ניווט רובוטים מונחה חיישנים - 00360044

אביב תשפ"ה 2025

חלק 1+2 של פרויקטון

מגיש:

שם	אימייל
אלון עומר בן דוד	alon.bendavid9@gmail.com
איל א'	

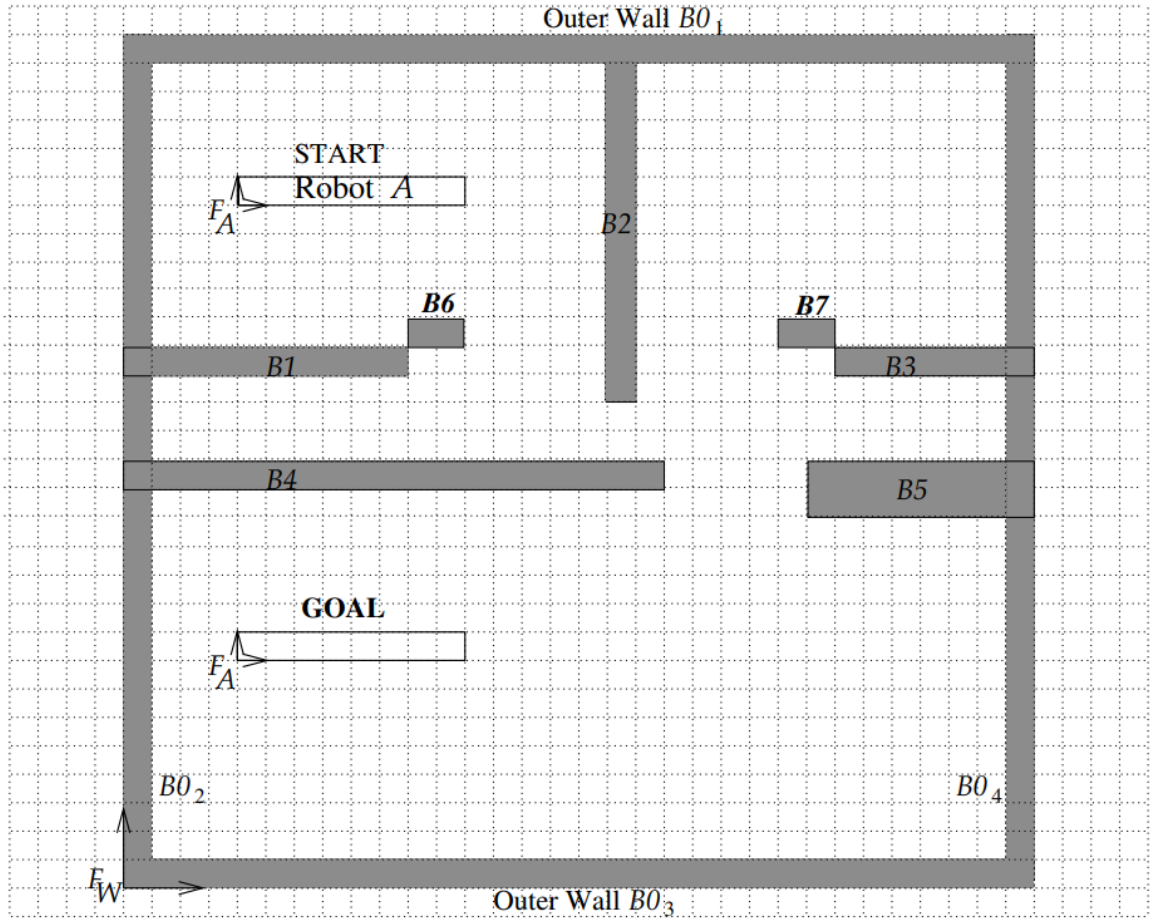
תוכן עניינים

מבוא – חלק 1

נתון:
רובוט מישורי ומכשולים במישור. הרובוט A פוליגון קמור והמכשול הכולל B_{tot} הינו פוליגון. אפשר לפרק את המכשול B_{tot} למכשולים B_i שהינם פוליגונים קמורים.

המטרה:
לבנות קוד להשגת ה-C-Obstacle בין הרובוט הקמור <u>באוריינטציה מסויימת</u> למכשול B_i קמור, $CB_{i,\theta=\theta_0}$, עבור זוויות שונות של הרובוט. לאחר מכן, להיעזר בקוד זה בשביל לקבל את ה-$CB_{\theta=\theta_0}$ בין הרובוט **לכל** המכשולים במישור B_{tot} ועם זה להשיג את ה-CB הכולל בין הרובוט למכשולים.
המכשול הכולל (הדירה) והרובוט מופיעים באיור 1.
את הקוד איתו עבדנו נוסיף בתור קבצי MATLAB.

הרובוט ומפת החדר בה צריך לנווט את הרובוט, נתונים ב-grid ריבועי

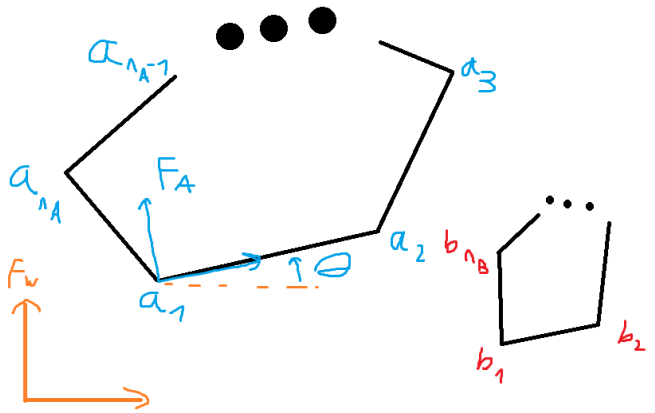


איור 1: ציור עם grid ריבועי הכולל את הרובוט A, החדר בו צריך לנווט את הרובוט והגדרת מ"צ העולם ומ"צ צמודת הרובוט.

סעיף 1 –קוד לקבלת ה-C-Obstacle בין רובוט ומכשול פוליגונים קמורים

תיאור כללי למטרת הקוד

הגדרת קודקודי הרובוט והמכשול הקמורים ומערכות הצירים, לחישוב ה-C-obstacle



איור 2: הגדרת הרובוט הקמור, המכשול הקמור, זווית הרובוט, קודקודיהם ומערכות הצירים.

המטרה כרגע היא לבנות קוד ליצירת ה-$CB_{\theta=\theta_0}$ לרובוט A פוליגון קמור עם מכשול B פוליגון קמור מסוים.

כניסות, כפי שמתואר באיור 2:

- קודקודי מכשול B, ביחס למערכת עולם F_W
- קודקודי הרובוט A, ביחס לראשית מ"צ F_A צמודת גוף, ביחס ל-F_W
- אוריינטציית הרובוט A, שנתונה בתור הזווית $\theta = \theta_0$ של מ"צ F_A

יציאות:

קודקודי ה-$CB_{\theta=\theta_0}$, בסדר הנכון, איתו אפשר לצייר את המעטפת של ה-C-Obstacle ולמעשה לקבל את כל האיזור במישור השייך ל-$CB_{\theta=\theta_0}$.

רעיון מרכזי:

מתכונות ה-CB, מפני שהגופים **פוליגונים קמורים**, מתקיים ש-$CB_{\theta=\theta_0}$ **<u>פוליגון קמור</u>** לכל θ_0.
זאת הסיבה שהשגת הקודקודים של $CB_{\theta=\theta_0}$ הינה מספיקה לידיעת <u>כל</u> שטח $CB_{\theta=\theta_0}$.

תיאור הקוד

הפונקציה איתה נעבוד נקראת [max_vert,Layers_vert] = plot_calc_CB(A_pts, B_pts,N,dA).

פונקציה plot_calc_CB מחשבת את ה-C-Obstacle בין הרובוט לחדר, עבור כל זווית בנפרד.

הקוד מקבל את הכניסות הבאות:

$\mathrm{A_{pts}} = \left[\boldsymbol{0}, \boldsymbol{a}_{2/1}, \dots, \boldsymbol{a}_{n_A/1}\right]_{2xn_A}$ – קודקודי הרובוט A, מסומנים בכיוון CCW מראשית מ"צ F_A. הנקודות מתארות את מיקום קודקודי פוליגון הרובוט ביחס ל-$\boldsymbol{a}_1$, במ"צ צמודת רובוט F_A.

זווית הסיבוב של F_A מסומנת בתור θ ומתארת את סיבוב הצלע $\boldsymbol{a}_1 - \boldsymbol{a}_2$ של הפוליגון.

$\mathrm{B_{pts}} = \left[\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_{n_B}\right]_{2xn_B}$ – קודקודי המכשול B, מסומנות בכיוון CCW מקודקוד $\boldsymbol{b}_1$, מתוארים במערכת העולם F_W.

N – מספר השכבות הרצויות לפלט.

d_A – מיקום התחלתי של קודקוד $\boldsymbol{a}_1$, בחלק זה של הפרויקטון לצורך ציורי בלבד

היציאות:

$\mathrm{Laters_{vert}}(2:(n_A+n_B):N)$ - מוציא את הקודקודים של אוסף $CB_{\theta=\theta_0}$ עבור θ שרץ ב-N צעדים לביצוע הקפה שלמה של הרובוט.

$\mathrm{max_{vert}}(1:N)$ - מוציא את מספר הקודקודים של כל פוליגון בשכבות ה-CB.

פירוט השלבים של הפונקציה [max_vert,Layers_vert] = plot_calc_CB(A_pts, B_pts,N,dA):

הקוד מגדיר את הזוויות עליהם מחושב ה-$CB_{\theta=\theta_0}$ בתור $\theta = 0:\frac{2\pi}{N}:2\pi - \frac{2\pi}{N}$.

לכל זווית הקוד מחשב את מיקום קודקודי הרובוט ביחס ל-$\boldsymbol{a}_1$, כלומר הוא מסובב את הרובוט.

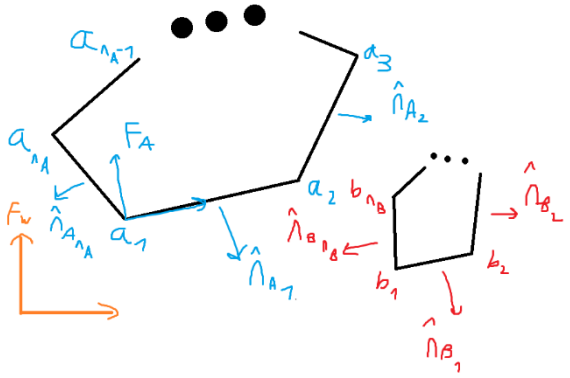
לאחר מכן, לכל θ הוא קורא לתת פונקציה בשם calc_CB_th_0 שמחשבת את הקודקודים של $CB_{\theta=\theta_0}$, ושומרת את אותם ומספר הקודקודים בתור היציאות של הקוד.

בנוסף, הקוד מדפיס את השכבות הנדרשות בבדיקה ומצייר ב-3D את כל N השכבות אחת על השנייה במרחב (d_x, d_y, θ).

תת הפונקציה (vertices = calc_CB_th_0(A_pts, B_pts):

בהינתן קודקודי המכשול ביחס למערכת העולם וקודקודי הרובוט ביחס ל-F_A במערכת העולם, מחשבת את קודקודי ה-$CB_{\theta=\theta_0}$ בשיטת מיזוג המעגלים (מחשבת את ה-C-Obstacle לזווית מסוימת).

הגדרת הנורמלים המוכללים של הרובוט והמכשול הקמורים



איור 3: הגדרת הנורמלים המוכללים לרובוט ולמכשול, עבור מציאת קודקודי גבול ה-C-obstacle ביניהם.

מהקודקודים הפונקציה מחשבת את הנורמלים, בהתאם לסדר באיור. בין כל זוג נורמלים היא מחשבת את תחום הזוויות של הסגמנט, בו ניעזר להשגת הקודקודים $\boldsymbol{d}_{ij} = \boldsymbol{b}_j - \boldsymbol{a}_i$, ושומרת את הזוויות כך:

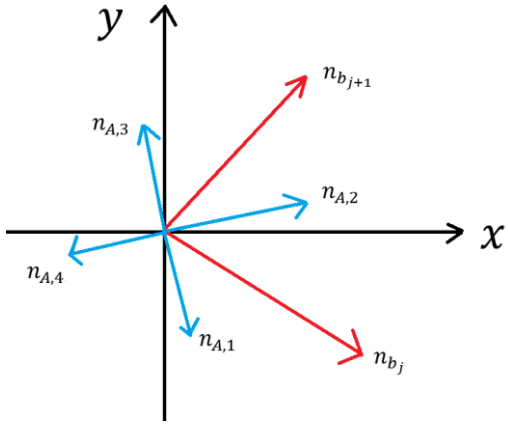
$$\text{angles}_{n_A} = \left[\begin{pmatrix} ang(\hat{\boldsymbol{n}}_{A,n_A}) \\ ang(\hat{\boldsymbol{n}}_{A,1}) \end{pmatrix}, \begin{pmatrix} ang(\hat{\boldsymbol{n}}_{A,1}) \\ ang(\hat{\boldsymbol{n}}_{A,2}) \end{pmatrix}, \dots, \begin{pmatrix} ang(\hat{\boldsymbol{n}}_{A,n_A-1}) \\ ang(\hat{\boldsymbol{n}}_{A,n_A}) \end{pmatrix}\right]$$

ויש מטריצה דומה למכשול B, angles_{n_B}.

כדי לחשב את קודקודי ה-C-Obstacle, הקוד עובר על החפיפות בין הסגמנטים של הנורמלים המוכללים של קודקודי הרובוט והמכשול ומאכסן את האינדקסים לחישוב הקודקודים בסדר המתאים לציור הפוליגון של $CB_{\theta=\theta_0}$. זה מתבצע בסדר הבא:

- מתחילים עם קודקוד במכשול $b_j \overset{j=1}{\leftrightarrows} b_1$.
- עוברים על כל הקודקודים של הרובוט $a_i, i = 1 \to n_A$.
  - אם יש סגמנט של הרובוט לו חיתוך עם סגמנט b_1, שומרים אותו בנפרד. ז"א, מוצאים את כל הסגמנטים של קודקודי הרובוט a_i בעלי חיתוך עם סגמנט b_1.
  - אז מתחילים את הכנסת האינדקסים לשמירת $\boldsymbol{d}_{ij}$-ים החל מהסגמנט עם הזווית ההתחלתית הכי קרובה לזווית ההתחלתית של b_1, עבור בדיקת חיתוכים בכיוון CCW.
- לאחר מכן, עוברים על שאר קודקודי המכשול עבור $j: 2 \to n_B$ באותה איטרציה לכל קודקוד מכשול.
- לבסוף, אפשר לחשב את קודקודי ה-$CB_{\theta=\theta_0}$ מהאינדקסים i, j ע"י $\boldsymbol{d}_{ij}$.

הסבר למעבר נבחר על הנורמלים המוכללים ליצירת ה-C-obstacle



איור 4: מעבר על הנורמלים המוכללים בשיטת מיזוג המעגלים: עבור הסבר השגת קודקודי גבול ה-C-Obstacle בסדר הנכון

באיור הנ"ל ניתן לראות איך תיעוד חיתוך הסגמנטים עובד: עבור סגמנט של b_j: $n_{b_j} \to n_{b_{j+1}}$, שומרים אינדסקים של נורמלי A שבתחום הסגמנט מ-b_j עד b_{j+1}.

