

תרגיל 1 - רובוטים ניידים (עקביה אחר קו - תנועה מבוקרת משוב, בקר PID, חישוב מסלול בעזרת אודומטריה)

תאריך: 2.12.25

קורס: רובוטים ניידים

מרצה: ד"ר יורם יקוטיאלי

מגישיים:

בנימין רוסין | ת"ז: 211426598

גיא דנין | ת"ז: 205372105

דוד ושלר | ת"ז: 209736578

קוד בGITהאב:

https://github.com/DavidWeschler/Mobile_Robots

תיקייה בדרכיב שמכילה את כל המדייה והקוד:

 [ex1_robots_media](#)

הקדמה

בתרגיל זה פיתחנו רובוט LEGO Mindstorms בעל הנעה דיפרנציאלית שמבצע עקיבה אחר קו כהה באמצעות חיישן אוור ייחיד. את האלגוריתמים והבקה מימושו בשפת NX, ואת ניתוח הנתונים (אודומטריה והציגת מסלולים) ביצענו ב-[Python](#).
הובודה כללת שתי משימות עיקבה: אלגוריתם בסיסי ואלגוריתם מתقدم המבוסס על בקר PID, וכן חישוב מסלול מדויק בעזרת אודומטריה והציגתו - הן בעזרתו **זוקן** כאשר החישוב בוצע "offline", והן על מסך הרובוט.

משימת עקיבה ראשונה

אלגוריתם ראשון -

בשלב הראשון התמקדנו בפיתוח אלגוריתם עקיבה פשוט שבוסס על ערכי חיישן האור. חיישן האור מציין ערך מסוים בתחום שבין 0 לבין 100. ערכים נמוכים מעידים על פני שטח כהים כמו הקו השחור או הכחול, וערכים גבוהים מעידים על פני שטח בהירים כגון הלבן. הגדרנו ערך סף ראשון של 50. כאשר ערך החישון קטן או שווה ל-50 הרובוט נמצא על הקו הכהה. כאשר הערך גדול מ-50 הרובוט נמצא על האזור הבHIR וסתה מהמסלול.



כאשר ערך החישון נמצא מתחת לערך הסף הרובוט ממשיך ישר. כאשר ערך החישון חוצה את הסף כלפי מעלה הרובוט מפסיק להתקדם ישר וועבר למצב סריקה. מצב הסריקה כולל פניות לסירוגין ימינה ושמאליה תוך תנועה לאחור עד שהחישון חוזר להציג ערך נmor שמייצגשוב אזור כהה. תדריות הדגימה הראשונה הייתה כל 10 מילישניות. מהירות הנסיעה ההתחלתית הייתה 60.

ביצענו ניסויים גם ב מהירות גבוהה יותר של 70. בזמן זה עדין השתמשנו בחישון במצב דלוק. בתצורה הצתת התוצאות היו פחותות יציבות והופיעו סטיות מהמסלול. הסיבה לכך הייתה שינוי גודל בערכי חיישן האור בין רגע לרגע, ככל הנראה בגלל השפעות תאורה חייזונית.

בין הניסויים בדקנו את ביצועי הרובוט כאשר החלון במעבדה פתוח ואשר הוא סגור. כאשר החלון היה פתוח נכנסה תאורה חיצונית חזקה שהשפעה מאוד על ערכיו החישוניים. שינויים קטנים באור נקבעו לשינויים משמעותיים בקריאת החישון ולכן התקבלו תוצאות פחות עיקריות. כאשר סגרנו את החלון ורמת האור בחדר התייצה התוצאות היו יציבות בהרבה.

ניסוי ראשון:

תוצאות הניסוי הראשון עם חישון דלוק, threshold שווה ל-50, דגימה כל 10 מילישניות ומהירות 60:

סיבוב נגד כיוון השעון: 29 שניות
סיבוב עם כיוון השעון: 25 שניות
תקופה זו התאפיינה בזמן ביצוע איטי יחסית ביחס לקבוצות אחרות.

