום התנועה ברכב האוטונומי מבוסס על מחקרים והדגמות מעולם הרובוטיקה תחת התחום של ניהול תנועה במרחב מציאותי, כלומר, הרובוט הנייד מכיר את המפה של המרחב שסביבו אך אינו מכיר בהכרח את כל המכשולים ובוודאי לא ניתן לסמוך על הכרות מקדימה זו. תחת הגדרה זו ישנם מחקרים המציגים פתרונות שונים ליכולת של הרובוט לנוע במרחב. אחד הווקטורים המרכזיים הינו תכנון הנתיב הקצר ביותר ליעד כלשהו בהתבסס על מפה ידועה ומכשולים שקיימים בדרך חתחת תרחישים שונים, החל מידע מושלם של הרובוט על הסביבה ועד ידע מינימלי והתמקדות בתרחיש בו הרובוט מגלה מכשולים ומשנה את הנתיב בזמן אמת. קיימים אלגוריתמים שונים לביצוע משימות אלו כדוגמת - visibility graph, cell decomposition and so on או אלגוריתמים אבולוציוניים ייעודיים וניתן למצוא עבודות רבות בנושא ([2,3,4,5]). כדי ״להבין״ את הסביבה הרובוט (או הרכב האוטונומי) משתמש במערך סנסורים מסוגים שוני המותכים יחד ומאפשרים לרובוט לקבל תמונה טובה על סביבתו הקרובה החל במרחקים של מכשולים שונים, סימונים שונים ועד רובוטים נעים אחרים או כלים מאוישים. בתרחישים מסוימים רובוטים נעים אלו יכולים לתקשר ביניהם ולהעביר מידע על מיקומם וכדו׳ ובכך לייצר תנועה משולבת טובה יותר.

הגישה המרכזית לתכנון נתיב קצר ליעד מבוססת על תכנון נתיב לפני תחילת הנסיעה, על פי המידע הקיים בשלב זה, ובהמשך עדכון הנתיב בהתאם למכשולים החדשים שהתגלו ולא היו מוכרים בשלב התכנון הראשוני. שלב מניעת התאונה (collision avoidance) מהווה שלב קריטי לפני תכנון נתיב מחדש. הרכב נדרש קודם כל להימנע מתאונה ובמקביל לספק פתרון אופטימלי המהווה את הנתיב הקצר להמשך הדרך. הפתרון נדרש להיות ״שלם״ בכך שהרכב לא מתנגש במכשול ומתכנן נתיב מגיע ליעד בצורה בטוחה. אופטימליות הפתרון דורשת הגדרה מורכבת יותר ובכך עוסקים המחקרים שמשתמשים בסימולטור שבנינו. אלגוריתם למניעת תאונה מניח שהרכב מקבל ובונה תמונת עולם שלמה ונכונה ושניתן מבחינה פיזיקלית להוציא לפועל את פקודות האלגוריתם.

הסימולטור שבנינו מבוסס על הנחות אלו ובכך מאפשר לבחון את מניעת ההתנגשות תוך התייחסות לנתיב קצר בצורה מבודדת.

המסמך מחולק לשלושה חלקים....

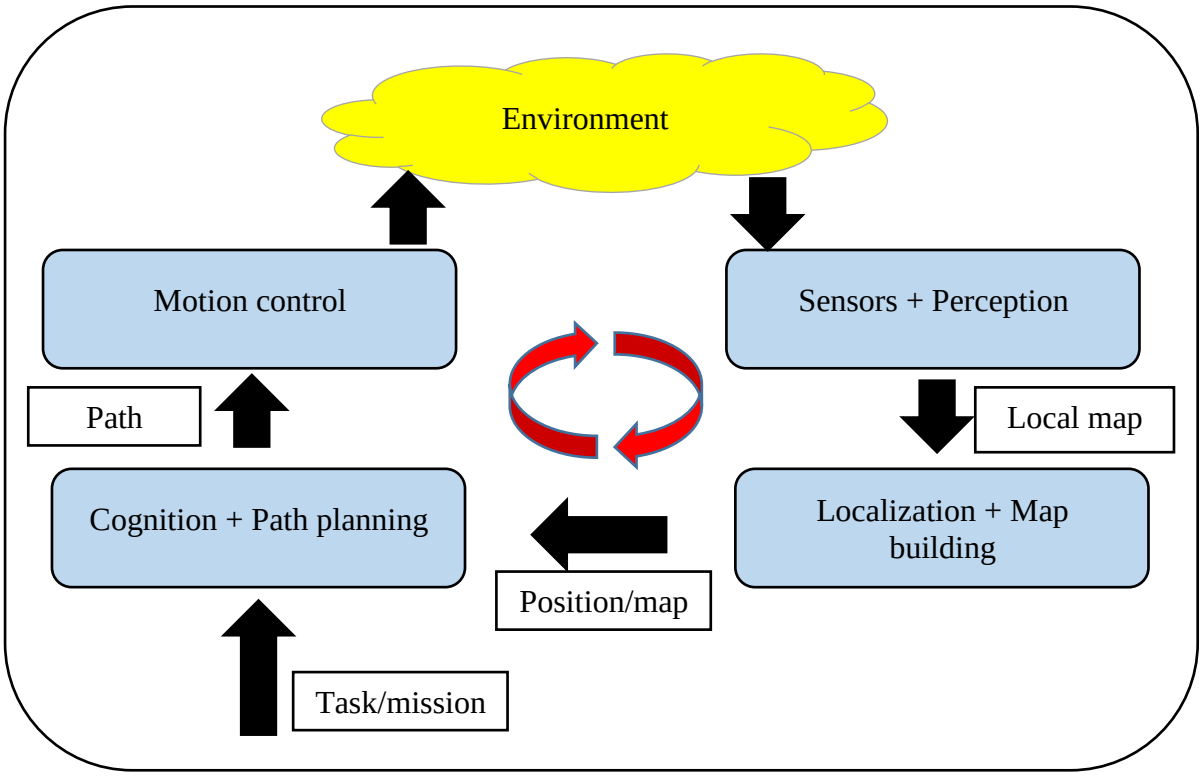
מניעת התנגשות ונתיב קצר

רקע

ארכיטקטורת הרכב האוטונומי

כפי שהזכרנו מעלה , הרכב האוטונומי מבצע את משימת הניווט מקצה לקצה על בסיס תהליכי המשנה הבאים : איסוף והיתוך מידע סנסורי, עיבוד המידע הסנסורי ובניית מפת

מציאות שלמה, הבנת תמונת המצב, ניתוח משמעויות וחיוב המשך המסלול ולבסוף הפקת פקודות ניהוג מתאימות.



איור 2 מבנה בסיסי של פעולת הרכב האוטונומי

סנסורים ותפיסה מרחבית – מבוססי סוגים שונים של סנסורים, החל ממצלמות באורכי גל שונים (לזיהוי רמזור, נתיבים ועדו) ועד מכמים מסוגים שונים (מכ״מ לייזר, אלקטרומגנטי ועוד). לכל סנסור יתרונות מסוימים והמגוון מאפשר להשיג תמונה שלמה. אחד האתגרים בנושא הינו היתוך המידע לכדי מיקום מרחבי וזמן זהים אל מול המידע הסנסורי ומטרת התהליך הינו ליצר תמונה מרחבית מדויקת ושלמה ככל האפשר החל בזיהוי המכשולים, מיקום מדויק, מהירויות יחסיות ועוד.

התמצאות ובניית מפה מקומית – לאחר בניית התמונה המרחבית נדרש לשייך אותה על גבי המפה המוכרת ובכך לייצר מפה המכילה את הרכב יחד עם תמונת העולם שמסביבו הדורשת שילוב בין המידע המקדים הקיים במפה לבין המידע החדש שנאסף ע״ב הסנסורים.

עיבוד המידע ובניית נתיב – בתת-תהליך זה הרכב מבוצע החישוב המשמעותי בין משימת הרכב ותמונת המצב הנוכחית. תהליכי תכנון הנתיב והימנעות מתאונה מחושבים בתהליך זה והסימולטור שבנינו מהווה תשתית לסימלוץ התהליך.

ניהול התנועה – לאחר חישוב המשימה של ברכב והנתיב בו הרכב צריך להמשיך תת תהליך זה נדרש לפרוט את המשימה לפקודות ניהוג פיזיקליות – כיוון ומהירות או היגוי ומצערת/בלם. ניהול התנועה צריך לייצר חווית נהיגה אנושית ונעימה. לעיתים המשימה

הנכונה איננה יכולה להתבצע מבחינה פיזיקלית (פנייה חדה מידי, עצירה במקום וכדו׳) והאלגוריתמים של ניהול התנועה צריכים לתמוך בכך [6].