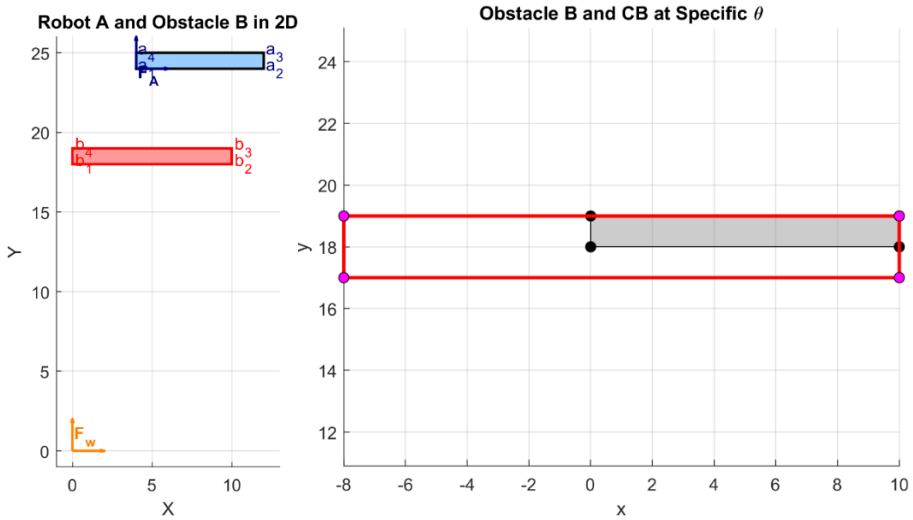
התמודדות עם ניוון במספר הקודקודים במיזוג מעגלים

בנוסף, קיימים מקרים בהם אחת מצלעות הרובוט מקבילה לצלע המכשול, אז מתקבלים נקודות מנוונות בחישוב ה-$\boldsymbol{d}_{ij}$-ים שעבורן לפחות 3 או יותר נקודות מהפוליגון המתקבל על אותו ישר. כדי לטפל בבעיה זאת, הקוד עובר על כל הפוליגון המחושב ומוריד חלק מהנקודות עבורן שתי זוגות נקודות סמוכות על אותו ישר. לכן, **מספר הקודקודים אינו תמיד** $n_A + n_B$, מודבר רק בחסם עליון אחרי הסרת נקודות מנוונות.

דוגמת פלט נדרשת עבור 32 שכבות בזווית

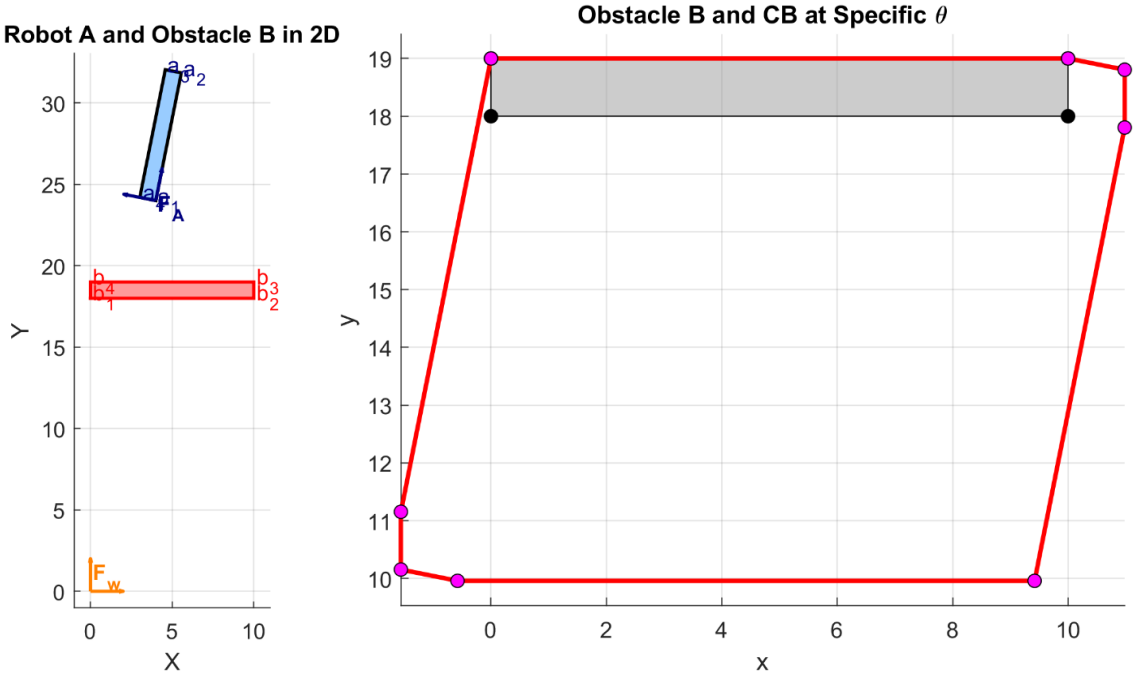
בעזרת הקוד שבנינו ופונקציות עזר נוספות, נדפיס את השכבות של CB עבור הרובוט A והמכשול B_1:

רובוט, מכשול ושכבת ה-C-obstacle עבור שכבה 1



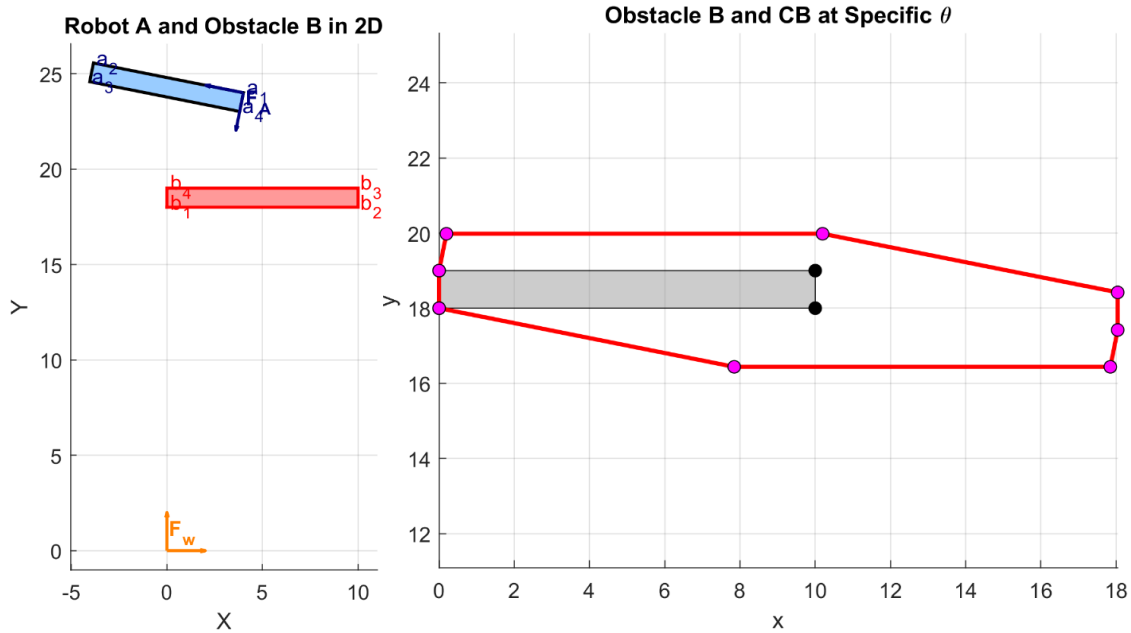
איור 5: משמאל, הרובוט בכחול והמכשול באדום, עם הזווית עבור שכבה 1.
מימין, גבול ה-C-Obstacle בין הרובוט למכשול לאוריינטציה זאת של הרובוט.

רובוט, מכשול ושכבת ה-C-obstacle עבור שכבה 8



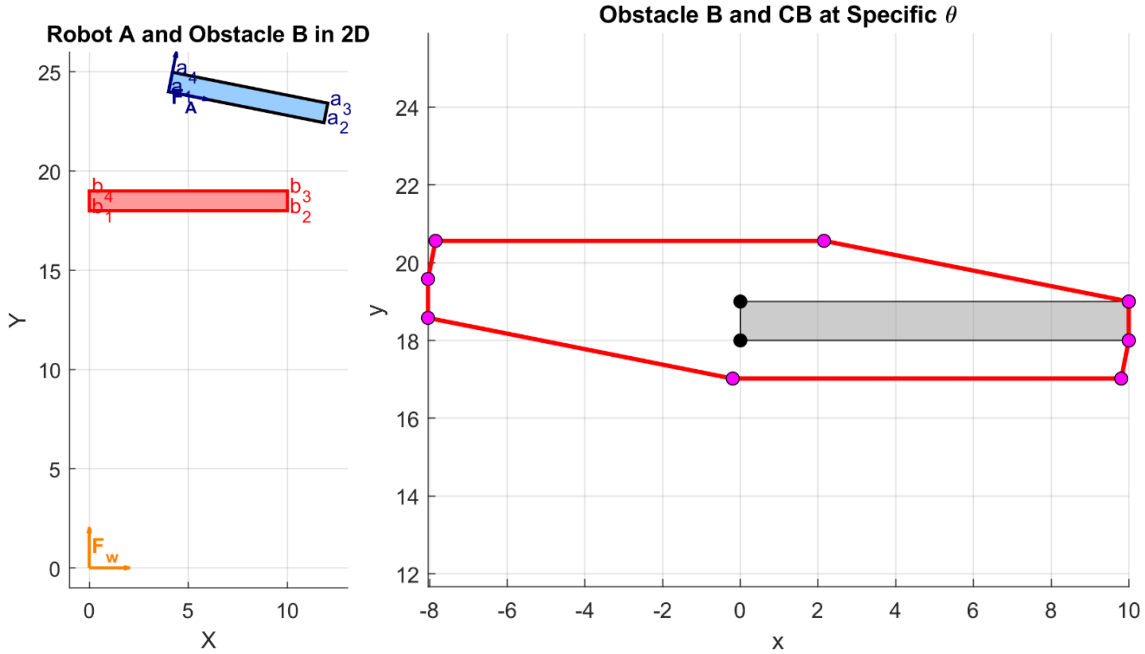
איור 6: משמאל, הרובוט בכחול והמכשול באדום, עם הזווית עבור שכבה 8.
מימין, גבול ה-C-Obstacle בין הרובוט למכשול לאוריינטציה זאת של הרובוט.

רובוט, מכשול ושכבת ה-C-obstacle עבור שכבה 16



איור 7: משמאל, הרובוט בכחול והמכשול באדום, עם הזווית עבור שכבה 16.
מימין, גבול ה-C-Obstacle בין הרובוט למכשול לאוריינטציה זאת של הרובוט.

רובוט, מכשול ושכבת ה-C-obstacle עבור שכבה 32



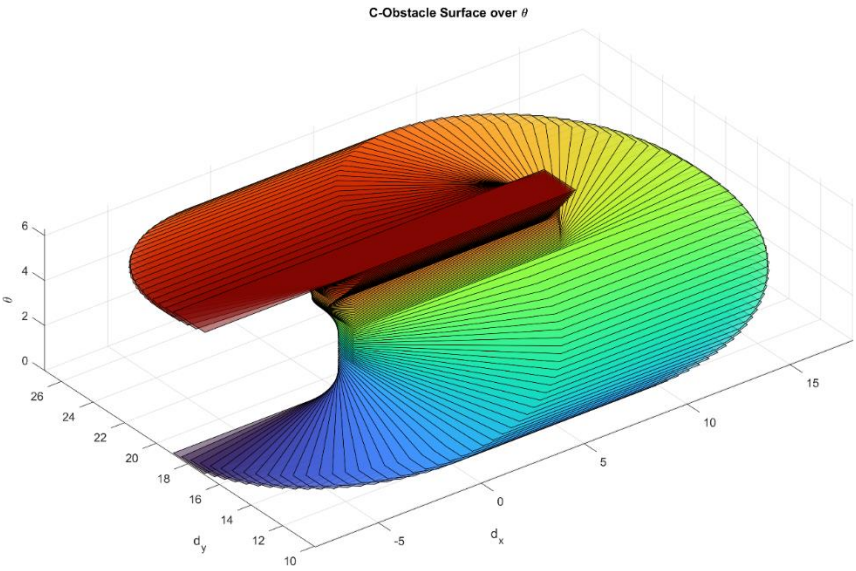
איור 8: משמאל, הרובוט בכחול והמכשול באדום, עם הזווית עבור שכבה 32.
מימין, גבול ה-C-Obstacle בין הרובוט למכשול לאוריינטציה זאת של הרובוט.

כאשר ערכי הזוויות הינם, עם צעד $\text{step} = \frac{2\pi}{32}(\text{rad}) = 11.25°$:

$$\theta = \{0°, 78.75°, 168.75°, 348.75°\}$$

גיאומטרית רואים שעבור $\theta = 0,90°, 180°, 270°$ צלעות הרובוט מקבילות לצלעות המכשול ולכן יש ל-$CB_{\theta=\theta_0}$ רק 4 קודקודים. לשכבות האחרות יש 8 קודקודים כמצופה מהמספר הנפוץ $n_A + n_B = 4$.

בנוסף, אפשר לקבל את ציור ה-CB כתלות במספר השכבות. עבור $N = 100$



איור 9: ה-C-Obstacle מודפס עבור 100 שכבות של זווית סיבוב. רואים את "התולעת המשובללת" שציפינו לקבל בין רובוט למכשול מלבנים.

רואים את ה-"תולעת המשובללת", באיור 9.

סעיף 2 – יישום הקוד לדוגמת הדירה

בחלק זה, נתמקד בבניית שתי פונקציות:

1. פונקציה אחת לאיחוד שתי פוליגונים קמורים. כאשר אם הם זרים אז הפונקציה מחזירה מערך ריק.
2. פונקציה שנייה שמקבלת n_p פוליגונים ומחזירה מערך של כל קבוצות פוליגונים זרות שיש להם חיתוך, עם מספר הקודקודים לכל קבוצה בהתאמה.

בעזרת פונקציות אלו, אפשר באופן איטרטיבי לקבל לכל שכבה את האיחוד של ה-C-Obstacle של כל מכשול ולקבל את ה-$CB_{\theta=\theta_0}$ הכולל לאותה שכבה בהתאמה.
הפונקציה למציאת איחוד בין שתי פוליגונים קמורים:

union_poly = intersect_two_polygons(P1,P1)

הקוד מקבל שתי פוליגונים בצורה P_i בתור משתנה POLYSHAPE

- נעשה שימוש במשתנה זה מעכשיו מפני שהוא יודע להתמודד בצורה נוחה עם חורים בפוליגון.

ומחזיר את האיחוד של שתי הפוליגונים בצורה union_poly שהוא גם POLYSHAPE.

הפונקציה הזאת נעזרת במשתנה polyshape והפונקציות המובנות intersect, union כדי למצוא את האיחוד.

הפונקציה למציאת האיחוד בין כמה פוליגונים נקראת: union_Plist = union_polyogns(P_list)

הקוד מקבל את הכניסות הבאות:

$\mathrm{P_{list}}\{n_p x1\}$ – קודקודים, במ"צ עולם F_W, של כל פוליגון קמור שרוצים לאחד. מספר הפוליגונים הינו n_p.
השימוש ב-cell מאפשר גמישות בכמות הקודקודים של כל פוליגון.

היציאות:

union_Plist$(n_F x1)$ - מוציא את הקודקודים של כל n_F קבוצות הפוליגונים שאפשר לאחד ואינם זרים.
באופן כללי אפשר לקבל באיחוד כמה פוליגונים, נראה בהמשך שזה לא המקרה בבעית הדירה.

<u>פירוט השלבים של הפונקציה</u>:

הפונקציה משתמשת במערך שנקרא active $= (1xn_P)$ שמאותחל ככולו True.