[לצפייה בסרטון](#)

במטרה לנautor לשפר את ביצועי הרובוט, עברנו למצב א/or חישון קבוע. מצב הא/or של החישון גרם לכך שהקראייה שלו מבוססת יותר על אור חיצוני ופחות על התאורה הפנימית שהוא מקרין. לאחר שางרנו את החלון, נקבעה מוארת באופן קבוע ולא שינויים חדפים, הורדנו את ערך הסף ל-29. ערך זה נקבע לאחר בחינת ערכיו הקראייה על הקו ומוחזקה לו. כתוצאה זו ערכיו חישון נמוכים מ-29 שניות לאחור כהה, וערכים גבוהים מ-29 לאחור בהיר.

ניסוי שני:

תוצאות הניסוי לאחר שינוי מצב החישון לערך קבוע ושינוי ערך הסף ל-29:

סיבוב נגד כיוון השעון: 29 שניות
סיבוב עם כיוון השעון: 22 שניות
כאן כבר נצפה שיפור מסוים בזמןי הסיבוב, בעיקר בכיוון הנגד לשלון.
אבל - נתקלנו באין יציבות של הרובוט בעיקר בפניות החdot, שם הוא לפעמים סטה מהמסלול.

[לצפייה בסרטון](#)

כדי ליזבב את הרובוט ולמנוע סטיות, הדרקנו חזרה את חישון הא/or. בשלב זהה הגדלנו את תדירות הדגימה של ערך החישון מדגימה כל 10 מילישניות לדגימה כל 1 מילישניה. שינוי זה שיפור את מהירות התגובה של האלגוריתם. ברגע ש- $\Delta t = 1$ מספיקים כדי לזהות עליה בערך החישון, הרובוט מצליח לתקן את המסלול מהר יותר וimbalance פחות זמן לאחור הבהיר.

ניסוי שלישי:

תוצאות הניסוי לאחר העלאת תדירות הדגימה:

סיבוב נגד כיוון השעון: 24 שניות
סיבוב עם כיוון השעון: 20 שניות
שיפור זה היה משמעותי והפרק את האלגוריתם למדוייק ומהיר יותר.

[לצפייה בסרטון של נסיעה נגד כיוון השעון](#)

[לצפייה בסרטון של נסעה עם כיוון השעון](#)

במהלך העבודה שיחקנו עם ערכיו המהירות הבאים: מהירות הנסעה (speed) ומהירות הסריקה (searching speed). בניסויים אלו רأינו בבירור את trade-off בין מהירות לבין יציבות. כאשר העלינו את המהירות הרובוט אمنם שהלים את הקפה מהר יותר, אך הופיעו יותר מצבים של סטייה מהמסלול. מצד שני, בחירה ב מהירות נמוכה יותר שיפורה את יציבותן אך האריכה את זמן הקפה. שינוי ב מהירות החיפוש יוצר השפעה ישירה על מידת

האגסיביות של תיקוני המסלול בזמן שהחישן יצא מהאזור הכהה. מהירויות גבוהות מדי גרמו לתנועות חדות ולפעמים לאובדן עקיבה, ומהירות נמוכות מדי גרמו לרובוט לשחות זמן רב במצב סריקה.

ניסינו גם מספר שיטות לבחירת כיוון הסריקה בעת חריגה מהcko. ניסינו להתחיל תמיד ימינה. ניסינו להתחיל תמיד שמאליה. ניסינו להחליף כיוון בכל פעם ללא קשר לערך האחרון. בנוסף ניסינו לשמור את הכוון האחרון שבו נמצאה הצלחה ולבסס עליו את הכוון הבא. בסופו של דבר הגיע השיטה ביותר הייתה שהיא הייתה שונה זו מזוה הצלחות של שגיאות לצד אחד ושיפרה את כיסוי אזור החיפוש.

קישור לקוד של אלגוריתם זה (הערה - זהו לא הקוד הסופי של משימת העקבה הראשונה)

- אלגוריתם שני וסוף -

לאחר שלא הצליחנו לרדרת זמן של 20 שניות לשיבוב באלגוריתם המקורי, עברנו לבדוק אלגוריתם אחר שמיועד לעקבה מהירה יותר על קו מבלי לבצע עצירות במהלך הנסעה. מתוך אתר [esigbots Bang Bang](#), והחלטנו לישם גרסה שלו המבוססת על המשתנים המופיעים בקוד שלנו.