אלגוריתמי תכנון נתיב ומניעת תאונה

כפי שהזכרנו קודם, אתגרי תכנון נתיב ומניעת תאונה משולבים יחד ברכב האוטונומי שכן הרכב אחראי למימוש המשימה בצורה יעילה יחד עם בטיחות הנוסעים ושתי דרישות אלה לעיתים ״מתנגשות״. נכון להדגיש את שני תרחישי קיצון המדגישים נקודות אלו. בהינתן מפת מכשולים ידועה מראש, ללא שינויים בזמ״א, האתגר מתמקד בתכנון נתיב קצר ביותר בלבד. מן צד השני, במידה ולא ידוע דבר על המכשולים במרחב הנתיב הקצר יהיה למעשה קו ישר ליעד ונדרש להפעיל רק אלגוריתם למניעת התנגשות ותכנון הנתיב מחדש יכלול קו ישר ליעד בלבד.
אלגוריתמי תכנון נתיב קצר מבוססים ברובם על דייקסטרא [7] ברמה מסוימת ומשפרים אותו במידה מסוימת על בסיס יוריסטיקה או שיטות נוספות. ניתן להזכיר את אלגוריתמי ה A*, D* [8,9] ועוד המהווים אלגוריתמים מוכרים ומוכחים כשלמים ואופטימליים (תמיד מוצאים את הנתיב הקצר ביותר אם קיים) וברוב המקרים רצים בזמן טוב יותר משמעותית מדייקסטרא.
ניתן להפריד בין אלגוריתמים לתכנון נתיב קצר ביותר אל מול מפה סטטית לעומת חישוב זמ״א לאחר בניית מפה מעודכנת בזמן נסיעה. אלגוריתמים אלו דורשים זמן ריצה real time ובהם נצטרך להגביל את זמן החישוב (מודול תכנון התנועה מצפה לעדכון כ 10 פעמים בשנייה) גם על חשבון אי מציאת פתרון מיטבי. בסימולטור שלנו מימשנו את מציאת הנתיב הראשוני בשלב הסטטי ב A* ואילו עדכון הנתיב בשלב הדינמי מהווה את נושא המחקר וניתן למימוש במסגרת המחקר בסימולטור.
כהשלמה לתכנון הנתיב, ניתן לסקור את האלגוריתמים הידועים שתומכים במניעת תאונה כגון vector field histogram, dynamic windows [10,11] ועוד. אלגוריתמים אלו בעיקר פותרים את בעיית הסריקה של המרחב הקרוב לכלי הרכב ומציאת פתרון לוקאלי נכון ונקי ממכשולים. הם אינם עוסקים במציאת פתרון ביחס לנתיב שכלי הרכב רצה לבצע וההתייחסות למשימה באופן כללי. כהשלמה לפתרונות המקומיים יש להבין שהפתרון הכללי לבעיית התנגשות איננו מוחלט ומהווה מעין סט כללים/מדיניות (policy) שהרכב ישתמש בהם בהינתן סיטואציה מתאימה שכן הרכב נדרש להחליט בתנאי אי ודאות לגבי הפעולה הנכונה מבחינתו כאשר ייתכן ועדיים לא גילה את כל המכשולים יחד עם תנועה לא צפויה של הרכבים האחרים. במרחב בו קיימת תקשורת בין כלי הרכב או שכלל הרכבים בעלי מדיניות זהה ניתן להגדיר מדיניות מרכזית כעקרון מקדים ובכך לצמצם את מרחב האפשרויות של האלגוריתם (חוקי התנועה המוכרים לנו כבר מהווים מדיניות נסיעה מרכזית ברמה מסוימת). בהקשר זה, ניתן לראות עבודות המשתמשות באלגוריתמים ״לומדים״ (גנטיים וכדו׳) הכוללים אלגוריתמיקה ״עמומה״ (fuzzy) להתמודד עם מרחב הפתרונות ואי הודאות המובנית.

קונפיגורציית מרחב הבעיה

מרחב הבעיה ניתן להגדרה על בסיס קונפיגורציות שונות ובאופן ישיר ידרשו אלגוריתמים שונים לכל קונפיגורציה. **תקשורת בין כלי רכב** – במדיה וקיימת יכולת לתקשורת נתונים בין כלי רכב אוטונומיים נוכל להעביר מיקום, נתיב בין כלי רכב ובהמשך, בהינתן ״חיכוך״ בין כלי הרכב הן יוכלו ״לתקשר״ ביניהם ולפתור את הבעיה בצורה אופטימלית יותר לכל אחד. זהו מאפיין חשוב

ומשפיע מהותית על דרגי הפתרון למניעת תאונה בין שני כלי רכב אוטונומיים. מן הסתם, תקשורת זו לא תתאפשר מול רכב מאויש (״ישן״) או כל מכשול אחר. **שליטה מרכזית מול שליטה פרטית** – קיימים מחקרים העוסקים בפתרון תנועה של מספר כלי רכב המנוהלים בשליטה מרכזית ולמעשה מהווים מרחב בעיה בעל ידע מלא על המכשולים והאלגוריתם נדרש לפתור בעיית אופטימיזציה מסוימת. מאפיין זה יוכל להימצא במפעלי תעשיה גדולים או בצי רכב ייעודיים. עבור רכב אוטונומי לפרט שיטה מרכזית אינה רלוונטית. **תנועה מאורגנת** - ניתן להבחין בין מרחב בעיה המכוון לתנועה מאורגנת בכבישים/נתיבים עם כיוון נסיעה מוגדר לעומת מרחב בעיה דו מימדי ללא כבישים בוא אין סדר בתנועה וכל הדרכים להגיע ליעד מותרות.
הסימולטור שבנינו תומך בתנועה לא מאורגנת, שליטה מקומית והתקשורת בין כלי הרכב ניתנת להגדרה בהתאם לצורך (כלומר, תומך בשתי הקונפיגורציות).
המכשולים במרחב ניתנים לחלוקה בין מכשולים סטטיים – מכשולים קבועים שלא יזוזו, ידועים מראש, לעומת, מכשולים דינמיים – מכשולים בעלי מהירות מסוימת ומיקום משתנה כדוגמת כלי רכב אחרים.

רקע לסימולטור

יעוד הסימולטור

הסימולטור שאפיינו ופיתחנו מהווה כלי לבחינת אלגוריתמים למניעת התנגשות תוך התייחסות למשימת הרכב האוטונומי להגיע ליעדו בנתיב מסוים. הסימולטור מציג את הנתיב הקצר ביותר בין כל רכב ליעדו ומאפשר להטמיע אלגוריתם לניהול התנגשויות בין הרכבים וניהול חישוב המשך הנתיב. הסימולטור מציג סטטיסטיקות התומכות במחקר וכן לוגים מפורטים המאפשרים תחקור מסלול הרכב.

מוטיבציה

הסימולטור פותח כמעבדת פיתוח ומחקר לעבודת גמר בנושא הרכב האוטונומי וביצוע מחקר לאפיון אלגוריתם למניעת התנגשות. הסימולטור מאפשר הגדרה אלגוריתמים שונים יחד עם תרחישים משתנים ובכך להשוות את תוצאות התרחיש הן בהיבטי מניעת ההתנגשות והן בהיבט אורך הנתיב שבוצע בפועל.

תשתית הסימולטור

הסימולטור מתבסס על המנוע הגרפי של פרויקט הקוד הפתוח sinalgo המהווה תשתית לבחינה של גרפי תקשורת וניתוב מידע בקונפיגורציות שונות. במסגרת פיתוח הסימולטור ביצענו התאמה של תשתית המנוע יחד עם פיתוח מלא של שכבת הרכב האוטונומי.

תיאור הפרויקט

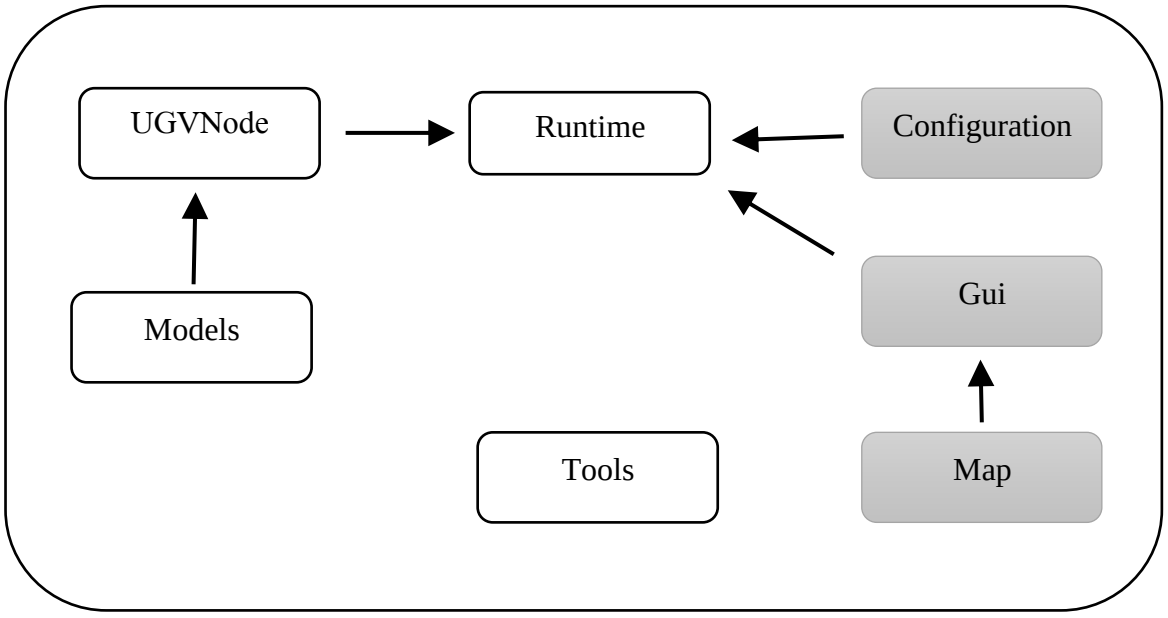
תיאור טכני

- הפרויקט נכתב בשפת ג׳אווה גרסה 8

- הפרויקט פותח על סביבת eclipse.
- הפרויקט רץ על thread אחד.
- מומלץ להגדיר את הjava heap ל MB 2048
- הפרויקט משתמש בחבילת swing לתצוגה הגרפית.

רכיבי התשתית – System view

המערכת בנויה משני חלקים מרכזיים שמופעלים דרך מודול "ניהול הריצה" שמנהל את כלל תהליכי המערכת.
החלק הסטטי (הרכיבים האפורים באיור 3) המאפשר את בניית התשתית של הסימולציה החל מעיבוד הקונפיגורציה ועד הפקת התצוגה למשתמש כולל המפה הקונקרטית שהמשתמש רוצה להפעיל. חלק זה מתבסס על התשתית שפותחה תחת פרויקט sinalgo תוך התאמתו לסימולטור.
החלק הדינמי מוגדר ע"י המודול של הרכב האוטונומי (UGVNode) עצמו יחד המודלים הרלוונטיים שמגדירים את התנהגותו וממשם את יכולות התנועה של הרכב. מעבר לכך קיים מודול של כלים סטטיים המאפשרים יכולות רוחביות שונות כגון, לוגים, סטטיסטיקות ועוד.