היא עוברת בלולאה של שני for-ים אחד בתוך השני וברגע שהיא מוצעת זוג עם חיתוך ביניהם, היא מחליפה את הראשון באיחוד ואת השני היא משאיר אבל מורידה ממנו את ה-active במערך הנ"ל. בנוסף, משתנה changed מחליף סימן ל-True כך שכל הלולאה מתחילה עוד הפעם.

כך לא מפספסים איחודים!

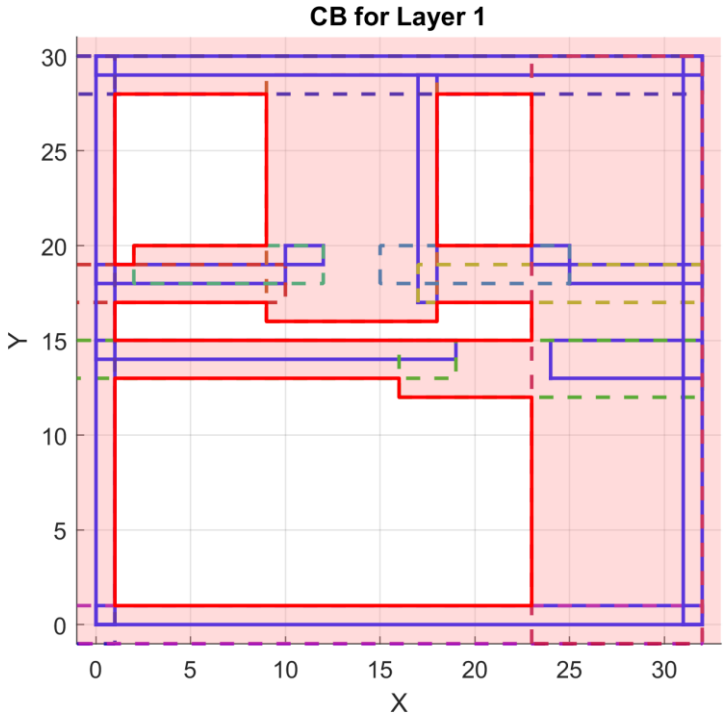
אחרי מעבר על כל הזוגות, מתקבל מערך $\mathrm{P_{list}}$ פנימי בו כל האיחודים הרלוונטים נשמרו, מי שלא active זה רק פוליגונים שכבר נכנסו לאיחוד עם אחרים. מנתונים אלו גוזרים את היציאות הרלוונטיות.

<u>מטרה</u>:

לכל זווית מאחדים את כל ה-$CB_{i,\theta=\theta_0}$ ל-$CB_{tot,\theta=\theta_0}$, כך שמתקבלים כל הפוליגונים שאפשר לאחד ביחד.
בבעית הניווט של הפרויקט מתקיים ש-$n_F = 1$ כי ה-$CB_{\theta=\theta_0}$ הוא פוליגון אחד קשיר עם חורים, לכל זווית.

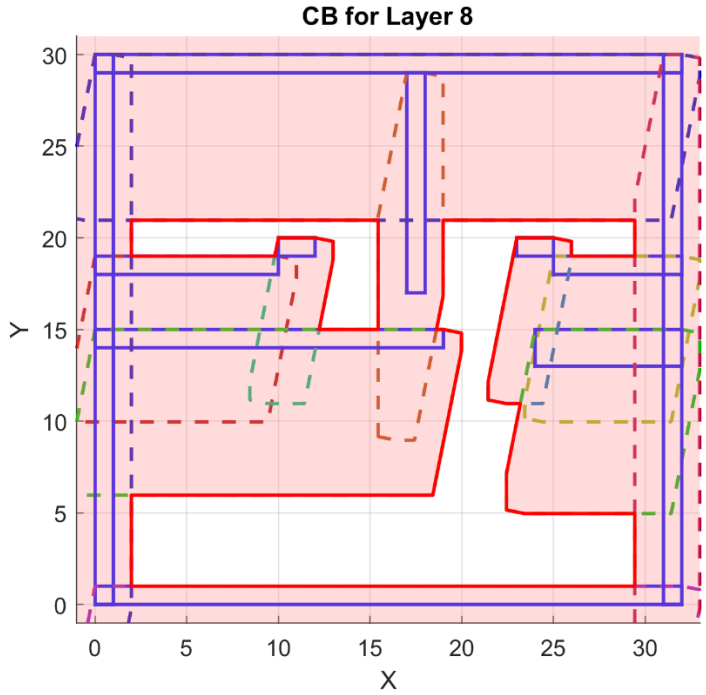
קיבלנו את השכבות הבאות, כאשר בכחול מופיע קירות הדירה:

שכבת ה-C-obstacle המלאה עבור שכבה 1



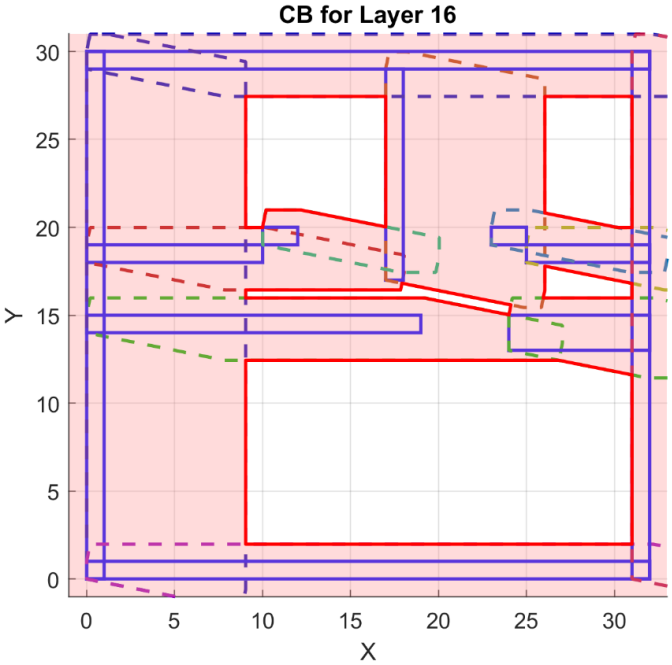
איור 10: גבול ה-C-Obstacle בין הרובוט לדירה לאוריינטציה של הרובוט בשכבה 1.

שכבת ה-C-obstacle המלאה עבור שכבה 8



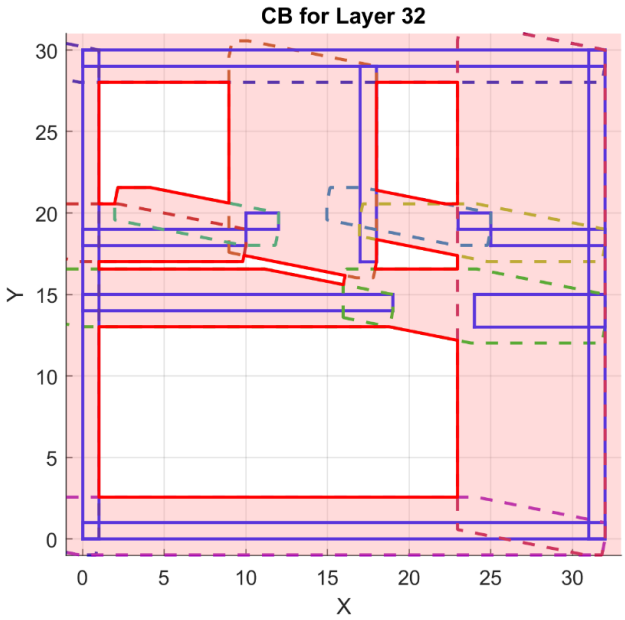
איור 11: גבול ה-C-Obstacle בין הרובוט לדירה לאוריינטציה של הרובוט בשכבה 8.

שכבת ה-C-obstacle המלאה עבור שכבה 16



איור 12: גבול ה-C-Obstacle בין הרובוט לדירה לאוריינטציה של הרובוט בשכבה 16.

שכבת ה-C-obstacle המלאה עבור שכבה 32



איור 13: גבול ה-C-Obstacle בין הרובוט לדירה לאוריינטציה של הרובוט בשכבה 32.

כאשר באדום מופיע איזור ה-CB_{tot} הכולל לאוריינטציה של הרובוט בשכבה, בקווים כחולים רציפים מופיעים המכשולים ובקווים מקווקוים בצבעים שונים אפשר לראות את ה-$CB_{\theta=\theta_0}$ שמשרה כל מכשול.

מראש הוספנו מלבן עם חור בצורה של החלל בו פנים הדירה כולה, בשביל שה-CB_{tot} ידגיש שהרובוט תקוע בתוך הדירה בבעיה הזאת.

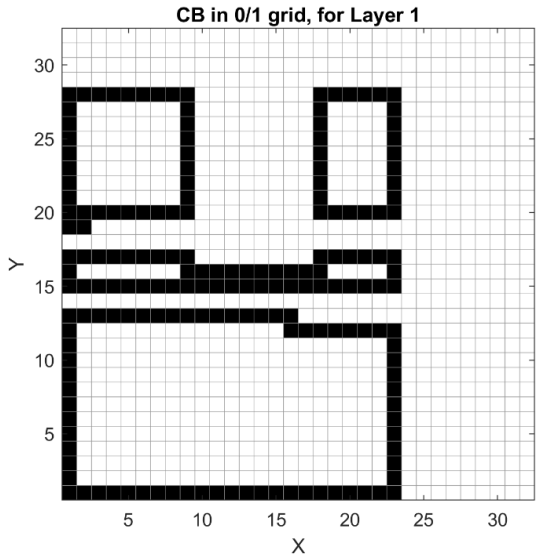
סעיף 3 – ציור המכשול במרחב הקונפיגורציה ברשת ריבועית עם 0,1

מהקוד של החלק הקודם, קיבלנו בצורה של PolyShape את ה-CB_{tot} לכל שכבה/אוריינטציה בנפרד.
פה אנחנו לוקחים את הנקודות של הגבולות הפנימיים, החורים, של ה-CB_{tot} לאותה השכבה, ובין כל 2 נקודות מציירים קו דיסקרטי, בעזרת אלגוריתם Bresenham's שיודע לתת קו דיסקרטי רציף.

הערה על הדיסקרטיזציה: בחלק ה-2 של הפרויקטון מצאנו שהדיסקרטיזציה בשיטה זאת שמרנית מידי וכפי שניתן לראות בשכבה 1, בחלק 1 של הפרויקטון, לא מאפשרת לנווט למטרה. נתקן זאת בחלק ה-2.

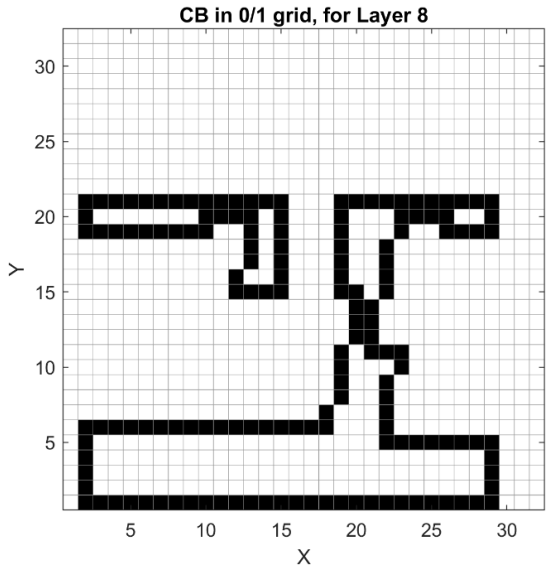
קיבלנו:

שכבת ה-C-obstacle המלאה עבור שכבה 1, אחרי דיסקרטיזציה של שיטה פה



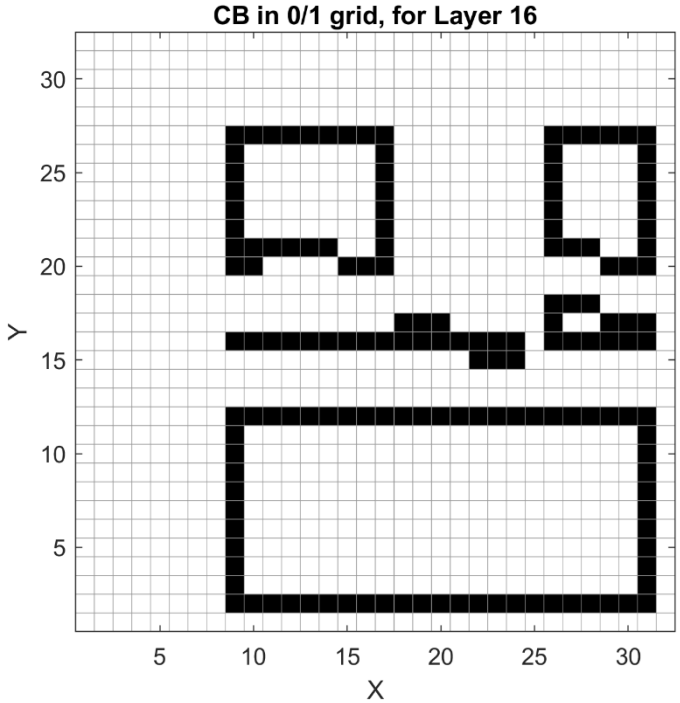
איור 14: גבול ה-C-Obstacle בין הרובוט לדירה לאוריינטציה של הרובוט בשכבה 1, אחרי דיסקרטיזציה לרזולוציה של 1x1.

שכבת ה-C-obstacle המלאה עבור שכבה 8



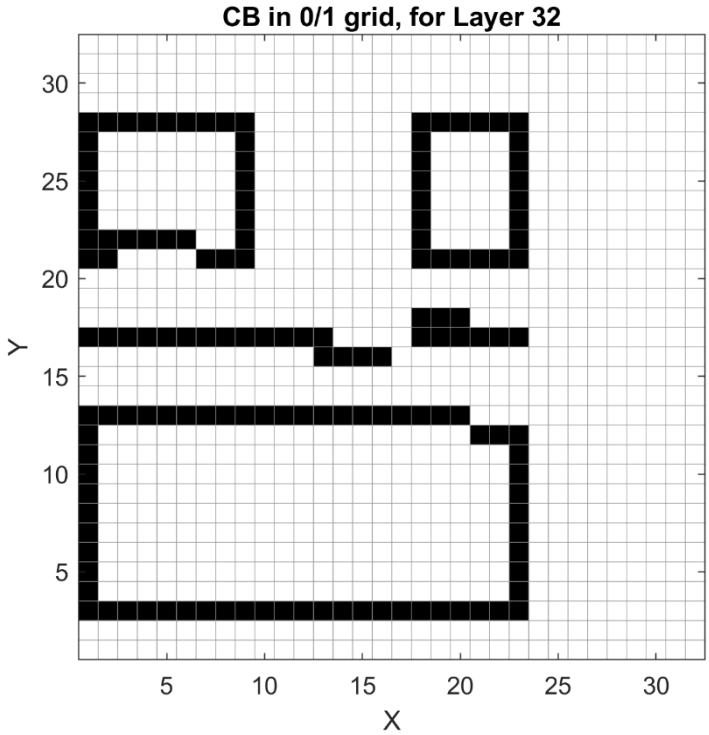
איור 15: גבול ה-C-Obstacle בין הרובוט לדירה לאוריינטציה של הרובוט בשכבה 8, אחרי דיסקרטיזציה לרזולוציה של 1x1.

שכבת ה-C-obstacle המלאה עבור שכבה 16



איור 16: גבול ה-C-Obstacle בין הרובוט לדירה לאוריינטציה של הרובוט בשכבה 16, אחרי דיסקרטיזציה לרזולוציה של 1x1.