באלגוריתם Bang Bang אין שימוש בערך סף קבוע מראש. במקום זאת ערך הסף THRESHOLD מחושב בהתאם ממוצע בין BLACK לבין WHITE. בקוד שלנו BLACK מוגדר כ-30 והוא מייצג את ערך חישון האור באזורי השחור, ו-WHITE מוגדר כ-60 והוא מייצג את ערך האור באזורי הלבן. מכיוון ש-THRESHOLD מחושב כממוצע בין האזורי השחור לבין, מתקובל ערך סף של 45. האלגוריתם מנסה לשמור את ערך החישון כמו שייתר קרוב לערך זה, מה שמקם את הרובוט על הקו הכהול.

לאחר חישוב הסף התחילנו בהרצות עם BASE_SPEED מוגדר ל-90. מהירות הפניה הוגדרה על פי אותויחס שבו השתמשנו בקוד, קלומר BASE_SPEED חלק 2. המשמעות היא שבפנייה מהירות המנוע האיטי הייתה 45. בדומה זו האלגוריתם היה יציב והרובוט השלים סיבוב מלא בזמן של 13 שניות. לא היה הבדל בין נסעה בכיוון השעון לבין נסעה נגד כיוון השעון.

לאחר מכין ביצענו סדרת ניסויים כדי לשפר את זמן הסיבוב. בכל הניסויים שמרנו על עיקרון הפניה מהקוד, קלומר שינוי מהירות אחד המנועים על פי יחס מוגדר של BASE_SPEED לעומת מהירות מתוקנת. הניסויים שבוצעו היו:

ניסוי ראשון:
ה��ירוט הפניה הוגדר ל-100. המהירות הפניה הוגדרה כ-100 חלק 2, קלומר 50. בניסוי זהה הרובוט יצא מהמסלול ולא השלים סיבוב.

לצפיה בסרטון

ניסוי שני:
ה��ירוט הפניה הוגדר ל-80. המהירות הפניה הוגדרה כ-80 חלק 1.5, קלומר בערך 53. גם בניסוי זהה הרובוט סטה מהמסלול.

ניסוי שלישי:

BASE_SPEED הוגדר ל-100. מהירות הפניה הוגדרה כ-100 חלקי 2.5, כלומר 40. בניסוי זה זמן הסיבוב התקבל כ-15 שניות.

ניסוי רביעי:

BASE_SPEED הוגדר ל-100. מהירות הפניה הוגדרה כ-100 חלקי 1.1, כלומר בערך 47.6. בצד זה הרובוט השלים סיבוב בזמן של כ- 9 שניות.

לצפייה בסרטון נסעה עם ציון השעון

לצפייה בסרטון נסעה נגד ציון השעון

מניתוך התוצאות אפשר לראות בבירור שהקיים trade off בין יציבות לבין זמן הסיבוב. כאשר BASE_SPEED הייתה גבוהה מדי הרובוט התחיל לסתות מהקוו לפני שהפיק תגובה מתאימה. מצד שני, כאשר מהירות הפניה לא הייתה גבוהה Enough מדויק ל-BASE_SPEED התיקון היה חד מדוי או איטי מאוד, וכטזאה מכרי הרובוט או יצא מהמסלול או איבד זמן. יחס המהירויות בין BASE_SPEED לבין BASE_SPEED חלקו מקדם הפניה היה הגורם המרכזי שהשפיע על הצלחת הפניה ועל זמן ההקפה.

בסיומו של דבר השימוש המוצלח ביותר היה BASE_SPEED שווה ל-100 ומהירות הפניה של בערך 47.6, והוא אפשר זמן הקפה של כ-9 שניות, שהוא הזמן הטוב ביותר שהשגנו עם אלגוריתם Bang.

קישור לקוד הסופי של משימת העקביה הראשונה

קישור לסרטון מלא של משימת העקביה הראשונה - שני סיבובים ועכירה בסוף (עם ציון השעון)

קישור לסרטון מלא של משימת העקביה הראשונה - שני סיבובים ועכירה בסוף (נגד ציון השעון)

משימת עקיבה שנייה

במשימה זו השתמשנו בבקר PID כדי לשפר את איות העקיבה ביחס לאלגוריתמים פשוטים. המטרה הייתה להציג תנואה רציפה, מהירה ויציבה יותר, תוך שימוש תיקוני כיוון חלקים המבוססים על השגיאה בין ערך חישון האור לבין ערך היעד. ביצענו סדרת ניסויים לכיוון מוקדי P ו-D עד למציאת שילוב אופטימלי.