איור 3 מבנה סכמטי של הסימולטור

ניהול הריצה - runtime.runtime & runtime.guiRuntime

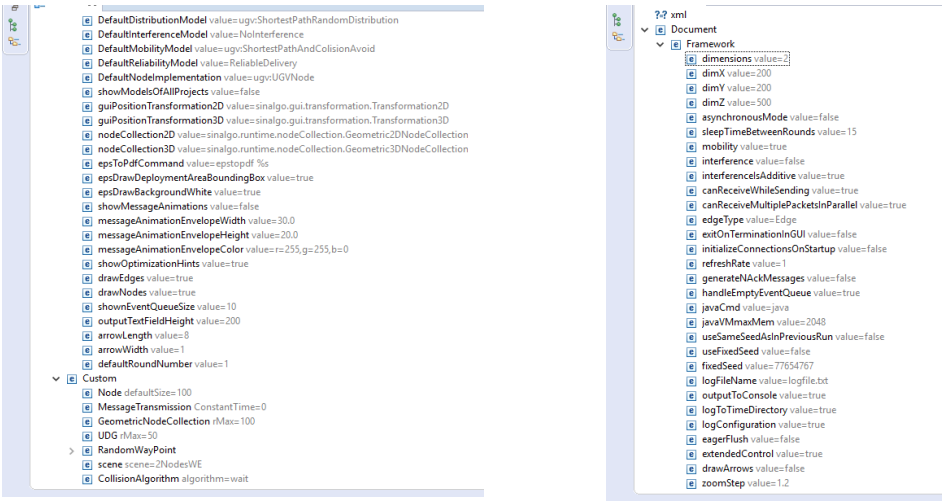
מודול ניהול הריצה בנוי משני חלקים מרכזיים. תהליך העלייה `initializeRuntimeSystem` ותהליך ניהול הריצה הדינמי run.
בתהליך העלייה המודול "קורא" נתונים מתוך קבצי הקונפיגורציה ובהמשך מתוך נתונים שמכניס המשתמש דרך התצוגה. בהמשך המודול מאתחל את כלל הרכבים המוגדרים

בקונפיגורציה ע״פ דרישות האתחול של כל רכב (מיקום, יעד ותכנון נתיב) ובסופו של דבר מייצר את התרחיש המלא ומוכן לשלב הריצה.
בתהליך ניהול הריצה הדינמי המודול מפעיל תהליכון `SynchronousRuntimeThread` שמריץ את תנועת כלי הרכב עד להגעת כולם ליעד (או לפי כמות הסיבובים המוגדרים).

קונפיגורציה

הקונפיגורציה של הפרויקט מוגדרת בקובץ config.xml ומאפשרת למשתמש להגדיר את הסימולטור בהתאם לרצונו. בנוסף, קיימים מספר פרמטרים שממשתמש מזין לסימולטור דרך התצוגה לאחר עליית המערכת ולמעשה ניתנים לשינוי בזמן ריצה. כלל היכולות נמצאות בחבילת configuration ומאפשרות לכלל רכיבי הסימולטור גישה לפרמטרים אלו.

מבנה קובץ ה config.xml :



איור 4 קובץ הקונפיגורציה

תצוגה

כלל רכיבי התצוגה מוגדרים בחבילת ה GUI . ישנם שני רכיבים מרכזים

controlPanel – הרכיב שאחראי לכל מעטפת התצוגה, ניהול התפריטים והחלונות ששולטים בסימולטור. תחת רכיב זה קיימים כמובן רכיבים קטנים יותר שמוטמעים בתוך הרכיב הכללי.

`GraphPanel` – הרכיב שאחראי לתצוגת המפה עצמה ומהווה את המנוע הגרפי לתנועת כל הרכב. רכיב זה מקבל את המפה הנדרשת, רזולוציה ומיקומים של הרכיבים ומציג אותם למשתמש.
הפאנל מרוענן כל זמן מסוים(שניתן לקנפוג, ברירת מחדל בכל סיבוב).

Popups, tooltip, menu, dialogs box ורכיבים נוספים שמוגדרים תחת החבילה הנ״ל ומאפשרים למשתמש יכולת אינטראקטיבית מול הסימולטור. מעבר לכך, קיימת יכולת תשתיתית להוספת פונקציות בתפריטים השונים בצורה קלה ומוגדרת.

מפה

המפה מתבססת על קובץ תמונה בפורמט gif או bmp ונמצאת בתיקיית תמונות בנתיב "projects/ugv/images"

בתהליך טעינת המפה מעבדים את התמונה וכל פיקסל שאיננו לבן נחשב כמכשול מבחינת ניתוח בתנועה של כלי הרכב . משיקולי ביצועים מומלץ לטעון מפה בשחור/לבן בגודל מקסימלי של Kb2 הגדרת מיקום המפה מוגדרת בקובץ קונפיגורציה תחת משתנה map ואובייקט MapIo.Map מנהל את עיבוד והצגת המפה בתצוגה.

כלי עזר

תחת חבילת כלי עזר קיימים מספר כלי עזר סטטיים שהתוכנה משתמשת בהם כגון כלי סטטיסטיקה ומבני נתונים נדרשים.

מצד המשתמש הגדרת אובייקט הלוג מוגדרת בחבילה זו tools.logging והגדרות מוד הפעלת הלוג, מיקום ושם קובץ הלוג ועוד הגדרות מוגדרות במודול זה . הגדרות הלוג מוגדרות בקובץ הקונפיגורציה תחת השדות הבאים: logFileName, outputToConsole, logConfiguration ועוד. הפעלת אובייקט הלוג מבוצעת ע"י מתודה סטטית logging.getLogger שיכולה להעביר קובץ לוג חדש או להפעיל את הלוג על הקובץ המוגדר בקונפיגורציה.

System view – רכב האוטונומי - UGV

כללי

רשימות חבילות המשנה בחבילה שממממשת את הרכב האוטונומי (להלן UGVNode) מצורפת להלן, כולן תחת חבילת ugv.

מעבר לכך ישנם מספר אובייקטים רוחביים המשמשים כספריות סטטיות ואינם תחת תיקיה.

Image

חבילת תמונות שמשמשות כרקע למפה. כל תמונה בתיקיה זו יכולה להטען דרך קובץ הקונפיגורציה.

Models

תיקיה זו מכילה את המודלים השונים שאנו משתמשים בהם

- collisionAvoidanceAlg
  - תיקיה המכילה אלגוריתמים לניהול אירוע תאונה. אלגוריתם זה הינו לב הסימולטור והמחקר. ישנה מחלקה אבסטרקטית המגדירה את התנהגות האלגוריתם ועל מימוש נדרש לרשת ממחלקה זו.
- connectivityModels
  - מודל סטטי (ספריה) המגדיר את הקשר בין הרכבים הנעים במפה. המודל מגדיר קשר בוליאני, כלומר, האם בין שני אובייקטים יש קשר ואם כן הם ברשימת השכנים אחד של השני. המימוש משתמש בפונקציית מרחק טריוויאלית.
- distributionModels
  - מודל המגדיר כיצד מחולקים הרכבים על פני המפה בעת חלוקה רנדומלית. המימוש מגדיר חלוקה על בסיס "קירות" המפה. ניתן לעקוף מודל זה ולייצר התחלה מוגדרת מראש כך שניתן לחזור על תרחיש מספר פעמים.
- messageTransmissionModels

- מודל המגדיר את ההודעה העוברת בין שני רכבים. המימוש הינו בסיסי ביותר ונדרש להרחבה אם יהיה צורך במודל החוקר תקשורת בין כלי הרכב. לא בשימוש מודל הנוכחי.

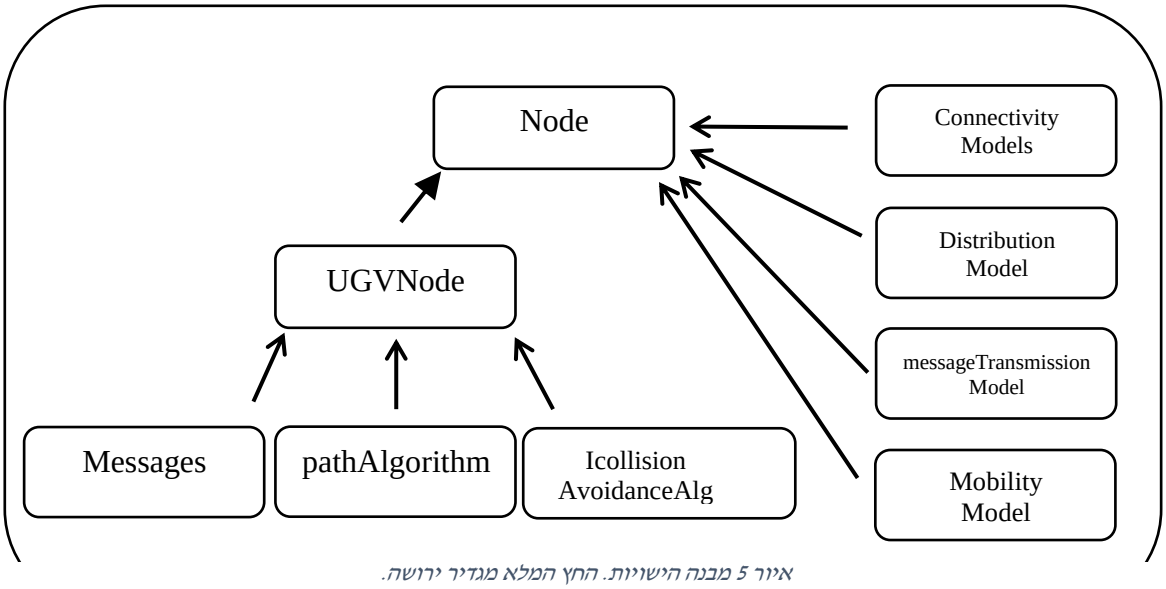
- mobilityModels
  - מודל התנועה של הרכב. מממש את הפונקציה המגדירה את הצעד הבא של הרכב ובכך מגדיר את ההתקדמות. המימוש בפועל מגדיר התקדמות על בסיס הנתיב המתוכנן אלא אם אנו במצב תאונה ואז נדרשת הפעלה של אלגוריתם לניהול תאונה.
- pathAlgorithm
  - אלגוריתם לתכנון נתיב. ממומש על בסיס A* ומבצע את החישוב של הנתיב הראשוני בין נקודת ההתחלה ליעד בהתחשב במכשולים הידועים. בהמשך אלגוריתם זה יוכל לחשב נתיב חדש מנקודה אמצעית כלשהי ליעד.

nodes

- Messages
  - תיקייה המכילה יכולת עתידית לשליחת הודעות בין רכבים. כרגע ממומש ברמת הכנה בלבד ודורש פיתוח.
- nodeImplementations
  - תיקייה המכילה את האובייקטים המרכיבים את הרכב האוטונומי. קיימת אפשרות עתידית לפתח רכב אוטונומי בעל מאפיינים אחרים היורש מהאובייקט האבסטרקטי ויושם בתיקייה זו. מעבר לישות הרכב – UGVNode - התיקייה מכילה מחלקה המגדירה את אירוע התאונה ומאפשרת ניתוח של אירועי יחסיים בין כלי רכב לטובת אלגוריתם שמתייחס לאירועי העבר ולטובת תחקור האלגוריתם. בנוסף, קיימת מחלקה המחזיקה את הנתיב של הרכב (על בסיס arraylist) ומאפשרת גישה לנתיב ועדכונו במידת הצורך.