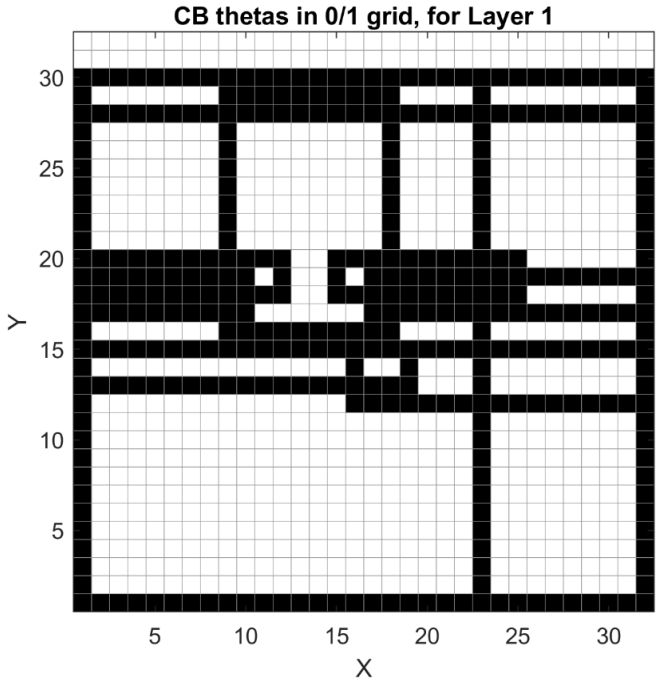
שכבת ה-C-obstacle המלאה עבור שכבה 32



איור 17: גבול ה-C-Obstacle בין הרובוט לדירה לאוריינטציה של הרובוט בשכבה 32, אחרי דיסקרטיזציה לרזולוציה של 1x1.

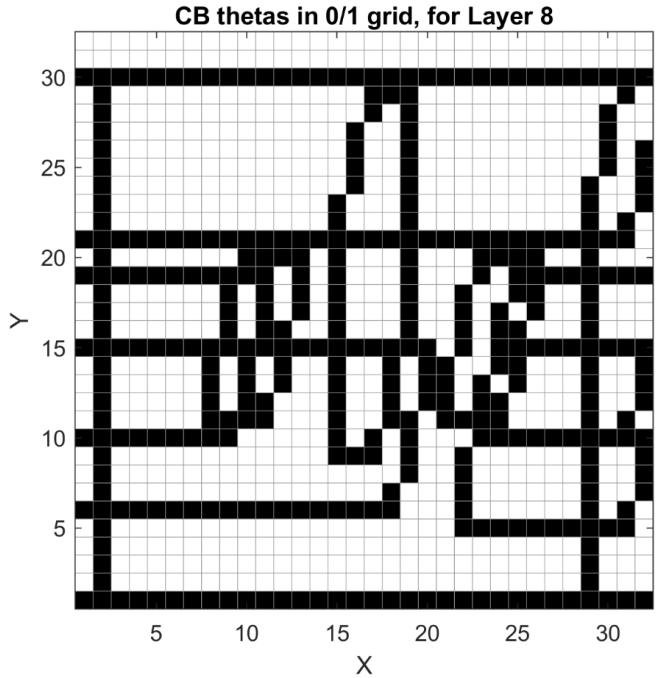
הבנו אחרי זה שהכוונה זה לצייר את הגבולות של כל C-Obstacle בנפרד. למרות שנעבוד עם ה-CB הכולל בפרויקטון.

שכבת ה-C-obstacle לפי גבולות כל מכשול עבור שכבה 1



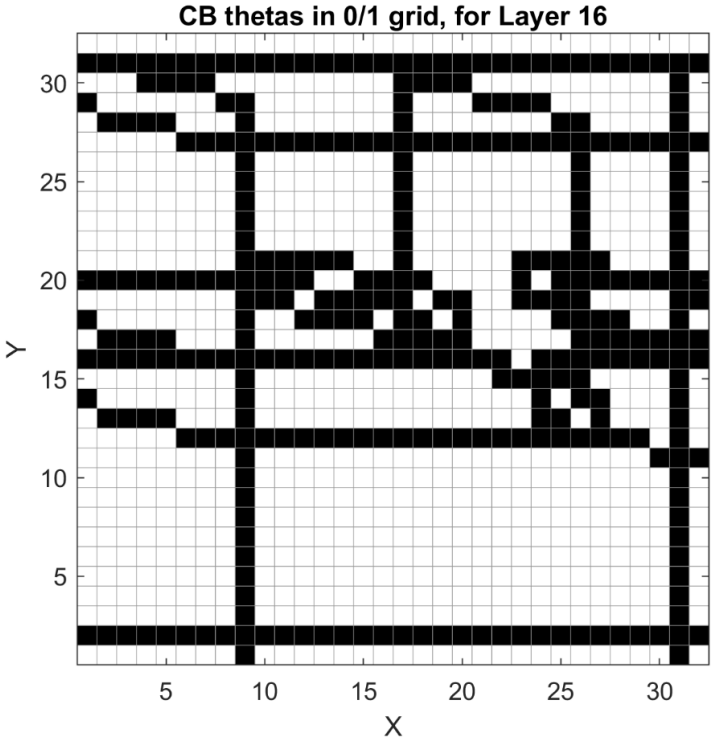
איור 18: גבול כל ה-C-Obstacle בין הרובוט **לחלקי הדירה** לאוריינטציה של הרובוט בשכבה 1, אחרי דיסקרטיזציה לרזולוציה של 1x1.

שכבת ה-C-obstacle לפי גבולות כל מכשול עבור שכבה 8



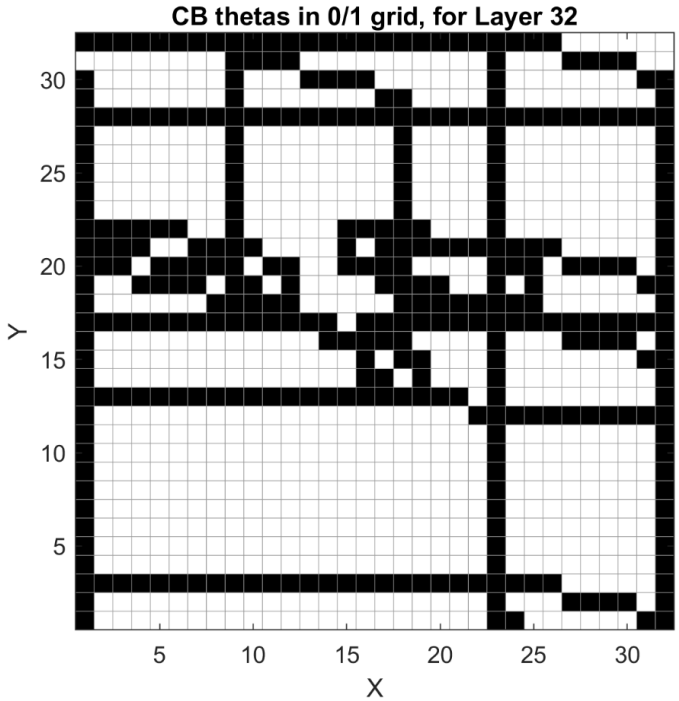
איור 19: גבול כל ה-C-Obstacle בין הרובוט **לחלקי הדירה** לאוריינטציה של הרובוט בשכבה 8, אחרי דיסקרטיזציה לרזולוציה של 1x1.

שכבת ה-C-obstacle לפי גבולות כל מכשול עבור שכבה 16



איור 20: גבול כל ה-C-Obstacle בין הרובוט **לחלקי הדירה** לאוריינטציה של הרובוט בשכבה 16, אחרי דיסקרטיזציה לרזולוציה של 1x1.

שכבת ה-C-obstacle לפי גבולות כל מכשול עבור שכבה 32



איור 21: גבול כל ה-C-Obstacle בין הרובוט **לחלקי הדירה** לאוריינטציה של הרובוט בשכבה 32, אחרי דיסקרטיזציה לרזולוציה של 1x1.

מבוא – חלק 2 של הפרויקט

מטרת חלק 2 של הפרוייקטון – ניווט גוף קשיח
מטרתינו היא לתכנן את מסלול התונעה של הרובוט ממיקום+אוריינטציית ההתחלה לסיום, תוך ביצוע הצעדים הבאים:

1. נבחר שיטת ניווט אחת מאלו במטלה.
2. נממש אותה תוך שימוש בחישוב של קודקודי ה-CB בעזרת הקוד מחלק 1.
3. נשרטט בעזרת מאטלאב הדפסה שמראה את תנועה הרובוט מהתחלה למטרה.

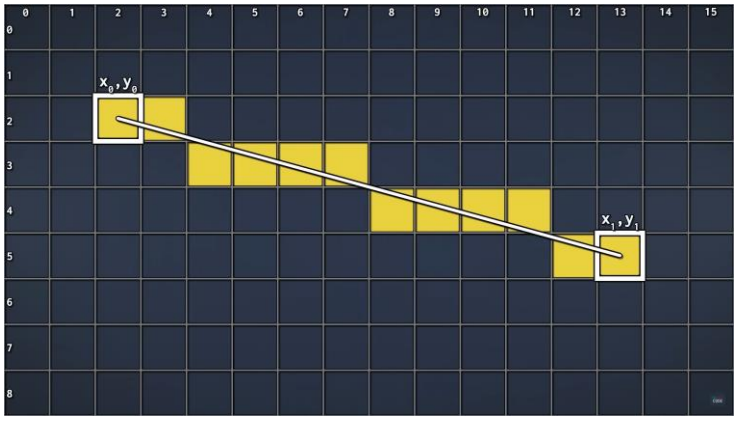
סעיף 1 – שיטת דיסקרטיזציה נבחרת ובחירת רזולוציה

אנחנו בוחרים לתכנן מסלול בעזרת אלגוריתם A^*, שיקבל עץ צמתים סופי עם ענפים בינייהם שמתארים את התנועה האפשרית של הרובוט במרחב החופשי בתוך הדירה, מבלי להיכנס בתוך הקירות **עם אפשרות "לגרד" את הקירות**, ויפתור עבור מסלול מ-S ל-T .

כעת, **שכבר יש לנו קוד שפותר עבור הגבולות הפנימיות** של $CB_{\theta=\theta_0}$, **צריך לבחור את הרזולוציה איתה נעשה דיסקרטיזציה למרחב החופשי בקואורדינטות** (x, y, θ).

שיטות דיסקרטיזציה בהם עבדנו

שיטת דיסקרטיזציה 1: יצירת קו בדיד בין שתי נקודות, רק מידיעת קודקודי הקצה

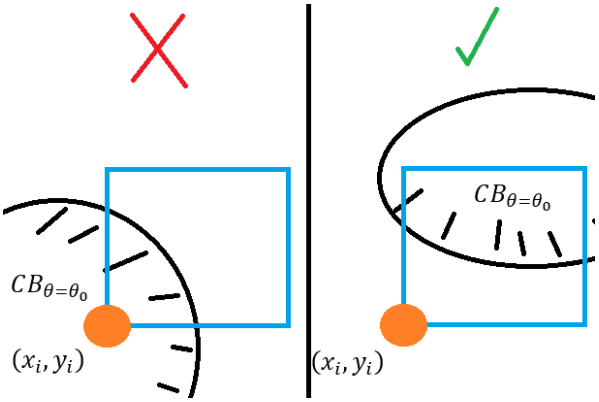


איור 22: תמונה להמחשת האלגוריתם לציור קו בדיד בין שתי נקודות (לאחר העגלת הנקודות לרזולוצית ה-Grid), מסרטונו של NoBS Code בקישור: https://www.youtube.com/watch?v=CceepU1vIKo.

שיטה 1: בחלק 1 של הפרויקט התמקדנו בדיסקרטיזציה פשוטה לישום, בה לכל זווית סימנו את הגבולות הפנימיים של ה-$CB_{\theta=\theta_0}$ ע"י לקיחת הקודקודים של גבולות האיזורים החופשיים בכל שכבה, והשתמשנו באלגוריתם Bresenham's להשגת קו דיסקרטי ביניהם, כפי שמתואר באיור 22.

שיטה 2: לאחר מכן, בגלל מעברים צרים בדירה, כמו המרווח של שני בלוקים 1x1 בין מכשול B4,B2, החלטנו לנסות גם דיסקרטיזציה בשיטה אחרת. בשיטה זאת לכל שכבה של אוריינטציה אנחנו **מסמנים את כל הפיסקלים בשכבה בהם יש חפיפה בין הקודקוד בו צמודה מ"צ F_A, שהוא מרכז הפיקסל/פינה שמאלית-תחתונה שלו** (עוד דרגת חופש לבחירתינו), **ובין $CB_{\theta=\theta_0}$**. ניתן לראות הסבר לקריטריון לזיהוי תאים חופשיים בפיקסל ה-(x_i, y_i) באיור 23. מפה אפשר להשיג את גבולות CB בלבד.

שיטת דיסקרטיזציה 2: בדיקת חופשיות הרובוט לכל פיקסל במפה

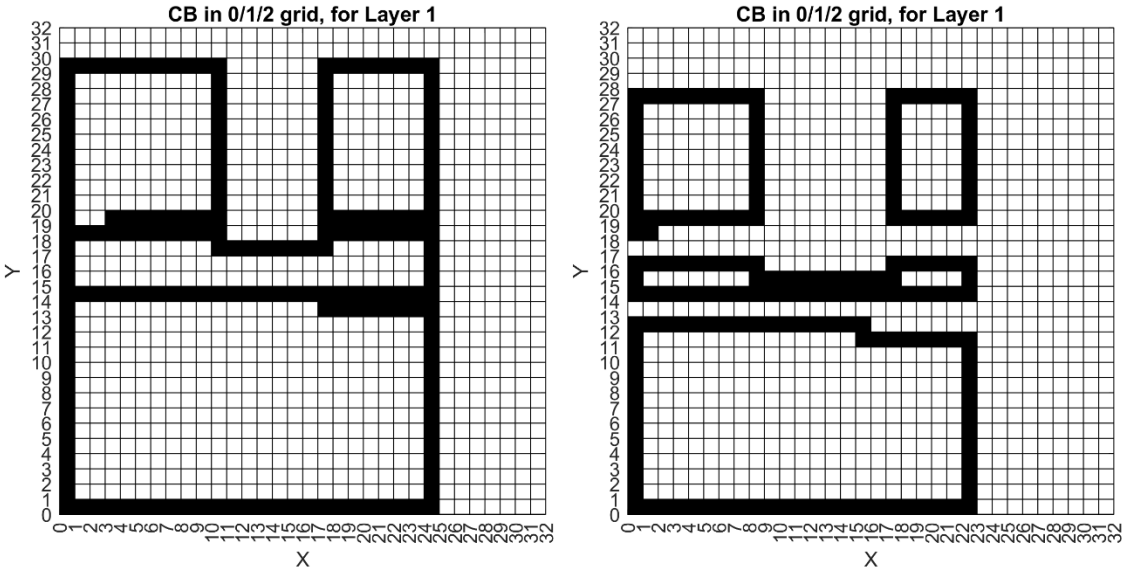


איור 23: בשיטה ה-2, בוחרים אם רוצים שהרובוט ינוע בין מרכזי הפיקסלים או בין כל פינה שמאלית-תחתונה של פיקסל, וליצירת המפה הדיסקרטית עוברים תא-תא ובודקים אם הקודקוד הנבחר ב-CB או באיזורים החופשיים הפנימיים של החדר.