ניסוי ראשון:

בחרנו להתחילה עם ערך קטן יחסית של (0.8) kp וערך גובה יותר של (1.2) kd כדי לבדוק תגובה רגישה יותר לתיקוני השגיאה המידית ולהמעיט את תגובה ה-P, מתוך תקווה שהתנהגות הרובוט תהיה יציבה ולא תגרום לרעידות יתר. kp הוגדר ל-0.8 ו-kd הוגדר ל-1.2. התנהגות הרובוט הייתה לא טובה, מכיוון שהוא לא עקב אחרי הקו השחור. אמונם הרובוט עשה סיבוב כלשהו, אך לא היה טעם למדוד אותו מכיוון שהוא קיצר הרבה פינוט. התוצאה הייתה תנואה עצבנית, והרובוט סטה מהמסלול הרבה.

לצפיה בסרטון

ניסוי שני:

בחרנו להעלות את ערך kp ל-1.5 ולהוריד את kd ל-0.5 כדי להגבר את ההשפעה של תיקון הסטייה המידית ולהפחית את השפעת ה-damping (המאופיין ב-kd), במטרה לבדוק האם הרובוט יעקוב טוב יותר אחרי הקו אך kali תגובה רכות מדי. kp הוגדר ל-1.5 ו-kd הוגדר ל-0.5. בניסוי זה הרובוט הפגין תוצאות טובות כאשר נגד כיוון השעון - 9.18 שניות להשלמת סיבוב, אך כאשר נגע עם כיוון השעון הרובוט ברוח מהמסלול.

לצפיה בסרטון נסעה עם כיוון השעון

לצפיה בסרטון נסעה נגד כיוון השעון

ניסוי שלישי:

בחרנו לשלב ערך ביןים של (1.2) kp עם ערך גובה יותר של (1.8) kd במטרה להחזיר קצר יציבות לתנועות הרובוט ולמנוע סטיות גדולות על ידי הגדלת השפעת רכיב ה-D, בתקווה לשפר את יציבותו לאורך המסלול. kp הוגדר ל-1.2 ו-kd הוגדר ל-1.8. התוצאות כאן היו דומות לניסוי השני, עם זמן של 9.51 שניות נגד כיוון השעון ובריחה מהמסלול עם כיוון השעון.

ניסוי רביעי וסوفي:

לאחר שניסינו כמה שילובים שונים, החלפנו לכיל את הבקר לערכי kp ו-kd גבוהים יותר (2.0) כדי לקבל תגובה מהירה יותר ואגרסיבית יותר לתיקון הסטייה, אך גם מאוזנת מספיק כדי לשמר על יציבות בתנועת הרובוט לאורך המסלול הקצר. קבענו מהירות בסיסית של 90, וקבענו את ערכי ה-IDP כך שהkp יהיה שווה ל-2.0 ו-kd יהיה שווה

ל-2.0. ערכים אלה נמצאו כנקודות איזון יعلاה במיוחד בין תగובה מהירה של הרובוט לבין יציבותו בתיקון הסטיה ו מהירותו הרובוט הייתה כ-9 שניות לסיבוב.

במהלך הנסעה נפתחה התנהגות יציבה: הרובוט שמר על קו עקיבה רציף, ביצע תיקוני ציון מהירים אך לא אגרסיביים, ולא נרשמו מצבים שבהם הוא סטה מהמסלול. התגובה של חישון האור לערכי ה-*PID* הייתה חלקה ולא נצפו תנודות או רטט במסלול.

ניסוי זה היה הניסוי המוצלח ביותר מתוך סדרת ניסויי ה-*PID*, והוא מראה את נקודת הסיום בתהיל'ר יכול של הבקר עברו חלק זה של העבודה.

לצפייה בסרטון נסעה עם ציון השען

לצפייה בסרטון נסעה נגד ציון השען

ניתוח תוכאות ביחס למשימת העקביה הראשונה:

ה-*PID* שיפר משמעותית את הביצועים ביחס לאלגוריתמים של משימה 1. אמנם זמן ההקפה היה דומה לאלגוריתם של משימה 1, כ- 9 שניות, אך העקביה הפכה יציבה יותר ולא איבודו קו. בנוסף, הפעורים בין הכוונים כמעט נעלמו, והרובוט הגיע טוב יותר לשינויים בתאורה. בסך הכל, ה-*PID* סיפק שילוב טוב יותר של מהירות, דיוק ויציבות.