מבנה ישויות סכמתי של מימוש הרכב האוטונומי

ישות הרכב האוטונומי יורשת מאובייקט כללי אבסטרקטי Node. חלק מהמודלים מוגדרים תחת ישות האב כי הם נדרשו למימוש ברמת התשתית. בכל מקרה קיימת לכלל המודלים גישה מהרכב האוטונומי. בתרשים מטה מוגדרים הקשרים בין המודלים לרכב האוטונומי.



איור 5 מבנה הישויות. החץ המלא מגדיר ירושה.

כלי הרכב אוטונומי – UGVNode

אובייקט זה הינו לב המערכת ומחזיק בתוכו את מודל התנועה יחד עם האלגוריתמים השונים לפעולתו.

מודלים/אלגוריתמים

connectivityModel.UGVConnectivityModel

מודל המגדיר מתי שני רכבים "קשורים" בניהם ומופיעים אחד ברשימה אחד של השני.

האלגוריתם בודק אם המרחק קטן מערך המוגדר בקובץ הקונפיגורציה "UDG/rMax". קיים מקרה קצה בו הרכב הגיע ליעדו ואז הוא "מרוקן" את רשימת החברים שלו ולא צובר חדשים גם אם הם מתקרבים אליו.

distributionModels.ShortestPathRandomDistribution

מודל יחסית פשוט המחלק מיקום התחלה בצורה רנדומלית לרכבים על "קירות" המפה. המודל משתמש בפונקציית ספרייה המוגדרת בתשתית ה common של הסימולטור.

AStarImp – מימוש אלגוריתם A* [12]

מימוש של האלגוריתם הידוע. הקלט לאלגוריתם הינו גרף המייצג את התמונה כאשר כל קדקוד בגרף מחובר לארבעת הקדקודים הסמוכים לו ובעל מרחק קבוע של אחד. במסגרת העיבוד של הגרף לקראת האלגוריתם אין קישור לאזורים השחורים (האגמים, האסורים לנסיעה) וכך האלגוריתם מוצא רק מסלולים שעוברים באזורים מותרים. הפלט של האלגוריתם הינו נתיב קצר ביותר בין נקודת ההתחלה ליעד של הרכב על בסיס קישוריות של ארבעה קדקודים לכל קדקוד.
למידע נוסף ראה הרחבה על האלגוריתם בנספח

mobilityModels.ShortestPathAndColisionAvoid

מודל פשוט המייצר את תנועת הרכב ע"פ נתיב קצר ומוגדר מראש של הרכב. המודל ממש פונקציה אבסטרקטית של `getNextPos` ומחזיר את הנקודה הבאה שהרכב צריך להימצא בה (על בסיס הניב שחישב הרכב בשלב האתחול). במידה והרכב נמצא במצב של "כמעט תאונה" המודל יפעיל את האלגוריתם למניעת תאונה.

אלגוריתם אבסטרקטי למניעת התנגשות – IcollisionAvoidanceAlg

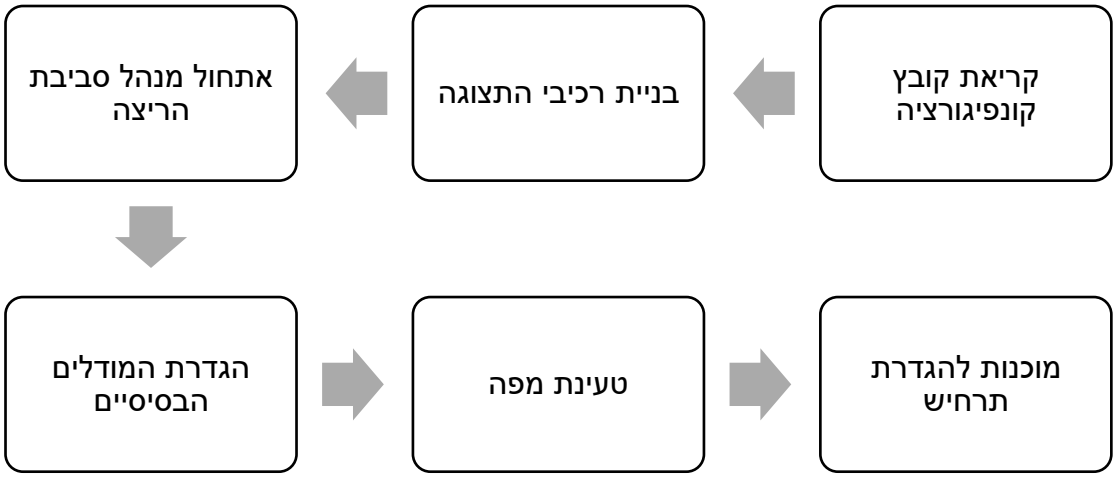
מחלקה אבסטרקטית המגדירה פונקציה מרכזית של getNextNode . פונקציה זו נקראת ע״פ זיהוי מצב של כמעט תאונה של הרכב ולמעשה מחליפה את מודל התנועה הסטנדרטי שהצגנו לעיל. הסימולטור מאפשר לממש אלגוריתמים שונים היורשים ממחלקה אבסטרקטית זו ולבחון את יעילותם.

Operation view – תהליכי המערכת

עליית הסימולטור

הפעלת הסימולטור תתבצע ע״ב הפעלת קובץ batch (אם לא מפעילים בסביבת eclipse. קובץ main נמצא ב runtime.main

תהליך עלית הסימולטור יורכב מהפעולות הבאות



הגדרת תרחיש

עקרונות הגדרת תרחיש:

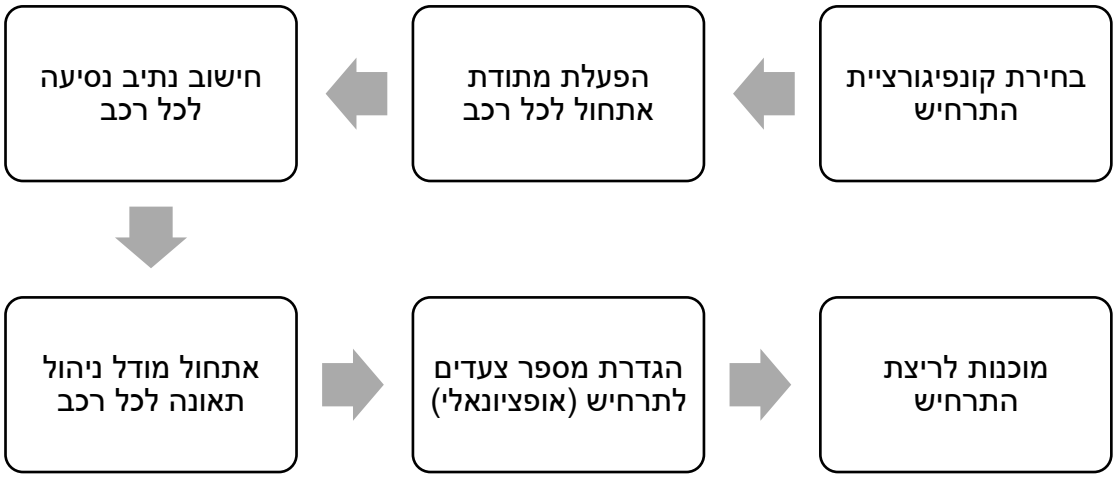
- מספר רכבים כאשר רכב מספר 1 הוא הרכב הנבדק (צבוע אדום)
- הגדרת אופן פריסת הרכבים על צידי המפה ונקודות היעד.
- בחירת אלגוריתם ניהול התנגשות לרכב הנבדק ולשאר הרכבים

*קיימים ערכים דיפולטיים המוגדרים בקונפיגורציה.

הגדרת התרחיש יכולה להתבצע בשני מודים עקרוניים:

1. ע״ב ממשק המשתמש בו בוחרים את מספר הרכבים. חלוקת רכבים על צידי המפה בצורה רנדומלית. אלגוריתם לניהול התנגשות חייבת להיות מוגדרת בקונפיגורציה.
2. הגדרת תרחיש ע״ב תרחישי ברירת מחדל שניתן לבחור בקובץ הקונפיגורציה.

תהליך הפקת התרחיש



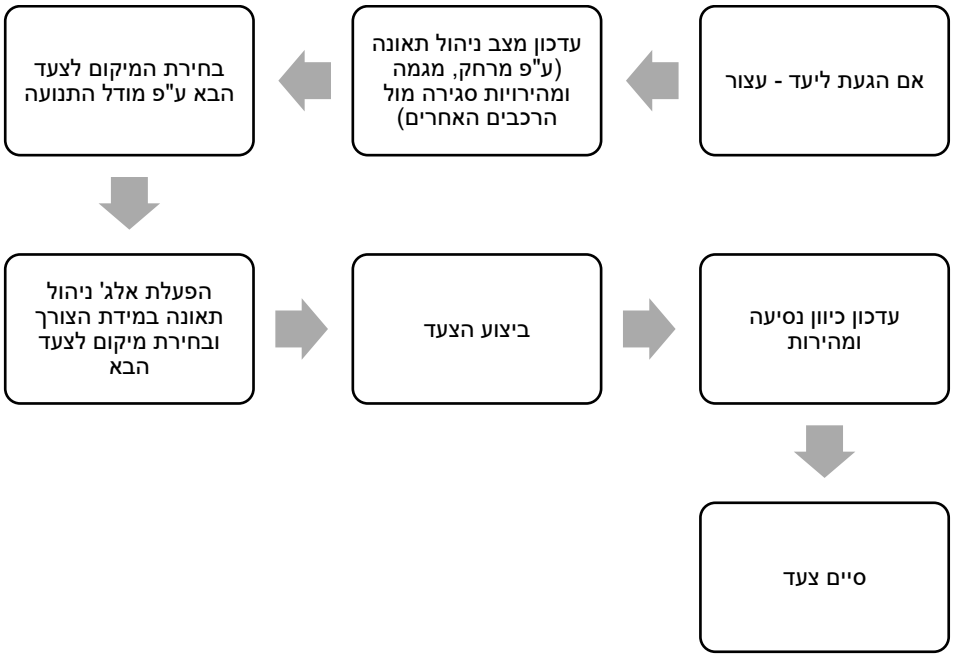
ריצת התרחיש

ריצת התרחיש מתחילה בפעולת משתמש לאחר מוכנות הסביבה ואפשור ההפעלה ע״י הסימולטור

ריצת הסימולטור מבוצעת ע״י הפעלת תהליכון המקבל את רשימת הרכבים המאותחלים ובכל צעד מקדם כל רכב לנקודה הבאה שלו ע״פ מודל התנועה

סיום צעד מוגדר לאחר סיום הפעלת כלל כלי הרכב ברשימה

תהליך הריצה לרכב בודד מתואר בתרשים הבא



חקירת התרחיש

לאחר סיום התרחיש ניתן לתחקר את התרחיש בשני אופנים.