בחירת שיטת דיסקרטיזציה

כעת אנחנו נראה דוגמה לשכבה בה רואים ששיטה 2 נותנת דיסקרטיזציה יותר מתאימה למרחב החופשי הפיזיקלי, באיור 24.

עבור השכבה ה-1 בחלוקה של 32 שכבות בזווית, הדיסקרטיזציה בשתי השיטות



איור 24: הדיסקרטיזציה של שכבה 1 (עבור 32 שכבות), כאשר מימין זאת שיטה 1 ומשמאל שיטה 2 לפי פינת הפיקסל.

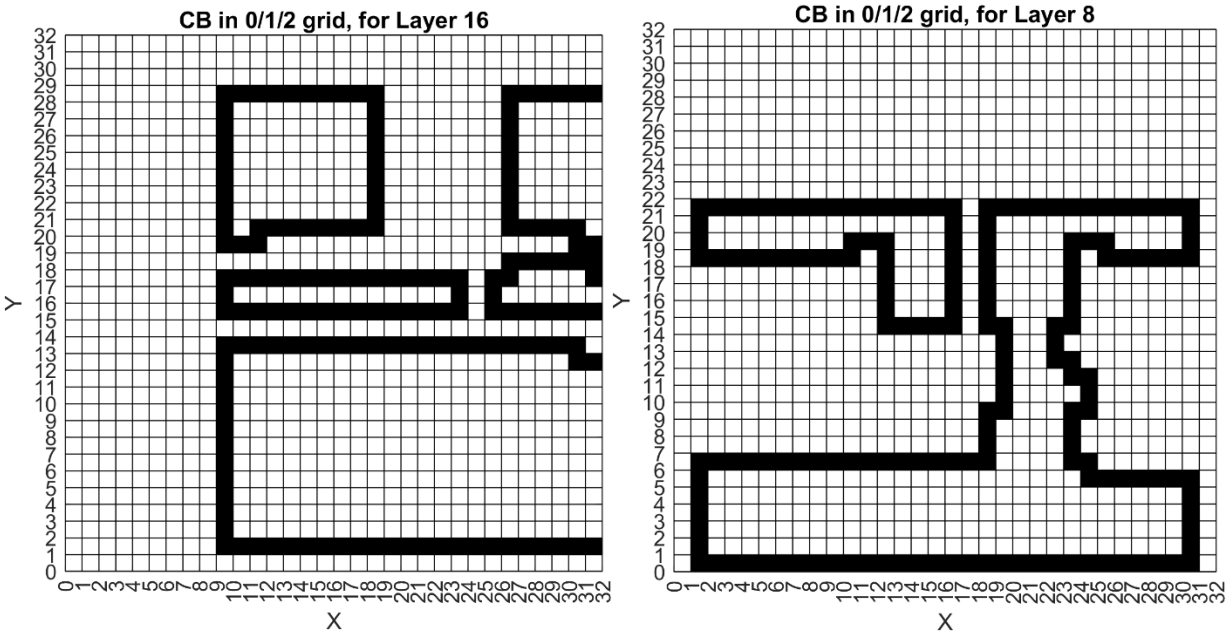
כאשר בשיטה ה-2 שפה אנחנו בודקים את חופשיות התא **לפי יכולת הימצאות קודקוד מ"צ F_A בפינה השמאלית-תחתונה של הפיקסל**.

רואים שהבדיקה של כל תא בשיטה 2 נותנת מפה יותר קרובה למרחב החופשי הפיזיקלי, שניתן לראות בחלק 1 בפרויקטון. שיטה 1 שמרנית ונותנת תאים במרחב החופשי אבל לא את כל התאים, כפי שניתן לראות בין מכשולים B2,B4. כדי להצליח לנווט ברזולוצית $x - y$ צריך לבנות מפה דיסקרטית הדוקה.

לכן, מאתה נעבוד עם שיטת הדיסקרטיזציה ה-2.

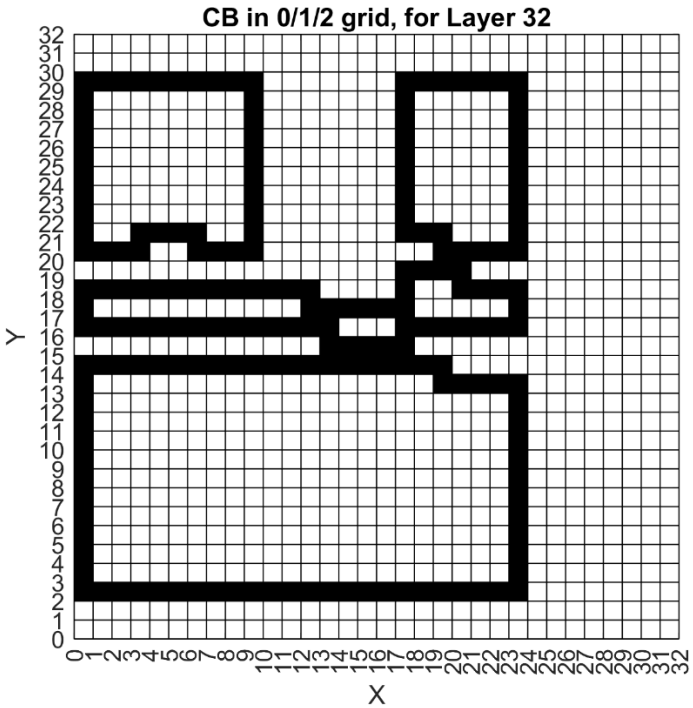
עכשיו, נראה את ההדפסה של השכבות של חלק 1 של הפרויקטון פה עם השיטה ה-2 עם התחשבות בפינת הפיקסל (חלוקה לסה"כ 32 שכבות, כאשר את שכבה 1 כבר הדפסנו באיור הנ"ל):

שכבת ה-C-obstacle המלאה עבור שכבה 8,16, עבור שיטה 2 לפינת הפיקסל



איור 25: הדיסקרטיזציה של שכבה 8 ושכבה 16 (עבור 32 שכבות), עבור שיטה 2 ופינת הפיקסל.

שכבת ה-C-obstacle המלאה עבור שכבה 32, עבור שיטה 2 לפינת הפיקסל

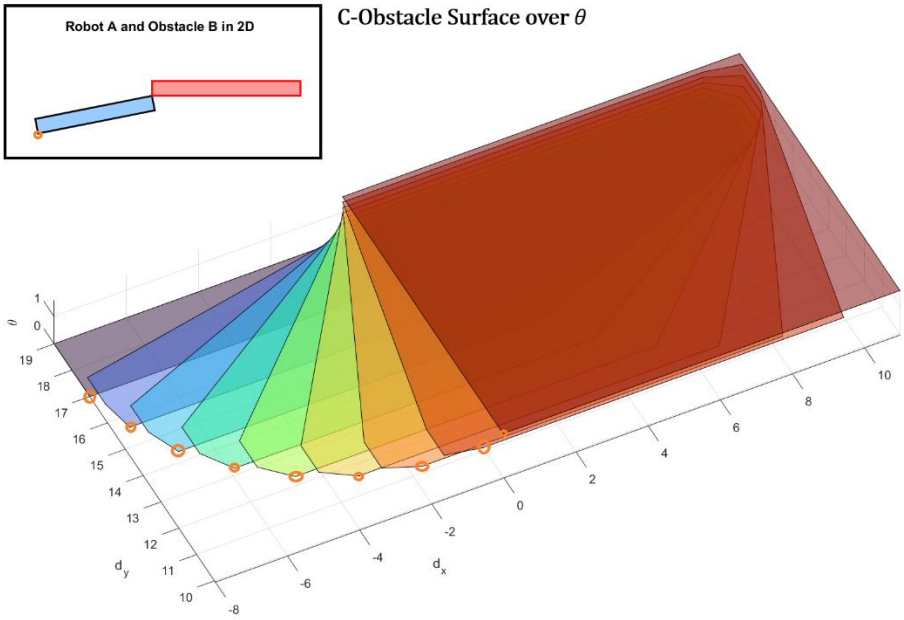


איור 26: הדיסקרטיזציה של שכבה 32 (עבור 32 שכבות), עבור שיטה 2 ופינת הפיקסל.

השפעת מספר השכבות על הדיסקרטיזציה

נדרשנו להגדיל את הרזולוציה בציר θ בלבד. כלומר, עלינו להבין את הבעיה שמופיע במספר שכבות נמוך ולנסח קריטריון למספר השכבות הנדרש $N_{\min}$. כדי להבין את השפעת הרזולוציה של השכבות, **נסתכל על תזוזת גבולות ה-C-obstacle בין הרובוט A למכשול B1**. בגלל הסימטריה בין הרובוט המלבני למכשול המלבני, נסתכל על תחום סיבוב $\theta: 0 \to 90°$.

ציור השכבות בתלת-מימד עבור חלוקה ל-$N = 32$ שכבות כולל, מ-0 מעלות עד 90 מעלות



איור 27: ציור שכבות CB בין הרובוט למכשלו מלבני יחיד B1 לחקירת השפעת הרזולוציה בזווית על תנועת גבולות ה-CB של החדר.

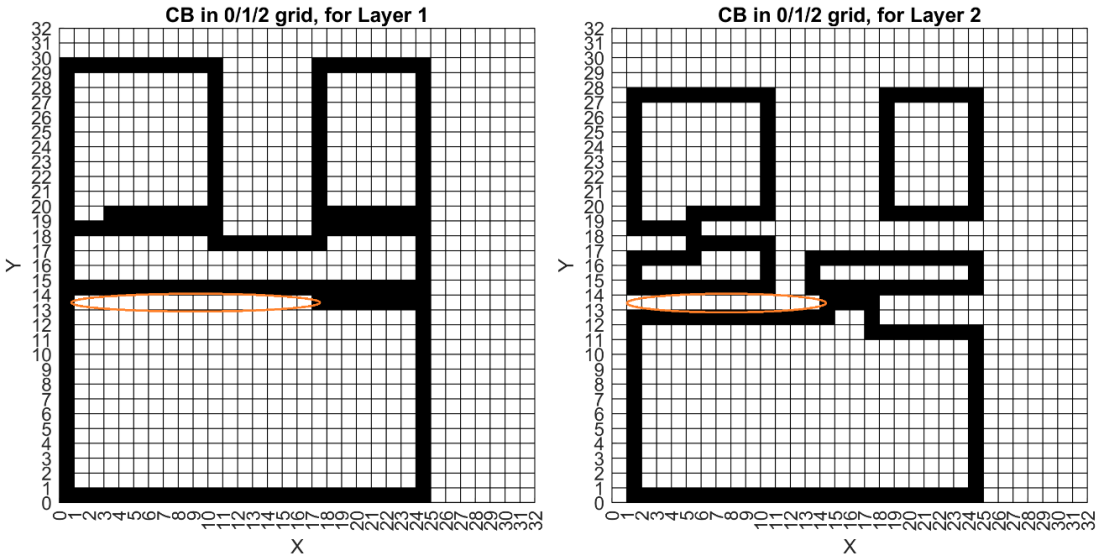
ניתן לראות שמעבר בין שתי זוויות עוקבות של אוריינטציית הרובוט "מסובב" את קודקודי גבולות ה-CB_{θ_0}.

ככל שנקטין את הרזולוציה בציר הסיבוב, כך המרחק שיעבור קודקוד מסוימם במישור $x - y$ בין שתי שכבות עוקבות יהיה יותר נמוך. הסיבה לרצות להגדיל את הרזולוציה נובעת מכך שברזולוציה נמוכה, קודקודי השכבה "מסתובבים" למרחק כה גדול, כך שגם גבולות שתי שכבות עוקבות מסתובבות הרבה ויכול לגרום לבעיה שמוצגת במטלה של חלק 2.

באיור 28 למשל, **אם נסמן בנפרד רק את הגבול של ה-C-obstacle** וננתח את הצמתים הלבנים המסומנים בשכבות 1,2, אנחנו עלולים להסיק שהצמתים הלבנים מעל בשכבה 2 הם **שכנים באיזור החופשי** לאילו בשכבה 1, ואפשר לגרום לרובוט לנווט בטעות לתוך המכשול!

לכן, כל עוד האיזור מחוץ לחורים הפנימיים של CB_{θ_0} בכל שכבה בצבע לבן, כמו באיזורים החופשיים, אנחנו מסיקים שיש לדאוג לרזולוציה ב-θ מספיק טובה כדי שכל גבול יקפוץ לכל היותר תא אחד בין שכבות עוקבות.

שתי שכבות עוקבות בדיסקרטיזציה של שיטה 2 (עבור פינת הפיקסל), עבור 32 שכבות

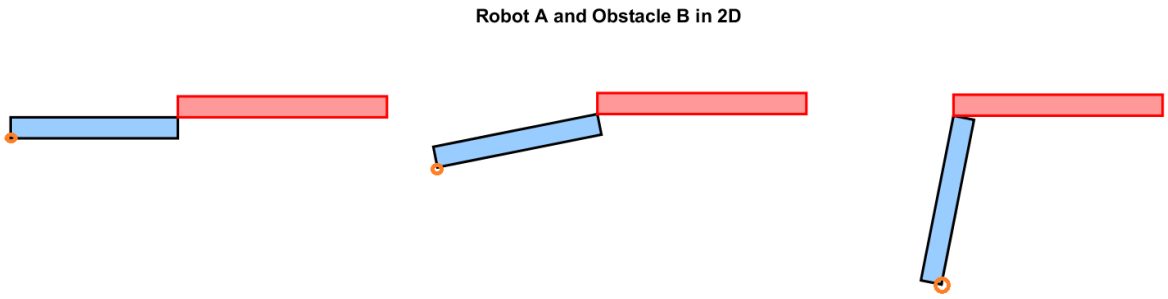


איור 28: שתי שכבות עוקבות בדיסקרטיזציה של שיטה 2, עבור בדיקת חופשיות פינת הפיקסל, עבור 32 שכבות בכולל.

קריטריון למספר שכבות מספיק

כעת, נוכל להגדיר קריטריון אלגברי למספר שכבות מספיק. אנחנו מבחינים שההתקדמות של גבולות ה-CB בין שכבות עוקבות תלוי **במימדי הרובוט בלבד**. לכן, **נוכל להתמקד בשינוי הגבול בין הרובוט A למכשול, למשל B1**.

המחשה של סיבוב קודקוד בגבול ה-CB בין שתי שכבות עוקבות באוריינטציה



איור 29: שלוש זוויות עוקבות בסיבוב של הרובוט, להמחשת ההשפעה של שינוי מספר השכבות על התנועה של קודקודי שכבות ה-CB, וכך גם ההשפעה על תנועת גבולות ה-CB.

באיור 29 רואים שבין שתי שכבות עוקבות הרובוט מסתובב בזווית המעבר $\Delta\theta$. קודקודי ה-CB מגיעים ממגע קודקוד-קודקוד בין הרובוט למכשול. הקודקוד שמסומן בכתום נובע ממגע קודקוד-קודקוד ועבורו רדיוס הסיבוב הכי גדול, כך שממנו יתקבל הקודקוד בגבול ה-$\mathrm{C-Obstacle}$ שנע הכי הרבה בסיבוב $\Delta\theta$.

מכאן מתקבל שההתקדמות של קודקוד בין שתי שכבות עוקבות הוא של יתר במעגל. אפשר להשתמש באורך היתר, אבל מפני שבזוויות קטנות אורך היתר דומה לאורך הסגמנט המעגלי ל-$\Delta\theta$, וגם אורך הסגמנט יותר ארוך מהיתר, נשתמש באורך הסגמנט המעגלי.