הקוד הסופי של משימת העקביה השנייה PID

קישור לסרטון מלא של משימת העקביה השנייה - שני סיבובים ועצירה בסוף (עם ציון השען)

קישור לסרטון מלא של משימת העקביה השנייה - שני סיבובים ועצירה בסוף (נגד ציון השען)

משימת חישוב המסלול על ידי אודומטריה

קישור לקוד של אלגוריתם PID שמחשב את המסלול באמצעות אודומטריה ושומר לקובץ
קישור לקוד של אלגוריתם PID שמחשב את המסלול באמצעות אודומטריה ומציג את המסלול על
המסך
קישור לקוד פיתון הממיר float של המסלול על סמך קובץ הפלט

השיטה בה השתמשנו לחישוב מיקום הרובוט מבוססת על קינמטיקה של הינע דיפרנציאלי. בשיטה זו, אנו מניחים שהרובוט נע על משטח דו-ממדי ושולט על כיוון התנועה וה מהירות אך ורק באמצעות שינוי המהירות היחסית בין שני הגלגים המקבילים.

הчисוב:

האלגוריתם מבצע סכימה של תנועת הגלגים בכל מחזור ריצה של הבקר. הנחתasis היסוד היא שבפרק זמן קצרים מאוד $0 \rightarrow dt$, הרובוט נע בקצב מעגלית או בקו ישר.

הчисוב מתבצע בשלושה שלבים עבור כל מחזור לולאה:

1. חישוב המרחק שעבר מרכז הרובוט ($\Delta Distance$):

אנו קוראים את המידע מהמקודדים (החו"שנים של המנווע) ומחשבים ממוצע של המרחק שעבר הגלגש השמאלי והימני. זהו המרחק שעבר מרכז הציר של הרובוט.

$$\Delta D = \frac{\Delta R + \Delta L}{2}$$

2. חישוב השינוי בכיוון ($\Delta\theta$):

ההפרש בין המרחק שעבר גלגל ימין לגלגל שמאל, כשהוא מחולק ברוחב הרובוט ($TrackWidth$), נותן לנו את הזווית (ברדיאנטים) שהרובוט הסתובב באותו מחזור.

$$\Delta\theta = \frac{\Delta R - \Delta L}{TrackWidth}$$

3. עדכון המיקום החדש (אינטגרציה):

אנו משתמשים בטריגונומטריה כדי להטיל את המרחק (ΔD) על מערכת הצירים הגלובליות (X, Y), בהתאם על הזווית הנוכחית, ומוסיפים זאת למיקום הקודם:

$$\theta_{new} = \theta_{old} + \Delta\theta$$

$$X_{new} = X_{old} + \Delta D * \cos(\theta_{old} + \Delta\theta/2)$$

$$Y_{new} = Y_{old} + \Delta D * \sin(\theta_{old} + \Delta\theta/2)$$

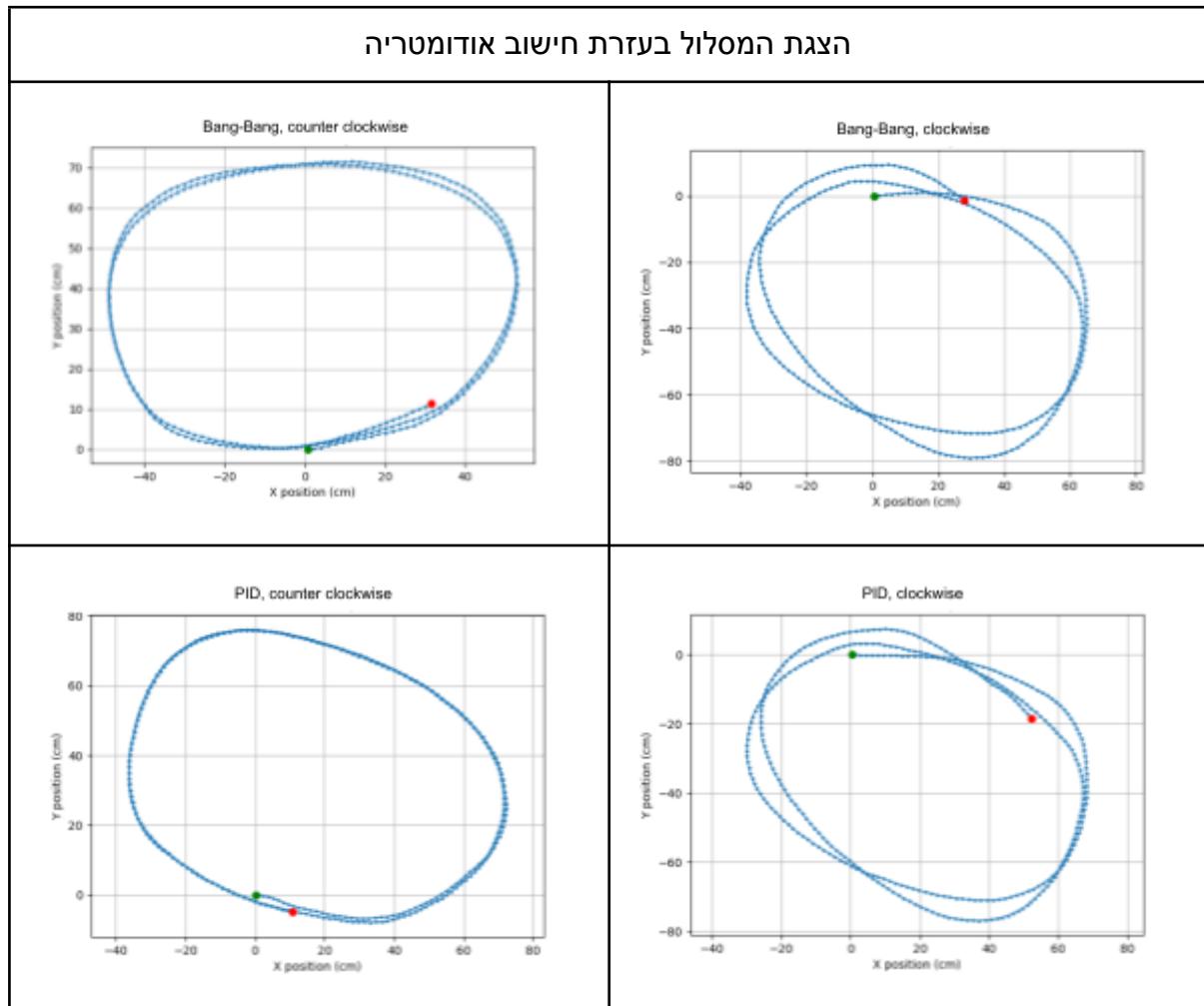
Trade-offs בין מהירות לדיקוק:

קיים מתח בין הרצון לנوع מהר לבין הצורך בבדיקה באודומטריה:

1. תדירות דוגמה מול מהירות נסעה: ככל שהרובוט נושא מהר יותר, הוא עובר מרחק רב יותר בין כל דוגמה של המעבד. בקשנותות חדות, דוגמה איטית מדי תגרום לחישוב "לחתו" את הסיבוב ולהפוך קשת חלקה לקו ישר (Aliasing), מה שיוצר שגיאה מצטברת במיקום.
2. החלקה: בתאונות גבוהות או בלימוט פתאומיות (ה貌יניות לאלגוריתם Bang-Bang), הגלגים נוטים להחליק על המשטח. המקודדים "חושבים" שהרובוט צז, אך בפועל הוא מחליק במקום. נסעה איטית יותר ממזערת החלקה ומספרת את הדיקוק הפיזי.

כפי שנitin לראות בגרפים הבאים (בעמוד הבא), בשני האלגוריתמים הרובוט נושא בצורה חלקה יותר נגד כיוון השעון מאשר עם כיוון השעון.

הסיבה לכך קשורה להבדל בהיקפי הגלגלים: גלגל שמאל יותר (היקף 17.5 ס"מ) וגלגל ימין גדול יותר (היקף 17.7 ס"מ). מאחר שגלגל ימין מתקדם מעט יותר בכל סיבוב, נוצר סטייה טבעית שתורמת לכך שהתנועה נגד כיוון השעון מאוזנת יותר עם הפער זהה, ולכן דורשת פחות תיקוני מסלול. לעומת זאת, כאשר הרובוט נושא עם כיוון השעון, אותו פער בין הגלגלים פועל נגד כיוון הסיבוב, מה שմגדיל את הצורך בתיקונים תכופים יותר. לכן מתקבל מסלול פחות חלך יותר "מצוגג".