תחקור זמן אמת מבוסס הדפסת נתונים לחלונית בתצוגה המציגה את מצבי ניהול האונה כפי שחווה אותם רכב מספר 1 .

תחקור מבוסס לוגים הכולל לוג טקסטואלי ע״פ כתיבות ללוג במהלך ריצת התכנית. בנוסף, קובץ CSV המכיל את כלל הצעדים ומאפשר תחקור ״עמוק״ של הסימולטור ונוצר בתיקייה summery

להלן דוגמא למבנה קובץ ה CSV

otherUGVHeading	thisUGVHeading	intersectionDistanceSquere	distanceSquere	pathIndex	roundsPerformed	colisionHandlingState	otherUGV	thisUGV
W	E	1104.5	2209	76	76	NOTHING	2	1
W	E	1012.5	2025	77	77	NOTHING	2	1
W	E	924.5	1849	78	78	NOTHING	2	1
W	E	840.5	1681	79	79	NOTHING	2	1
W	E	760.5	1521	80	80	IGNORE	2	1
W	E	684.5	1369	81	81	IGNORE	2	1

thisUGV – מזהה הרכב

otherUGV – מזהה הרכב השני

colisionHandlingState – מצב ניהול התאונה (של הרכב המתוחקר)

roundsPerformed – מספר צעד

distanceSquere – מרחק אוקלידי ביין הרכבים

intersectionDistanceSquere – מרחק אוקלידי לנקודת התנגשות הצפויה

thisUGVHeading – כיוון של הרכב

otherUGVHeading – כיוון של הרכב השני

Main functions and algorithms

Runtime.initializeRuntimeSystem()

מתודה המפעילה את סביבת הריצה ע״פ שורת הפקודה, מגדירה את המודלים הדיפולטיביים ובסיומה סביבת הסימולטור הוקמה.

UGVNode.init()

מתודה המופעלת בכלי הרכב ומאתחלת אותו כחלק מהכנת ריצת התרחיש

AStarImp.computePath()

מתודה המופעלת כחלק מאתחול כלי הרכב ומפעילה את חישוב אלגוריתם A* לפי נקודת ההתחלה והיעד של הרכב.

GuiRuntime.Run()

הפעלת ריצת התרחיש. מתודה זו נקראת ע״פ בחירת המשתמש בלחיצה על פקד הפעל בתצוגה.

ShortestPathAndColisionAvoid.getNextPos()

בחירת המיקום הבא. מתודה זו נקראת במהלך ריצת התרחיש כאשר ניהול הריצה קורא לכל רכב ומבקש ממנו להגדיר את המיקום הבא אליו הוא מתכנן לצעוד.

UGVNode.neighborhoodChange()

לאחר כל צעד מופעלת מתודה שבוחנת אם השתנו כלי הרכב המוחזקים ברשימת השכנים של הרכב ומעדכנת את הרשימה בהתאם של כל הרכבים. רשימה זו מהווה בסיס לבחינת מצב סטטוס "ניהול תאונה" שכל רכב מנהל לעצמו.

UGVNode.updateColisionHandlingState()

מתודה זו נקראת בתחילת כל צעד ומעדכנת את סטטוס ניהול התאונה לרכב ע"פ רשימת השכנים שעודכנה.

UGVNode.draw()

בסיום הצעד לכל הרכבים מופעלת מתודה זו ומציירת את הרכב, קוו ההתקדמות והנתיב על גבי המפה.

כלים/ספריות שהשתמשנו בבניית הסימולטור

JDOM

ספרית "קוד פתוח" java document object model הבונה את אובייקט ה XML ומאפשרת לתכנית לשלוף ולערוך את המידע בקובץ ה XML.

Swing

ספריית "קוד פתוח" המאפשרת לתכניות ג'אווה להציג ישויות גרפיות בארכיטקטורת MVC . הספרייה (ככל תכנית ג'אווה) יכולה לרוץ על כל מערכת הפעלה. הספרייה ממוקמת תחת javax.swing ובתוכה יש מחלקות לאובייקטים שונים כדוגמת jPanel, jButton ועוד.

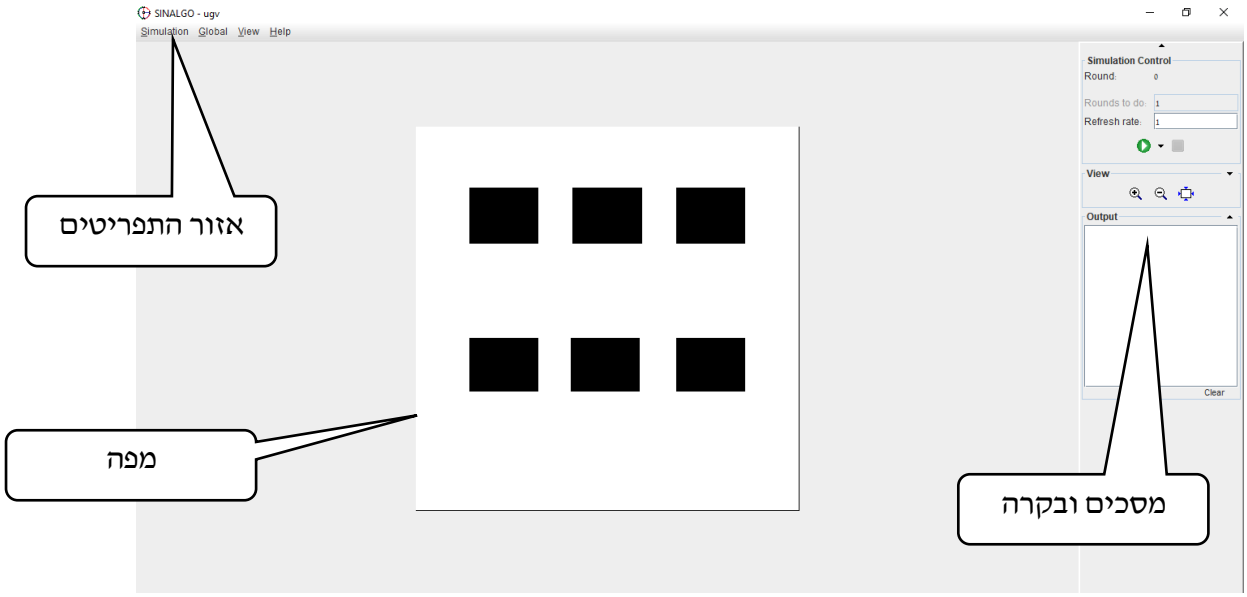
Sinalgo

סימולטור "קוד פתוח" המאפשר ניהול גרפים מקושרים בתצורות שונות המכוונות לחקירת גרפים במאפיינים שונים. הסימולטור שבנינו התאים והרחיב את sinalgo simulator

פעולות ותצוגת משתמש

כללי

מסך הסימולטור מורכב ממודול המפה ומודול הניהול המורכב מאזור התפריטים ואזור הבקרה.



איור 6 תמונת מסך של הסימולטור

תפריטים

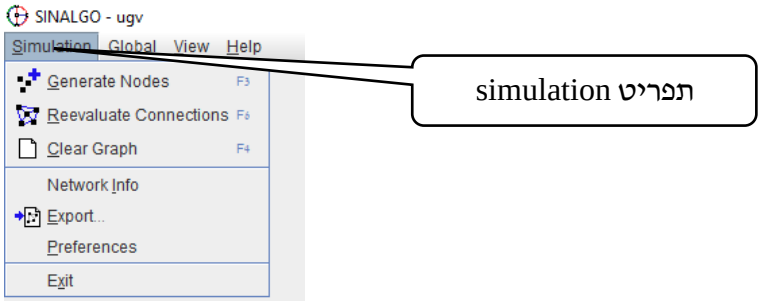
אזור התפריטים – Simulation, Global, View, Help

Simulation

מאפשר בעיקרו את בחירת הרכבים (generate nodes) וניקוי המפה לטובת תרחיש חדש.

בנוסף, ניתן להגדיר הגדרות משתמש מסוימות לטובת התרחיש (אופן ציור וכדו') ולייצא את התוצר לקובץ PS

<u>זהו התפריט המרכזי לשימוש הסימולטור שבנינו.</u>



איור 7 תפריט simulation

Global

מאפשר הגדרת ערכים ברמת כלל הסימולטור ובנוסף הצגת ערכי הקונפיגורציה שאיתם אותחל הסימולטור

View

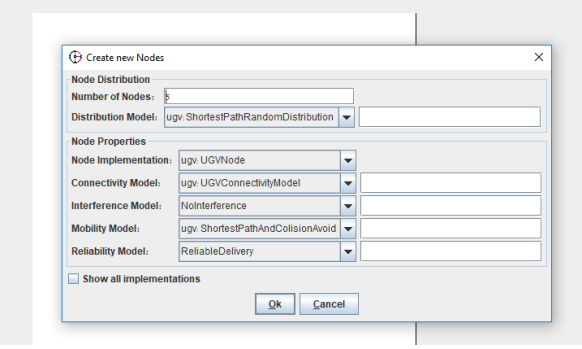
תפריט לניהול מרחק מצוגה

Help

תפריט אינפורמטיבי לגבי הסימולטור

תפריט יצירת רכב – Generate nodes

מאפשר להגדיר כמה כלי רכב נרצה לייצר לתרחיש ואת המודלים הרלוונטיים.
המודלים מוגדרים בסימולטור שבנינו באופן דיפולטי ואינם ניתנים לשינוי.
במידת הצורך, ניתן להרחיב אפשרות זו ולייצר מודלים נוספים שנוכל לבחור בשלב זה.