לכן, קריטריון אלגברי למספר שכבות דרוש:

$$\Delta s = \text{distance travled by vertex of CB} = \Delta\theta R \leq \delta$$

כאשר $R = \text{Diagonal of Robot A} = \sqrt{8^2 + 1^2} = \sqrt{65}$ ו-δ מסמן רזולוציה בצירים x, y (גודל תאים ברשת הריבועים). בנוסף, נשים לב שקיים קשר בין מספר השכבות לסיבוב ביניהם:

$$\Delta\theta = \frac{2\pi}{N}$$

לכן:

$$\Delta\theta R = \frac{2\pi R}{N} \leq \delta \rightarrow N \geq \frac{2\pi R}{\delta} \rightarrow \boxed{N \geq \left\lceil \frac{2\pi R}{\delta} \right\rceil}$$

לכן, עבור רזולוצית x, y מסוימת, נקבל את הרזולוציה הדרושה בציר θ.

קריטריון זה מבטיח שכל גבול של CB לא נע יותר מתא אחד בין שכבות עוקבות של זווית.

סעיף 2 – בניית עץ צמתים וניווט עם אלגוריתם *A

אנחנו בוחרים לנווט בעץ שמבוסס על השיטה השנייה של הדיסקרטיזציה, בה בוחרים תאים חופשיים לפי יכולת הימצאות קודקוד הרובוט במרכז/פינת הפיקסל, עם רזולוציה מרחבית של $\delta = 1$ ומספר שכבות N שאפשר לכוונן לשיפור המסלול.

הסבר אלגוריתם *A ופונקצית המחיר איתה נעבוד

עבור מפה דיסקרטית שכוללת סימון לצמתים החופשיים, ניצור בקוד שתי משתנים ליצירת גרף הניווט:

1. vertss – שכולל את התאים החופשיים בכל שכבה, ממוספרים לפי סדר הופעתם במעבר על מטריצת הפיסקלים.
2. graph – שומר לכל תא ממוספר $\boldsymbol{x}_i$ את התאים השכנים שלו $\boldsymbol{x}_j$, שבמרחב החופשי בלבד ולא מחוץ לגבולות המרחב החופשי, עם מספור להם וחישוב המחיר ביניהם לתא הנוכחי $l_e(\boldsymbol{x}_i, \boldsymbol{x}_j)$

בעזרת עצמים אלו, בנינו גרף ניווט מתאים לשימוש ישיר באלגוריתם A^*.

מבני נתונים באלגוריתם:

1. O רשימת הצמתים הפתוחים
2. C רשימת הצמתים הסגורים
3. מצביע-אחורה, back-pointer, מכל צומת ב-O לצומת בגרף שגילתה אותה
4. מחיר $t(\boldsymbol{x})$ לכל צומת שהתגלתה

סימונים:

1. $l_e(\boldsymbol{x}_1, \boldsymbol{x}_2)$ – אורך המסלול בין $\boldsymbol{x}_1$ ל-$\boldsymbol{x}_2$, כאשר:

$$\boldsymbol{x}_1 = (x_1 \quad y_1 \quad \theta_1), \boldsymbol{x}_2 = (x_2 \quad y_2 \quad \theta_2)$$

$$l_e(\boldsymbol{x}_1, \boldsymbol{x}_2) = |x_1 - x_2| + |y_1 - y_2| + |\min(|\theta_1 - \theta_2|, 2\pi - |\theta_1 - \theta_2|)|$$

חיסור ערכים מוחלטים מהווה דרך לחשב מרחק ב-Grid של קוביות והשימוש במינימום בזווית נועד להתחשב במחזוריות של הסיבוב ולאפשר לרובוט להתייחס לסיבוב בכיוון השלילי (בכיוון השעון) באופן שקול לסיבוב בכיוון החיובי (נגד השעון).

2. $l_b(\boldsymbol{x})$ – אורך מסלול המצביעים לאחור מ-$\boldsymbol{x}$ חזרה ל-S

$l_b(\boldsymbol{x})$ יהיה מחושב ע"י סכימת $l_e(\boldsymbol{x}_1, \boldsymbol{x}_2)$-ים בין צמתים במסלול המצביעים לאחורה.

המחיר עבור A^*:

$$t(\boldsymbol{x}) = l_b(\boldsymbol{x}) + h(\boldsymbol{x}, \boldsymbol{T})$$

כאשר, נשתמש בנוסחה הבאה בין המטרה לצומת $\boldsymbol{x}$:

$$h(\boldsymbol{x}, \boldsymbol{T}) = l_e(\boldsymbol{x}, \boldsymbol{T})$$

<u>האלגוריתם:</u>

אתחל את הרשימות עם: $C = \{\phi\}, O = \{S\}$

x_{best} – הצומת הכי טובה לבחור באותו צעד.

בצעד ה-i:

1. בחר x_{best} מתוך O בתור צומת עבורה $t(\boldsymbol{x})$ מינימלי.
2. העבר את x_{best} ל-C , אם $x_{best} = T$ **עצור**.
3. עבור סביבה מקומית של x_{best}:

- זהה את כל צמתי השכנים שאינם ב-C.
- אם השכן חדש, הוסף אותו ל-O עם מצביע אחורה ל-x_{best}.
- אם השכן $\boldsymbol{x}_j$ כבר ב-O , תעדכן את צומת ההורה שלו (מצביע אחורה שלו) עם המסלול חזרה ל-S דרך x_{best}, אם מתקיים: $l_b(\boldsymbol{x}_{best}) + l_e(\boldsymbol{x}_{best}, \boldsymbol{x}_j) + h(\boldsymbol{x}_j, \boldsymbol{T}) < t(\boldsymbol{x}_j)$
- אם O ריקה, **תעצור**. אחרת, תמיין אותה ותמשיך.

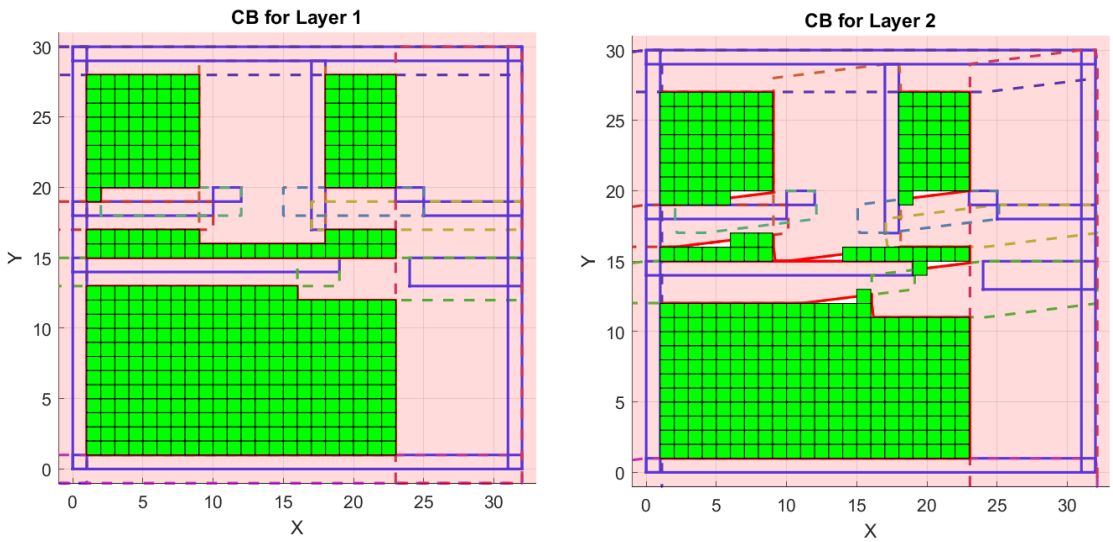
האלגוריתם עוצר כאשר T ברשימה הסגורה, כלומר $x_{best} = T$, **או** ש-O ריקה.

<u>הדרך שבחרנו ליצירת הצמתים לניווט</u>

עבור בחירת $\delta = 1$, לפי סעיף 1: $N \geq \left\lceil \frac{2\pi R}{\delta} \right\rceil = \lceil 50.657 \rceil \rightarrow N = 51$.

אמנם, בדיסקרטיזציה ה-2 (עם בדיקת חופשיות במרכז הפיקסל) אנחנו נתקלים בבעיה של תאים חופשיים עם שכנים מחוץ לגבול, **גם עם קיום הקריטריון שהוצא בסעיף 1**, כפי שניתן לראות בפקיסלים במרכז המפה.

דיסקרטיזציה לפי שיטה 2 (בדיקת מרכז הפיקסל) וה-CB הפיזיקלי עליו



איור 30: דוגמה לדיסקרטיזציה בין שתי שכבות ואתגר השכנים מחוץ לגבולות.

לכן, קיום הדירשה אינו פותר את האתגר עם שכנים בתוך ה-CB.

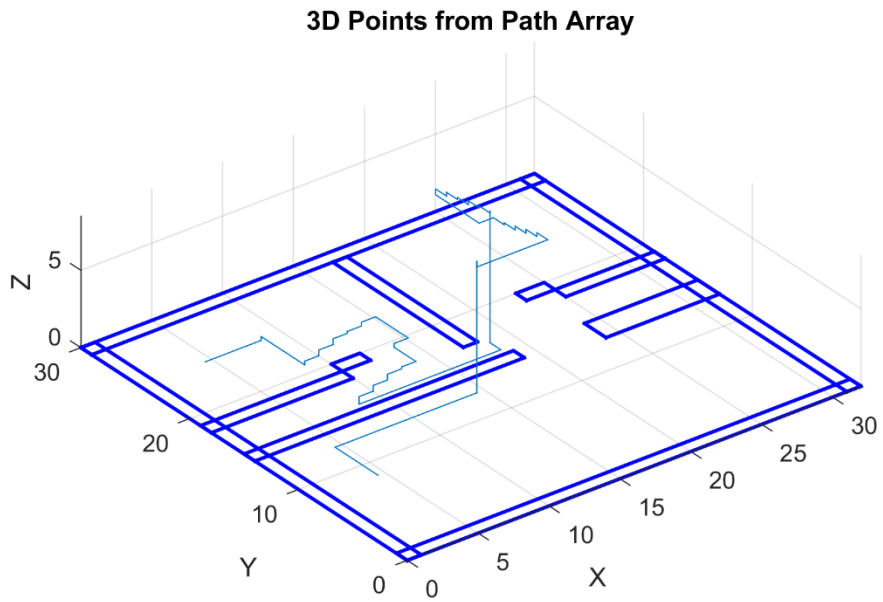
הפתרון שמצאנו: ביצירת העץ לניווט, בודקים אם כל תא הוא חופשי ואם הוא ממש בתוך "החורים" הפנימיים של ה-CB. כך אנחנו מבטיחים שהרובוט לא ייכנס לתוך המכשולים.

עבור שיטת דיסקרטיזציה 2, $\delta = 1, N = 70$ ומיקום הרובוט **במרכז** כל פיקסל

עבור בחירת $\delta = 1$, לפי סעיף 1: $N \geq \left\lceil \frac{2\pi R}{\delta} \right\rceil = \lceil 50.657 \rceil \rightarrow N = 51$.

אמנם, יצא שעבור $N = 51$ שכבות אין מסלול להגיע למטרה. לאחר מאט כיוונון, הגענו למספר שכבות $N = 70$. עכשיו הצלחנו לנווט למטרה! מספר הצמתים הכולל בגרף הניווט הינו 17,928, מספר הצמתים במסלול המתקבל מההתחלה למטרה הינו 117, מספר הצמתים שנסגרו הינו 5,786.

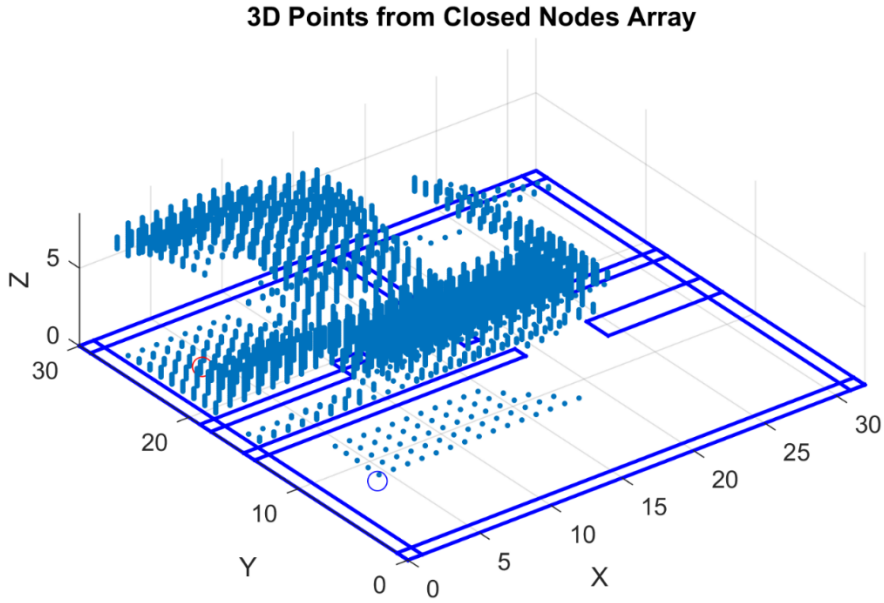
המסלול מההתחלה לסוף בתלת מימד, עם צירי מיקום במישור וזווית בגובה, עבור שיטת ניווט 1



איור 31: המסלול מההתחלה עד הסוף, בתלת מימד בקואורדינטות (x, y, θ), עבור הניווט הראשון.

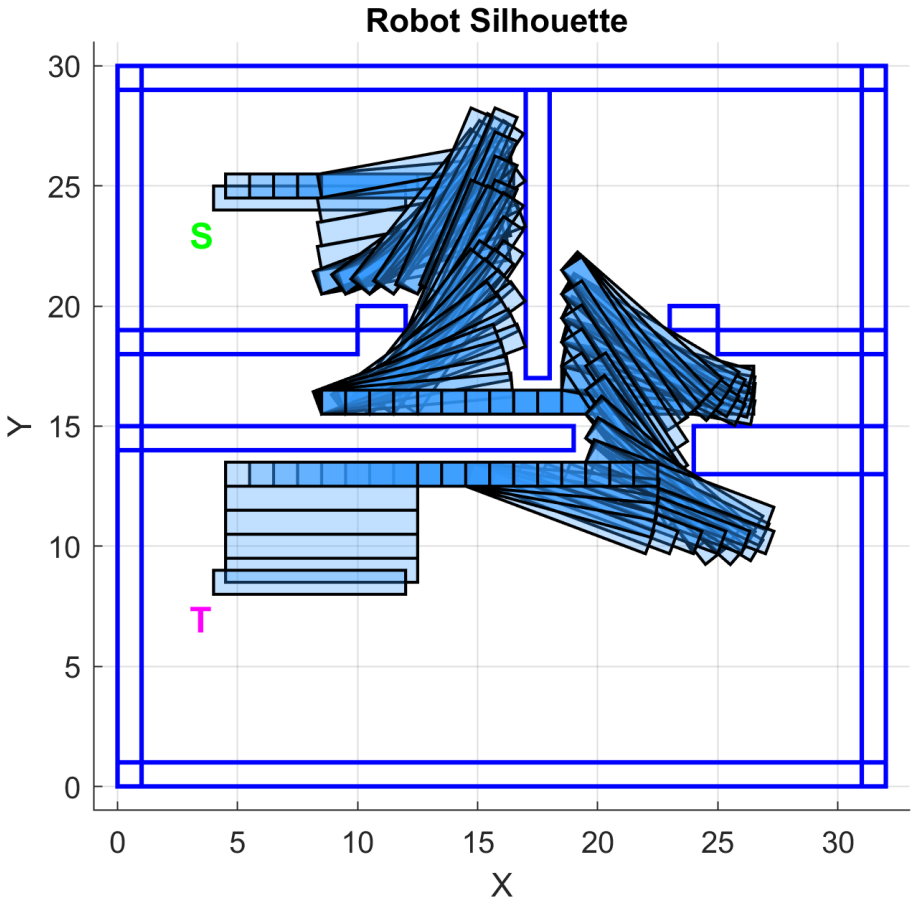
כאשר כל קפיצה אנכית היא סיבוב בין הזוויות $\theta = 0°, 360° - \frac{360°}{N} = 354.86°$.