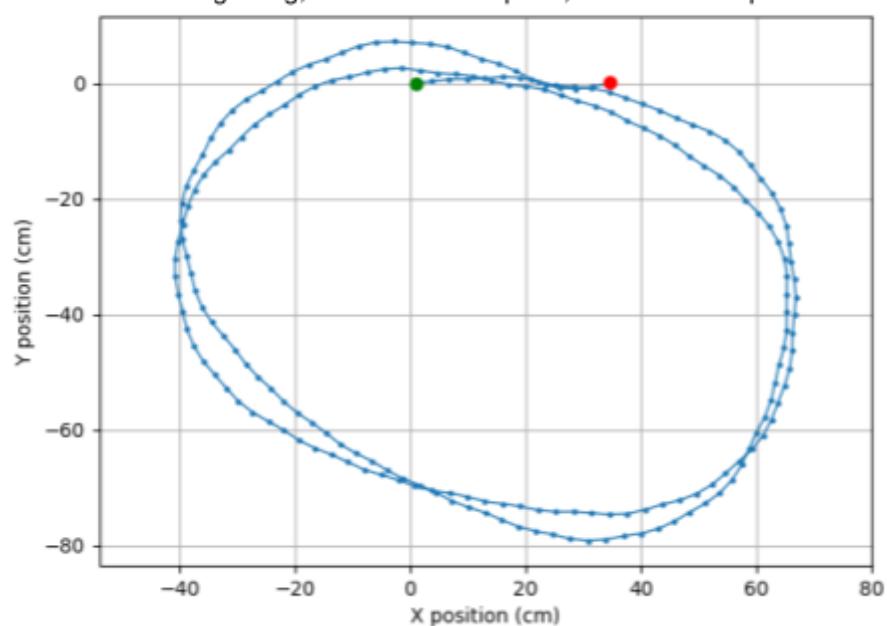
הציגת המסלול על גבי המסך,
2 סיבובים באלגוריתם PID נגד כיוון השעון



הציגת המסלול על גבי המסך,
2 סיבובים באלגוריתם PID עם כיוון השעון



Bang-Bang, with increased speed, trade-off example



ביבליוגרפיה

אודומטריה ו חישוב מסלול:

.Going from Odometry to Position in Two Dimensions .1

<http://credentiality2.blogspot.co.il/2010/06/going-from-odometry-to-position-in-two.html>

.Odometry .2

ויקיפדיה.

<http://en.wikipedia.org/wiki/Odometry>

.Rossum, R. Differential Steering for Mobile Robots .3

<http://rossum.sourceforge.net/papers/DiffSteer/>

.Calculate position of differential drive robot .4

.Robotics StackExchange

<http://robotics.stackexchange.com/questions/1653/calculate-position-of-differential-drive-robot>

:בקר PID

.PID Controller .5

ויקיפדיה.

http://en.wikipedia.org/wiki/PID_controller

.PID Controller for LEGO Mindstorms Robots .6

.Inpharmix

http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html

.Using a PID Controller .7

.Seattle Robotics Society

http://www.seattlerobotics.org/encoder/200108/using_a_pid.html

דוקומנטציה שפת NXC :

.NXC API Documentation .8

<https://bricxcc.sourceforge.net/nbc/nxcdoc/nxcapi/index.html>

מאמרם מקצועים נוספים:

.Nahmed, N. Wheel Odometry Model for Differential Drive Robotics .9

.Medium

<https://medium.com/@nahmed3536/wheel-odometry-model-for-differential-drive-robotics-91b85a012299>

שימוש בכלים נוספים:

. שימוש במערכת Gemini לתיקון שגיאות תחביר (syntax) בקוד .10

אלגוריתם למשימת העקבה הראשונה:

Purdue SIGBots. (n.d.). *Bang-bang control*. Retrieved November 24, 2025, from .11

<https://wiki.purduesigbots.com/software/control-algorithms/bang-bang>