איור 8 יצירת תרחיש מהתצוגה

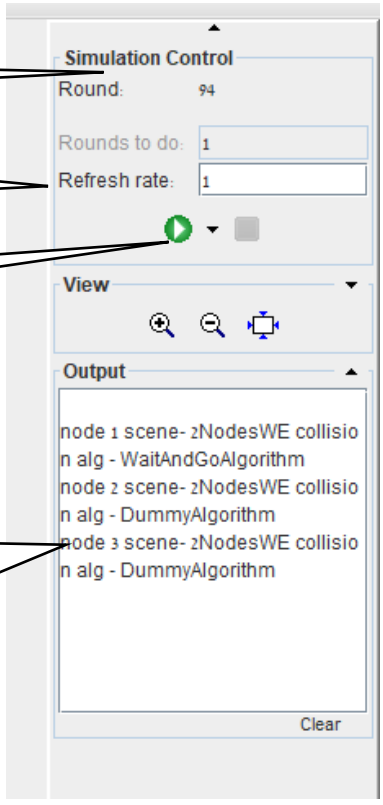
בקרת תרחיש ומסך תמונת מצב

מספר הצעדים מתחילת התרחיש

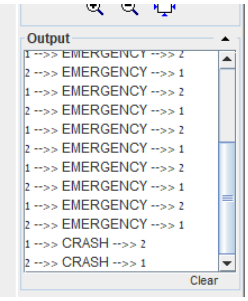
תדירות עדכון המפה. לצפייה חלקה נדרש לבחור 1

לחצן הפעלה ועצירה

מסך פלט זמן ריצה. מציג את תרחיש העלייה של הסימולטור ואלגי ניהול ההתנגשות בהמשך מציג את ניהול התאונה של הרכב הנחקר (איור 10).



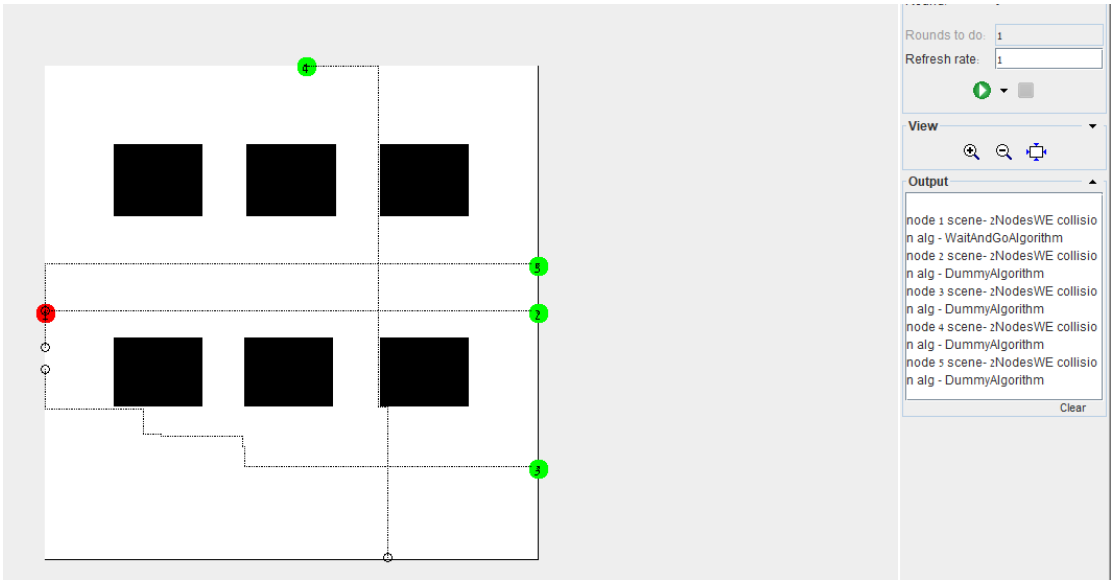
איור 9 ותמונת מצב באמצע הרצת תרחיש



איור 10 ניהול מצב התכנות לתאונה בין רכב מס 1 לאחרים

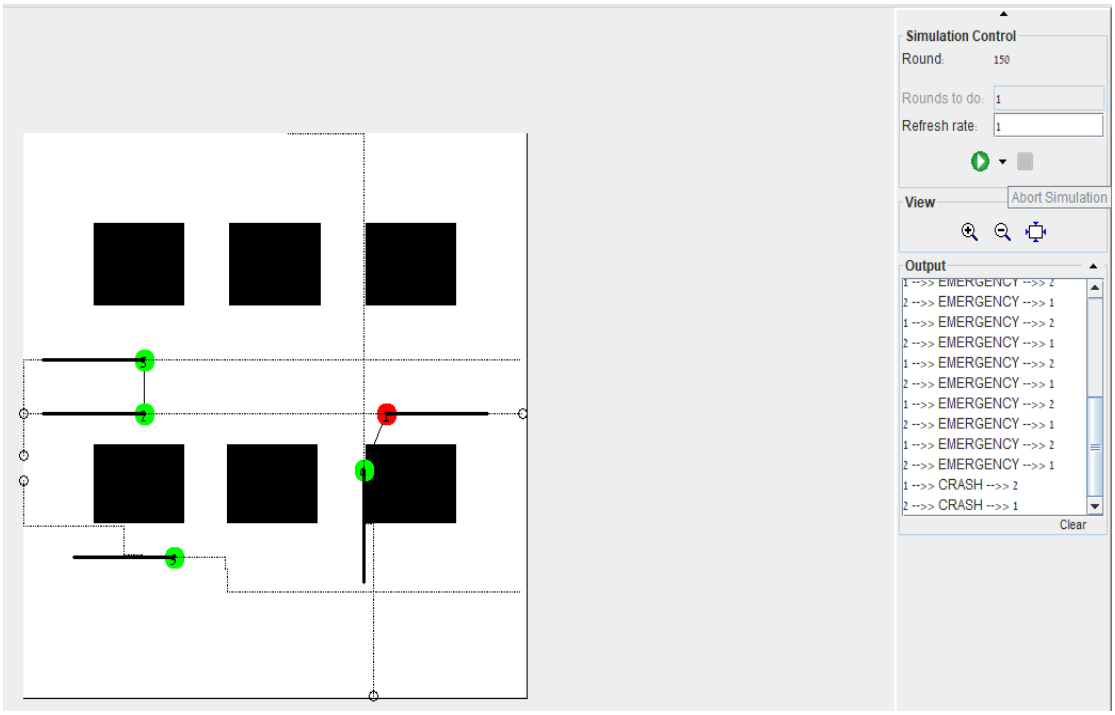
הרצת תרחיש

לאחר בחירת מספר הרכבים ניתן לראות את הפיזור הרכבים על המפה. הרכב האדום (מספר 1) הינו הרכב הנחקר ואילו שאר הרכבים (הירוקים) מהווים רכבים נוספים שנוסעים במרחב.
לכל רכב יש יעד המוצג במפה ע"י עיגול ריק קטן ומסלול הנסיעה המתוכנן אליו בקו מקווקו.
הריבועים השחורים במפה מגדירים אזורים אסורים לנסיעה.



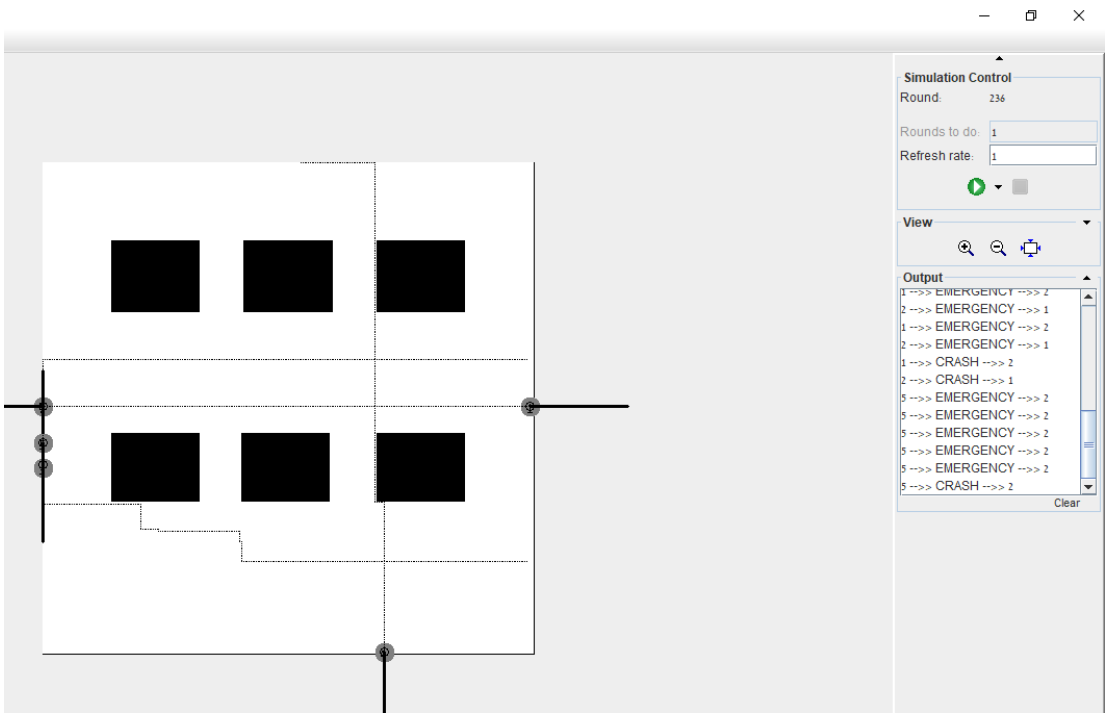
איור 11 הסימולטור מוכן להרצת התרחיש

באיור 12 ניתן לראות את מהלך ריצת התרחיש. לכל רכב מוגדר קו שחור עבה המגדיר את אזור העניין של הרכב. מתחת לטווח זה (של הקו העבה) מגדירים מצב סכנה ומתייחסים להתנהגות הרכב. חיתוך עתידי של שני קווי נסיעה מגדיר תאונה עתידית.
במסך הבקרה ניתן לראות שיש מצב חירום בין רכב 1 ל 2, בין הרכב האדום לירוק הקרו אליו עקב קרבה ביניהם.