הצמתים שנסגרו בתלת מימד, עם צירי מיקום במישור וזווית בגובה, עבור שיטת ניווט 1



איור 32: הצמתים שנסגרו, בתלת מימד בקואורדינטות (x, y, θ), עבור הניווט הראשון.

Silhouette של המסלול שבוצע עבור שיטת ניווט 1



איור 33: צללית של המסלול שבוצע, בדו-מימד, עבור הניווט הראשון.

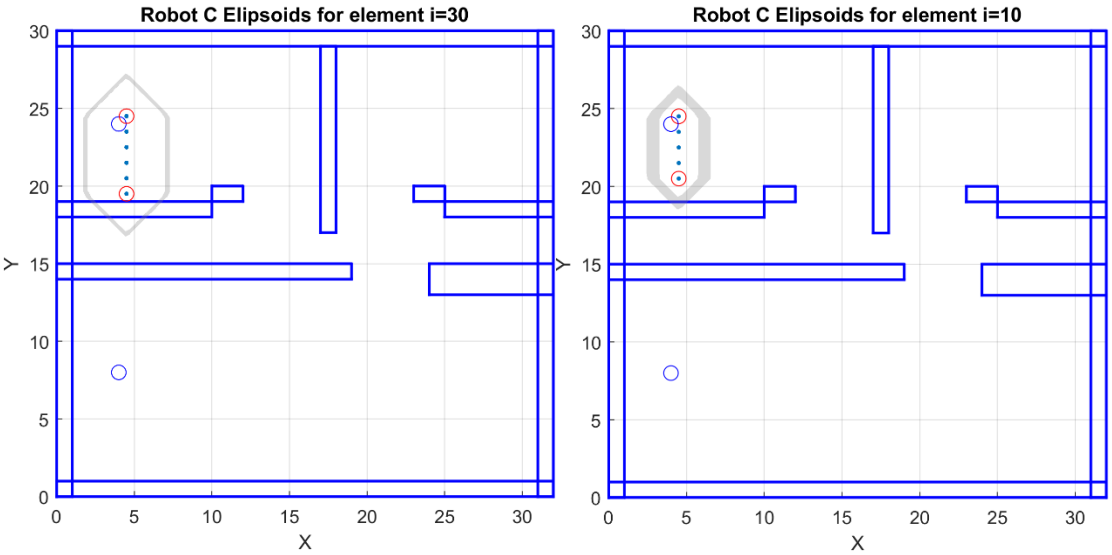
הצמתים פה נוצרו כך שיש חיבור בין צמתים בשכבה 1 לשכבה N. המחיר למסלול הינו 79.1. נסתכל כעת על האליפסה המתקבלת לכמה צמתים שנסגרו. האליפסה שמתקבלת, עבור המחיר נורמה 1 שקבענו, היא:

$$\|\boldsymbol{x}_{best} - \boldsymbol{S}\|_1 + \|\boldsymbol{x}_{best} - \boldsymbol{x}_{\text{closed}}\|_1 \leq l_{\min}$$

כאשר $l_{\min}$ זה מרחק מינימלי לאורך הגרף בין ההתחלה לצומת שנסגרה. היא מתארת את חסימת כל הצמתים שנסגרו, אחד אחרי השני.

נשים לב שבגלל השימוש בנורמה 1 בגרף של grid ריבועי, מתקבל שאליפסה נורמה 1 חוסמת את הצמתים שנסגרו ולא אליפסה נורמה 2 חלקה. עבור אותו $l_{\min}$, **אליפסה מבוססת נורמה 2 תחסום את האליפסה של נורמה 1 ולכן אפשר לצייר גם אותה כחסם לצמתים שנסגרו**.

האליפסה נורמה 1 החוסמת לצומת ה-10 וה-30 שנסגרו

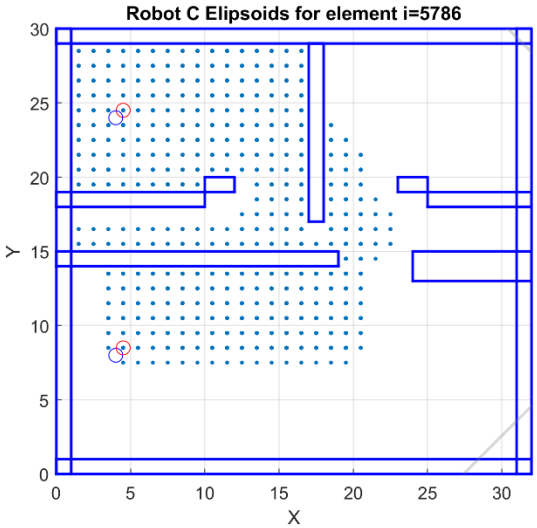


איור 34: האליפסה נורמה 1 שחוסמת את הצמתים שנסגרו עבור הצומת ה-10 וה-30 בהתאמה.

כאשר באדום זה מרכזי האליפסה ובכחול זה נקודות ההתחלה והסיום (אינם במרכז הפיקסלים). **ההיקף החיצוני של הצורה ההאפורה זה האליפסה החוסמת**, יש לה עובי כי היא מצויירת בתלת-מימד וזה מבט על.

המרחקים לשתי המקרים הנ"ל הם $l_{\min} = 4.1232,5.616$.

האליפסה נורמה 1 החוסמת למטרה שנסגרה



איור 35: האליפסה נורמה 1 שחוסמת את המטרה שנסגרה.

עבור אליפסה זאת, $l_{\min} = 79.1$. רואים שהאליפסה יוצאת מהחדר, אבל חוסמת את כל הצמתים שנסגרו (צלע אחת שלה מימין-למטה).

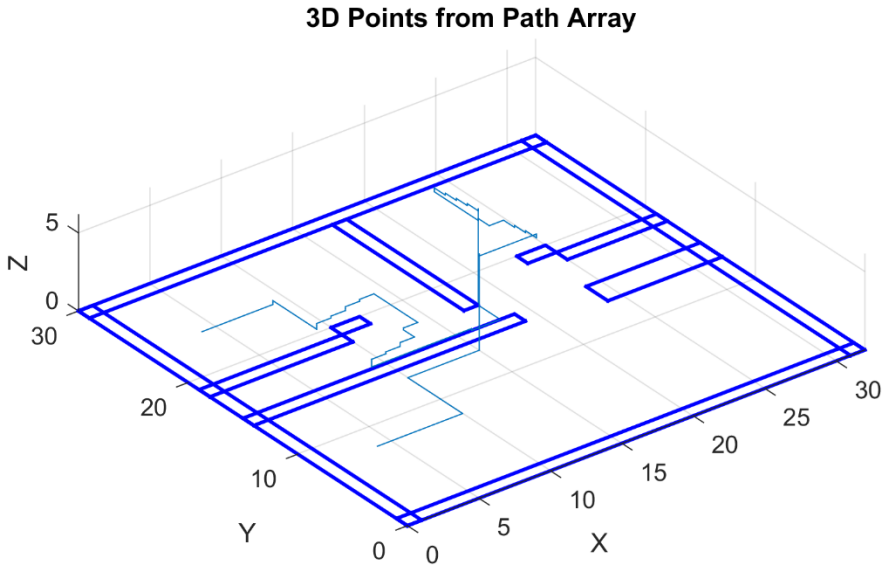
בנוסף, אם רוצים לראות את חסימת הצמתים הסגורים בתלת-מימד, יש צורך בהתחשבות במחזוריות של הזווית בהדפסת האליפסה החוסמת, אבל לבדיקת חסימת הקואורדינטות x-y הדפסה זאת מספיקה.

עבור שיטת דיסקרטיזציה 2, $\delta = 1, N = 51$ ומיקום הרובוט **בפינת** כל פיקסל

מכיוון שפה אנחנו בוחרים תאים בדיסקרטיזציה לפי חופשיות פינת הפיסקל, יהיה יותר קל לנווט למטרה, למשל באיזורים כמו המרווח של שני הבלוקים בין B2,B4.

מספר הצמתים הכולל בגרף הניווט הינו 14,188, מספר הצמתים במסלול המתקבל מההתחלה למטרה הינו 99, מספר הצמתים שנסגרו הינו 3,382.

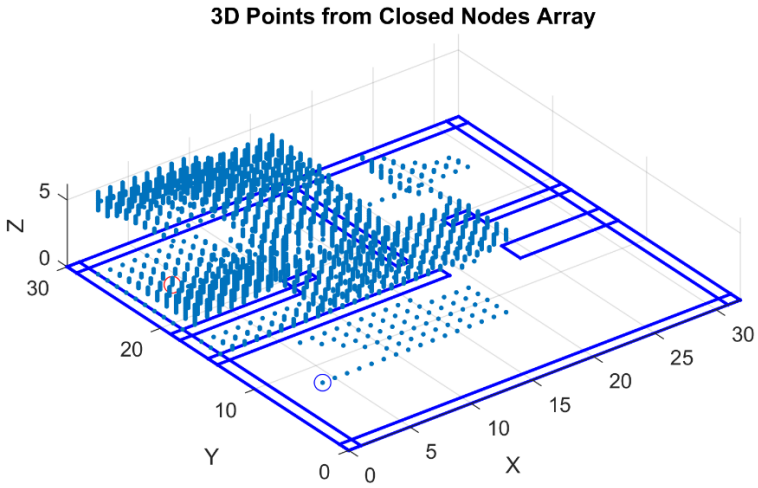
המסלול מההתחלה לסוף בתלת מימד, עם צירי מיקום במישור וזווית בגובה, עבור שיטת ניווט 2



איור 36: המסלול מההתחלה עד הסוף, בתלת מימד בקואורדינטות (x, y, θ), עבור הניווט השני.

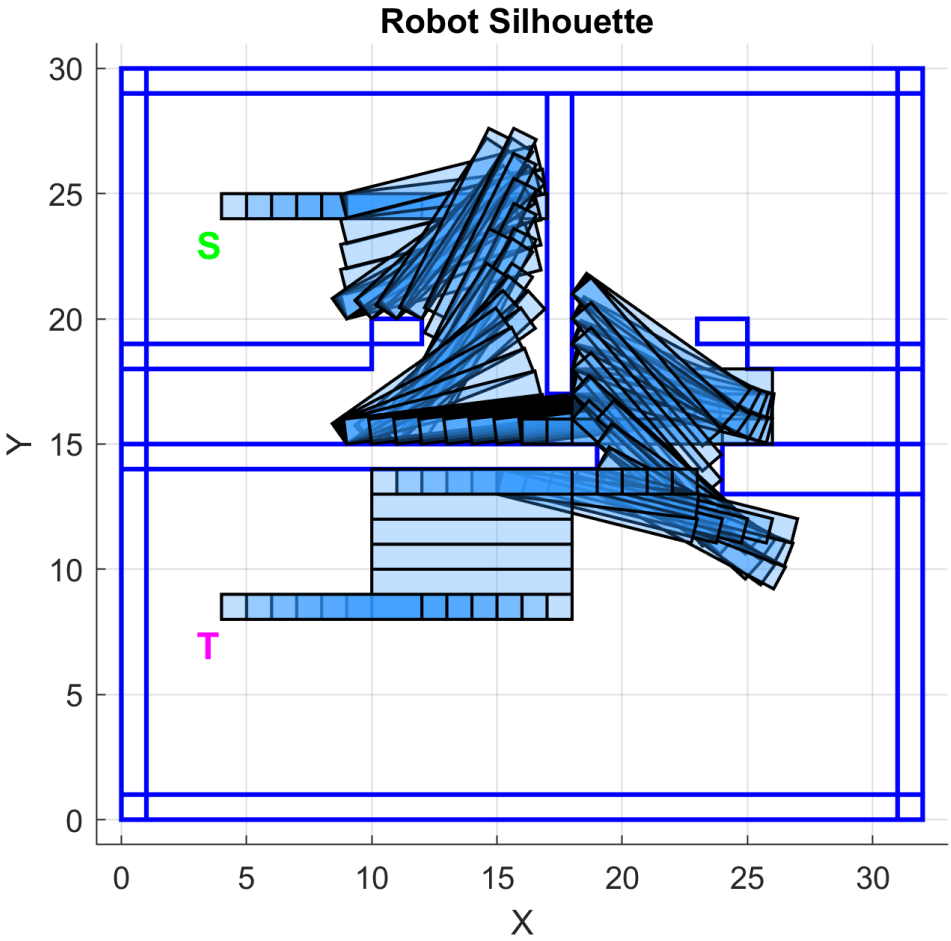
כאשר כל קפיצה אנכית היא סיבוב בין הזוויות $\theta = 0°, 360° - \frac{360°}{N} = 352.94°$.

הצמתים שנסגרו בתלת מימד, עם צירי מיקום במישור וזווית בגובה, עבור שיטת ניווט 2



איור 37: הצמתים שנסגרו, בתלת מימד בקואורדינטות (x, y, θ), עבור הניווט השני.

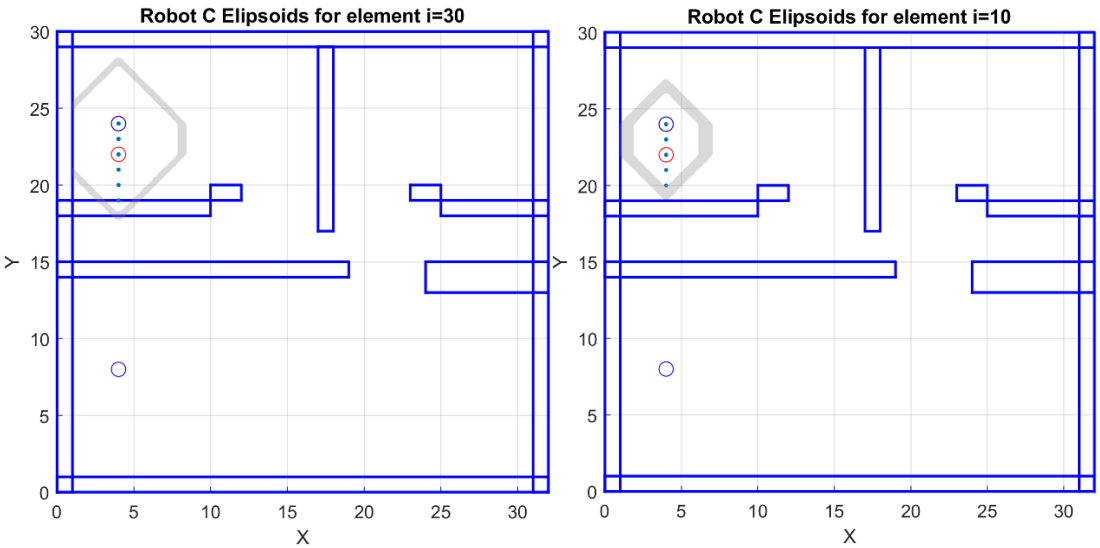
Silhouette של המסלול שבוצע עבור שיטת ניווט 2



איור 38: צללית של המסלול שבוצע, בדו-מימד, עבור הניווט השני.