איור 12 במהלך הרצת התרחיש

באיור 13 ניתן לראות את סיום התרחיש. כל רכב שהגיע יעדו נצבע באפור.
בנוסף, ניתן להרחיב את הנתונים של כל רכב (קליק ימני) ולראות כמה רכבים הוא ראה במהלך הריצה ואת מספר הצעדים שהוא רץ בפועל (לעומת התכנון).



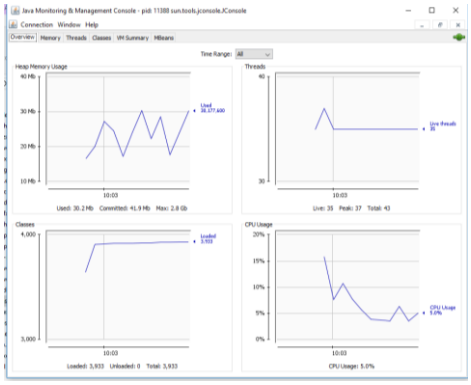
איור 13 סיום התרחיש

ביצועים

כללי

חישוב ביצועי הסימולטור יתמקד בשני מדדים מרכזיים:

- ניצול משאבים (CPU וזיכרון) ע"י שימוש ב jConsole
  - jConsole הינו מוצר של אורקל המשולב ב JDK של התקנת הג'אווה ומאפשר ממשק נוח לביצועי ה JVM בכל פלטפורמה.



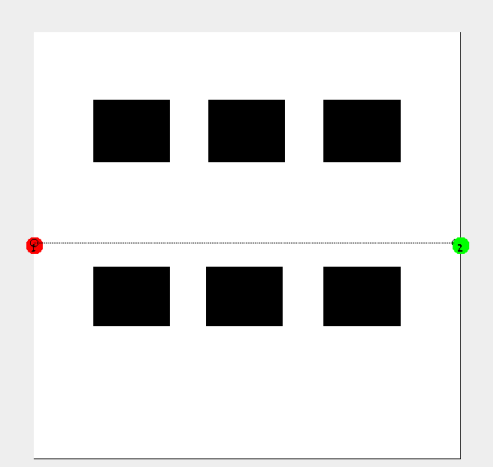
איור 14 תמונת מסך ראשי jConsole. ניטור (זכרון, מעבד, תהליכונים ואובייקטים)

- זמן ריצה אפליקטיבי של חישוב הנתיב הקצר וזמן ביצוע צעד
  - זמן חישוב נתיב קצר – נמדוד בעזרת כתיבה ללוג לפני ואחרי תחילת החישוב. בתיאוריה ככל שהמפה מורכבת יותר אז זמן החישוב יהיה ארוך יותר.
  - זמן ביצוע צעד לכל כלי הרכב – נמדוד את משך ביצוע איטרציית צעד לכל כלי הרכב על בסיס הנתיב הפשט ללא חישוב אלגוריתם ניהול התנגשות. הנחת העבודה היא שמימוש אלגוריתם למניעת התנגשות רלוונטי רק לרכב הנחקר (מס' 1) עקב מאפייני המחקר ואילו שאר הרכבים יתנהלו ללא אלגוריתם שכזה.

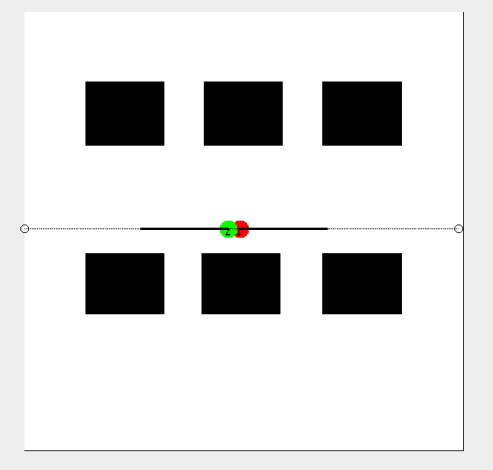
נשים לב שמטרת הסימולטור הינה לחקור את אלגוריתם מניעת ההתנגשות ולכן אירוע הכולל יותר מ 5 כלי רכב נראה לא סביר.

דוגמת הרצה

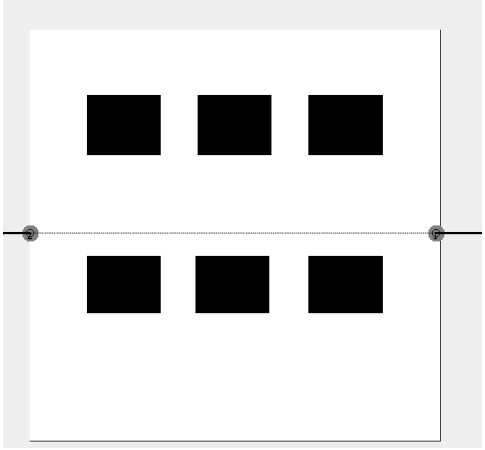
2 כלי רכב, אחד מול השני, לרכב שלנו (מס' 1, האדום) יש אלגוריתם מניעת תאונה מסוג "המתנה". המפה מורכבת משש קוביות המסמלות את המכשולים



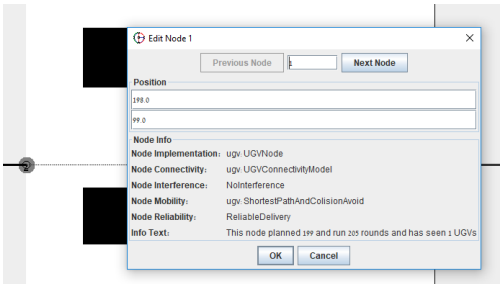
איור 15 תחילת הריצה



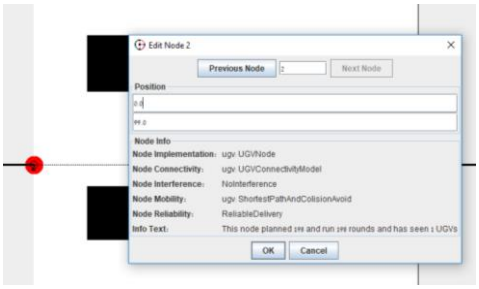
איור 16 מהלך הריצה



איור 17 סיום הריצה



איור 18 הרחבת נתונים עבור רכב מס׳ 1 - ניתן לראות בשורה התחתונה שהוא תכנן מספר צעדים לנתיב ובפועל רץ יותר בגלל ההמתנה



איור 19 הרחבת נתונים עבור רכב מס׳ 2 (ללא "המתנה")

בטבלה מטה מוצג קיטוע מתוך קובץ ה CSV המסכם את סטטוס ניהול התאונה של רכב מס׳ 1 ניתן לראות המתנה במקום החל מצעד 97 עד צעד 103 (ע״י העובדה שרכב מס׳ 1 נשאר באותה נקודה בנתיב וגם, לפי המרחק ביניהם, מהירות הסגירה קטנה בחצי) כתוצאה ממצב ניהול תאונה ברמת emergency ע״פ אלגוריתם מניעת תאונה מסוג ״המתנה״.

בצעד 102 התרחשה תאונה (crash). ניתן לראות שכיווני הרכבים היו מנוגדים כך שלמרות ה״המתנה״ של רכב מס׳ 1, רכב מס׳ 2 המשיך להתקדם לכיוונו.

otherUGV Heading	thisUGV Heading	intersectionDistanceSquere	distance Squere	pathIndex	roundsPerformed	collisionHandlingState	other UGV	this UGV
W	E	144.5	289	91	91	IGNORE	2	1
W	E	112.5	225	92	92	IGNORE	2	1
W	E	84.5	169	93	93	IGNORE	2	1
W	E	60.5	121	94	94	HANDLING	2	1
W	E	40.5	81	95	95	HANDLING	2	1
W	E	24.5	49	96	96	HANDLING	2	1
W	E	12.5	25	97	97	EMERGENCY	2	1
W	E	8	16	97	98	EMERGENCY	2	1
W	E	4.5	9	97	99	EMERGENCY	2	1
W	E	2	4	97	100	EMERGENCY	2	1
W	E	0.5	1	97	101	EMERGENCY	2	1
W	E	0	0	97	102	CRASH	2	1
W	E	0	1	97	103	HANDLING	2	1
W	E	0	9	98	104	HANDLING	2	1
W	E	0	25	99	105	HANDLING	2	1

דוגמת הדפסה מקובץ הלוג

זמן חישוב נתיב קצר לכל רכב וסטטיסטיקה על ביצוע צעדי הריצה.

```
time to path compute for ugv number 1 - 27.0
time to path compute for ugv number 2 - 7.0
rounds time stat:
max: 14.0
min: 2.0
mean: 10.850340136054422
stdv: 2.627312694208855
```

חישוב הנתיב קצר

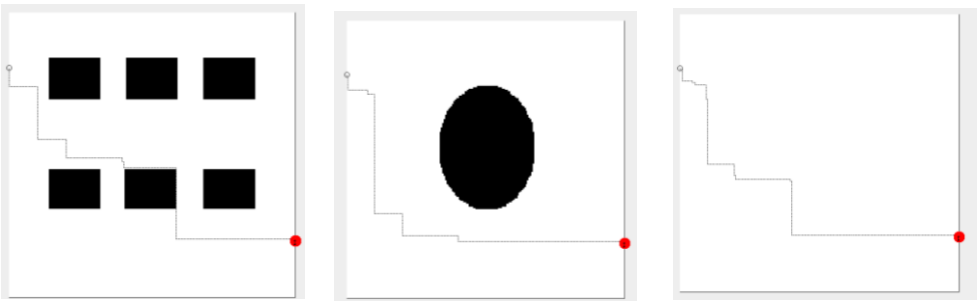
זמן חישוב הנתיב מושפע ממורכבות המפה וכמובן ליניארי לכמות הרכבים ורזולוציית המפה נבחן את החישוב לרכב בודד על סוגי מפות שונות על מפה חלקה, מפת דגל יפן, מפה עם 6 ו 25 ריבועים ומפת ״מלכודת״.

כל המפות בגודל של 200X200 פיקסלים

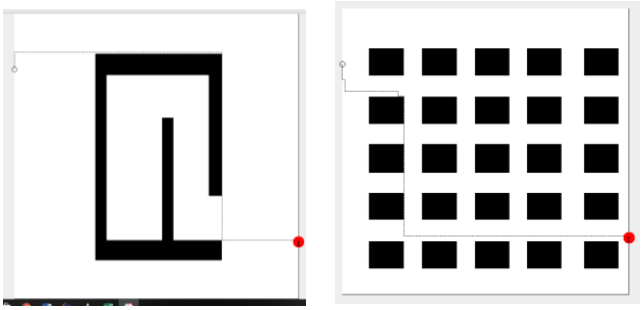
נקודת ההתחלה והסיום תהיה זהה.

הערה: כמות הקדקודים היא שמשפיעה בעיקר על זמן החישוב והיא דטרמיניסטית (לעומת זמן הריצה שמעט משתנה). התוצאה המפתיעה היא שבמפה חלקה מספר הקדקודים הוא הגבוה ביותר. הנ"ל נובע מהעובדה שאין מגבלות לבדיקה ולכן יש הרבה קדקודים בעלי מרחק זהה הן להתחלה והן לפונקציית היוריסטיקה. ככל הנראה, בגרף שמקושר ל 8 קדקודים (כולל האלכסונים) התוצאות תהינה שונות.

סוג מפה	זמן חישוב (מ"ש – מס' קדקודים שנבדקו)	אורך נתיב
חלקה	25525 - 23	318
דגל יפן	22835 - 18	318
6 קוביות	17879 - 16	318
25 קוביות	16757 - 16	318
"מלכודת"	19038 - 27	342



איור 20 מפה חלקה, דגל יפן, שש קוביות



איור 21 25 קוביות, מלכודת

גודל תמונה

נבחן את התוצאות למפות זהות בעלות רזולוציה גבוהה יותר – 600X600 פיקסלים

נוכל לראות שיש יחס ריבועי (טריוויאלי) במספר הקדקודים שנבדקים לעומת הגודל הקודם.

מומלץ להשתמש בתמונות ביחס של 200/200 משיקולי תאימות לגרף הסימולטור ואיכות החוויה.

סוג מפה	זמן חישוב (מ״ש – מס׳ קדקודים שנבדקו)	אורך נתיב (מס׳ קדקודים)
חלקה - 600	245385 - 110	958
25 קוביות - 600	170421 - 85	958
״מלכודת״ - 600	173445 - 80	1026

מספר כלי הרכב

נבחן את ביצועי הסימולטור ביחס לכמות כי הרכב הן מבחינת זמן ריצה של כל צעד והן מבחינת משאבי מערכת – Heap size + CPU.

התרחיש העקרוני פחות מעניין ונבצע אותו על מפה חלקה.

ניתן לראות שהסימולטור מתחיל לעבוד באיטיות ב 300 כלי רכב וביחס לתרחיש הסביר הביצועים מספקים בהחלט.

מספר כלי רכב	זמן ממוצע לצעד (מ״ש)	זמן צעד מינימלי (מ״ש)	זמן צעד מקסימלי (מ״ש)	CPUmax (%)	RAMmax (Mb)
5	7.4	4	120	21	57
50	15	5	150	55	123
100	21	5	180	55	539
300	480	33	1898	70 (ממוצע 28)	1510

וכן ביחס לחישוב נתיב למספר כלי רכב על מפה חלקה

מספר כלי רכב	CPUmax (%)	RAMmax (Mb)
5	2.5%	MB50
50	9	58
100	17	70
300	29	75

סיכום

כפי שהצגנו בהקדמה, פרויקט זה נועד להוות תשתית לסימולטור לחקירת אלגוריתמים בנושא של מניעת התנגשות בהתייחסות לנתיב נסיעה (השאיפה קצר ביותר). הסימולטור השיג את מטרותיו ומהווה תשית מצוינת כפי שבאה לידי ביטוי בעבודת גמר הנסמכת עליו. הסימולטור דורש מהמשתמש החוקר רמה טכנולוגית גבוהה עקב ההתאמות שנדרש לבצע לתחום המחקר הספציפי ואינו מיועד לשימוש למחקר תיאורטי בלבד.
תהליך הפיתוח ההנדסי היה מאתגר ומרתק הן ברמת ארכיטקטורת התוכנה והן ברמת האפיון ההנדסי בנדרש למימוש.

מה למדתי מהתהליך

תחום הרכב האוטונומי הינו תחום מאתגר ופורץ דרך. במהלך הפרויקט למדתי חלקים מהתחום הן ברמה התיאורטית והן ברמה ההנדסית. נושא זה מרתק אותי באופן אישי וביצוע הפרויקט בנושא נתן לי את הצעד הראשון לכניסה עתידית לתחום זה.
בנוסף, תחום המחקר האלגוריתמי בדגש על חשיבה אלגוריתמית ופיתוח סימולטור לבדיקה התברר לי כמתודולוגיה מעניינת ומאתגרת. המתודולוגיה של פיתוח כלי לבדיקת אלגוריתם כלשהו איננה מובנת מאליה והיוותה הפתעה חיובית מבחינתי. העובדה שאתה כחוקר מגדיר אלגוריתם ומפתח את הכלי הבודק מאפשרת מתודולוגיית מחקר יחסית קצרה ושלמה מבחינת תוצאות האלגוריתם.

האתגרים בפרויקט

במהלך הפרויקט התמודדתי עם אתגרים רבים ומעניינים וביניהם למידה של פרויקט "קוד פתוח", הבנת הארכיטקטורה המאתגרת ומידול הפיתוח הנדרש בהתבסס על תשתית חלקית קיימת.

בנוסף, פיתוח ממשק משתמש בג'אווה בהתבסס על ספריית SWING יחד עם מורכבות ההמרות בין קואורדינטות שונות (תמונה מול תצוגה) ולבסוף פיתוח סימולטור דינמי המאפשר חוויית משתמש טובה ובעלת משמעות למחקר דרשה רמת פיתוח גבוהה ומחקר עמוק של הנושא.

אפשרויות להמשך פיתוח

ניתן להרחיב את הסימולטור כמעט לכל תחום במחקר העוסק בתנועות רכבים אוטונומיים. אציין שני תחומים שנראים לי מעניינים:

1. חקירת תחום מניעת התאונה במערכות מתקשרות – ניתן בקלות יחסית לפתח את מנגנון ההודעות בין הרכבים ובכך לשתף מידע כגון נתיב מבוצע, מצב פנימי של הרכב ועוד. מידע זה יוכל לשפר משמעותית את האלגוריתם למניעת תאונה וכמובן דורש התאמה של האלגוריתם למידע החדש.
2. תשתית נתונים מחקרית ללמידת מכונה ע"י יצירת קלט ופלט מתאים. ניתן להתאים את נתוני הסימולטור למידע מובנה הדרוש לאימון רשת נוירונים או אלגוריתם לומד אחר. מחקר שכזה יוכל ליצור אלגוריתמים מורכבים יותר למניעת תאונה.

מבחינה טכנית/הנדסית - בהתאם לצורך המחקרי ניתן להרחיב את הסימולטור להיות גרף 8-מקושר. הרחבה זו נדרשת למימוש בשלב בניית הגרף מהתמונה ובשלב ניהול התנועה ומניעת התאונה. הרחבה זו תיתכן ותשפיע על ביצועי הסימולטור מבחינת ביצועים ונדרש לבדוק בהתאם.

ביבליוגרפיה

1) E. D. C. Group. *Sinalgo - simulator for network algorithms*. *http://dcg.ethz.ch/projects/sinalgo/*, 2008.

2) Latombe, J. C. (2012). Robot motion planning (Vol. 124). Springer Science & Business Media.

3) Bresson, G., Alsayed, Z., Yu, L., & Glaser, S. (2017). Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving. *IEEE Transactions on Intelligent Vehicles*, *2*(3), 194-220.

4) Frazzoli, E., Dahleh, M.A. and Feron, E., 2002. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, *25*(1), pp.116-129.

5) Yu, J. and LaValle, S.M., 2013, May. Planning optimal paths for multiple robots on graphs. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on* (pp. 3612-3617). IEEE.

6) Bareiss, D., & van den Berg, J. (2015). Generalized reciprocal collision avoidance. *The International Journal of Robotics Research*, *34*(12), 1501-1514.

7) Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), pp.269-271.

8) Likhachev, M., Gordon, G.J. and Thrun, S., 2004. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in neural information processing systems* (pp. 767-774).

9) Koenig, S. and Likhachev, M., 2002. D^* Lite. *AAAI/IAAI*, *28*.

10) Borenstein, J., & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. IEEE transactions on robotics and automation, 7(3), 278-288.

11) Fox, D., Burgard, W. and Thrun, S., 1997. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, *4*(1), pp.23-33.

12) Hart, P.E., Nilsson, N.J. and Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, *4*(2), pp.100-107.

נספח א' - אלגוריתם A*

אתגר הנתיב הקצר בגרף מקושר העסיק את החוקרים לא מעט. קיימים אלגוריתמים שונים כגון דייקסטרא, בלמן פורד ואחרים הפותרים את הבעיה אך לא בהכרח בצורה יעילה. פתרון יעיל דורש קודם כל נכונות ושלמות, קרי, מציאת נתיב קצר ביותר (אם קיים כזה) ובהמשך, יעילות, המוגדרת בעיקרה ע"י מספר הקדקודים שבדק האלגוריתם. האלגוריתם של דייקסטרא הוא נכון ושלם אבל בודק את כל הקדקודים ולכן ניתן (אולי) לשיפור בכך שנוכל לבדוק פחות קדקודים ועדיין לקבל פתרון נכון ושלם.

אלגוריתם A* משתמש בפונקציית תעדוף משולבת כדי לבחור את הקדקוד הנכון בכל איטרציה - פונקציית המרחק (המשמשת כפונקציית "מחיר") עד לקדקוד הנבדק בצירוף פונקציה יוריסטית המעריכה את המרחק (המחיר) מהקדקוד עד ליעד.

נניח ש n הינו קדקוד מסוים.

נגדיר את g(n) להיות פונקציית המחיר להגיע אל הקדקוד מנקודת ההתחלה

נגדיר את h(n) להיות פונקציית היוריסטיקה להגיע אל מהקדקוד אל היעד

אזי פונקציית התעדוף תהיה $f(n) = g(n) + h(n)$ כאשר ערך נמוך מהווה תעדוף גבוה.

נשים לב שאלגוריתם דייקסטרא הינו מקרה פרטי של פונקציה זו כאשר $h(n)=0$ לכל הקדקודים.