האליפסה לצומת ה-10 וה-30 שנסגרו עם המרחקים $l_{\min} = 3.1232, 3.3696$:

האליפסה נורמה 1 החוסמת לצומת ה-10 וה-30 שנסגרו

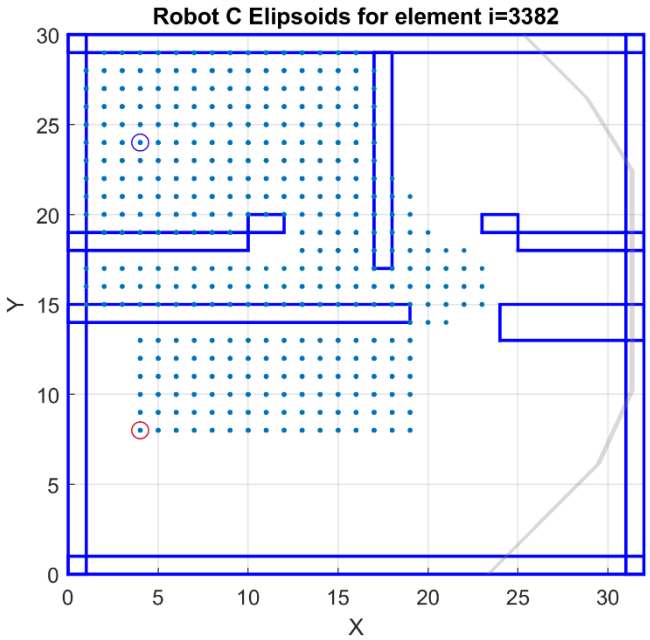


איור 39: האליפסה נורמה 1 שחוסמת את את הצמתים שנסגרו עבור הצומת ה-10 וה-30 בהתאמה.

כאשר באדום זה מרכזי האליפסה ובכחול זה נקודות ההתחלה והסיום (אינם בפינת הפיקסלים).

פה נקודת ה-Start במטלה היא תמיד אחת ממרכזי האליפסה. האליפסה גדלה בין הצומת ה-10 וה-30 כי הצומת ה-30 עם הפרש יותר גדול בזווית ביחס להתחלה מאשר הצומת ה-10.

האליפסה נורמה 1 החוסמת למטרה שנסגרה



איור 40: האליפסה נורמה 1 שחוסמת את המטרה שנסגרה.

עבור אליפסה זאת, $l_{\min} = 71$. רואים שהאליפסה יוצאת מהחדר וחוסמת את כל הצמתים שנסגרו.

סעיף 3 – ניווט עם אלגוריתם A* במצב online

בפרק זה נתמקד בבעית הניווט של רובוט דיסק בקוטר $D = 1$ בחדר הנתון בבעיה ועם האילוצים הבאים לניווט במצב online:

1. הרובוט יודע את מיקומו בכל רגע $(x(t), y(t))$
2. יודע את מיקום המטרה $(x_T . y_T)$
3. עם חיישני מכשולים, יודע את ה-free-space בתאים הסמוכים במפה
4. **אינו יודע את המפה של הדירה מראש!**

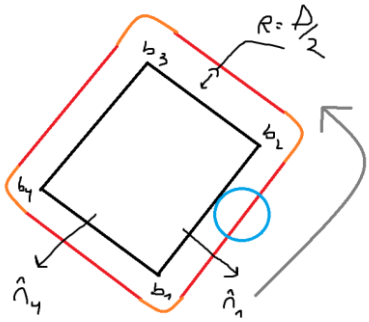
אנחנו מתמקדים בתכנון אלגוריתם ניווט מבוסס A* שיועד להתחשב באילוצים אלו. ניווט ע"י A* ב-offline יתן את המסלול האופטימלי במפה, כך שהפעלת האלגוריתם מסעיף 2 תהווה השוואה טובה לכמה המסלול ב-online "יותר גרוע". לסעיף ה-online, נגדיר גם:

$$(x_S . y_S) = (4.5,24.5),\ \ (x_T . y_T) = (4.5,8.5).$$

בניית המפה הדיסקרטית לרובוט דיסק

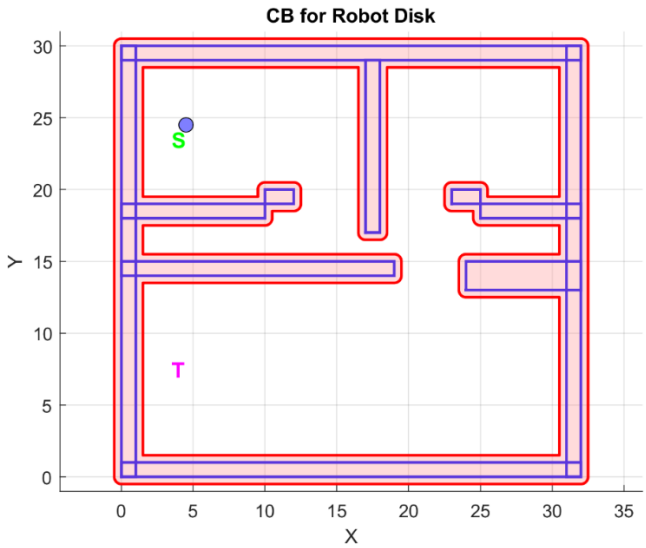
קודם ניצור את ה-CB הגיאומטרי בין הרובוט הדיסק למכשולים.

ציור ה-CB בין דוגמת פוליגון קמור לרובוט דיסק בדו-מימד



איור 41: תכונות ה-CB בין רובוט דיסק למכשול פוליגון קמור.

ה-CB בין רובוט הדיסק בקוטר 1 לדירה

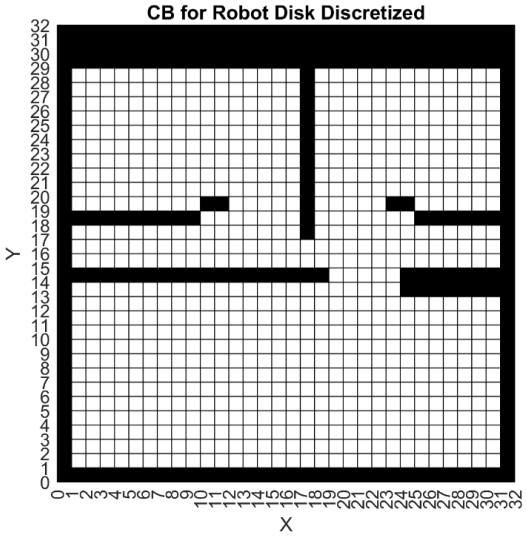


איור 42: ה-CB הגיאומטרי בין הרובוט דיסק לכל המכשול בדירה.

בדומה ליצירת ה-CB בין הרובוט המלבני לחדר, ניצור את ה-CB בין הדיסק למכשול קמור, ואז נבצע איחוד לקבלת ה-CB הכולל. כפי שניתן לראות באיור 41, ה-C-Obstacle בין הדיסק למכשול קמור מורכב מצלעות שמוזזות באורך $\text{displacement} = R = \frac{D}{2}$ וביניהם סגמנטי מעגלים ברדיוס הדיסק.

לאחר חישוב ה-C-Obstacle בין הדיסק לכל אלמנט קמור בדירה, מתקבל ה-CB בין הרובוט הדיסק לדירה כולה באיור 42. עכשיו שיש לנו מרחב חופשי, נבצע לו דיסקרטיזציה לפי היכולת של מרכז הדיסק להימצא במרכז כל פיקסל.

דיסקרטיזציה של ה-CB בין דיסק קוטר 1 לדירה, עבור הימצאות הרובוט במרכז כל פיקסל

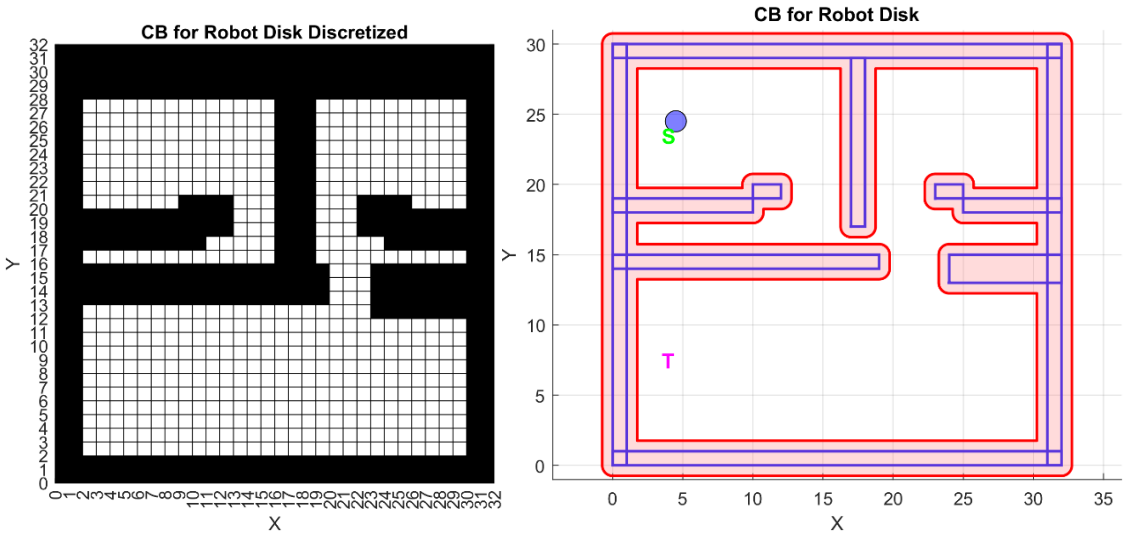


איור 43: ה-CB לאחר דיסקרטיזציה בין הרובוט דיסק לכל המכשול בדירה, לפי הימצאות הרובוט במרכז כל פיקסל.

נבצע ניווט במפה מאיור 43. יש התאמה לציפייה שלנו שעבור דיסק 2D עם רדיוס חצי.

בנוסף, עבור דיסק בקוטר 1.5, **<u>לא בשביל הניווט אחרי זה</u>, בשביל להראות אפשרויות**:

CB בין דיסק קוטר 1.5 לדירה, גיאומטרי ודיסקרטיזציה, עבור הימצאות הרובוט <u>במרכז</u> כל פיקסל



איור 44: ה-CB לאחר דיסקרטיזציה בין הרובוט דיסק לכל המכשול בדירה, לפי הימצאות הרובוט במרכז כל פיקסל.

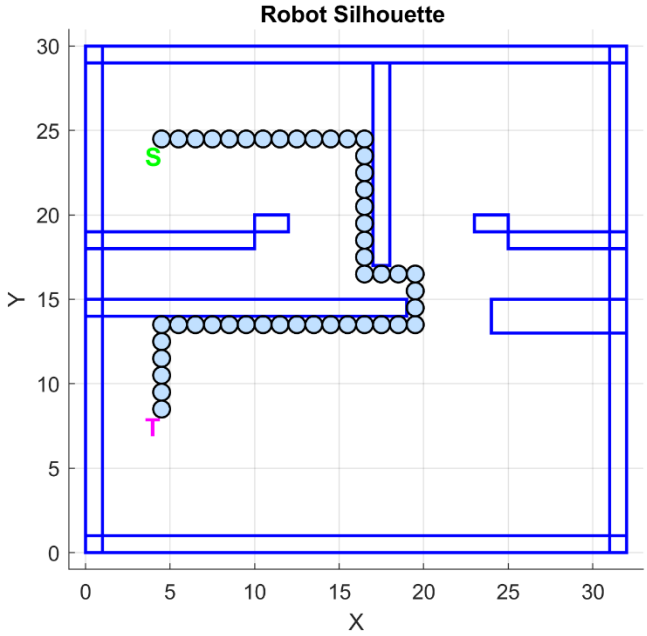
עכשיו נלך לבצע ניווט offline.

ניווט במפה עם A* במצב offline

אחרי הרצת A*, עם אותו $l_e(x_1, x_2)$ מהסעיף הקודם, המסלול האופטימלי המתקבל מופיע באיור 45.

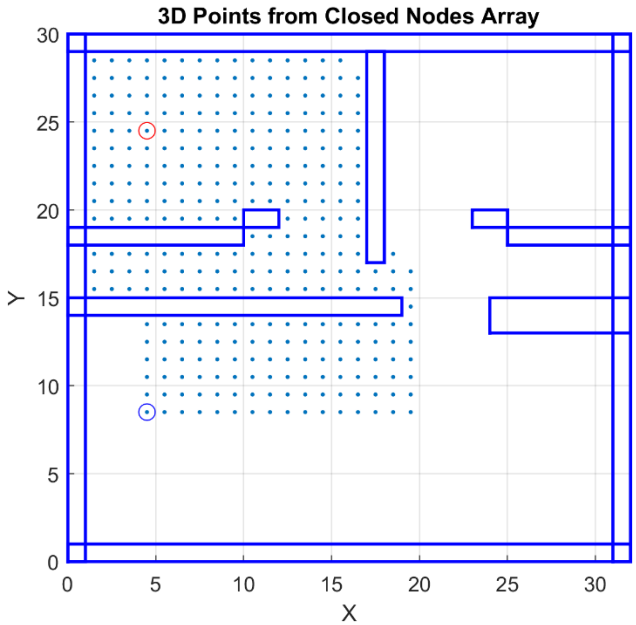
קיבלנו 777 צמתים בגרף, במסלול האופטימלי 47 צמתים (כולל ההתחלה והסוף) ונסגרו 316 צמתים (41% מהצמתים). הצמתים שנסגרו מופיעים באיור 46.

המסלול המתקבל מהרצת A* על המפה מאיור 43



איור 45: המסלול האופטימלי לפי A* של הרובוט הדיסק במפה של איור 43 מההתחלה למטרה.

הצמתים שנסגרו בהפעלת A* על המפה באיור 43



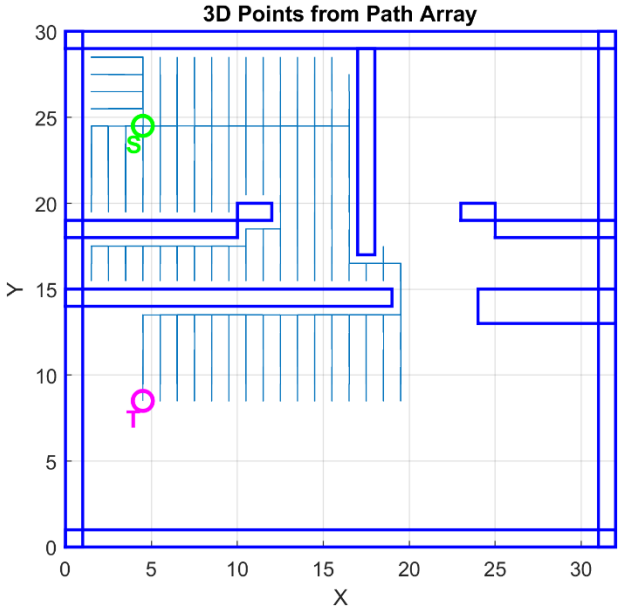
איור 46: הצמתים שנסגרו בהפעלת A* על המפה באיור 43 מההתחלה למטרה.

ניווט במפה עם A* במצב online שיטה 1

השיטה הראשונה שנציע לשינוי A* לעבוד ב-online היא: בניווט נשתמש ב-A* אבל ברגע שרוצים לעבור ל-x_{best} הבא מצומת x_j הנוכחית נבחר לחזור עם המצביעים לאחור של x_j ל-S ואז ללכת עם המצביעים לאחור של x_{best} בכיוון ההפוך עד ההגעה ל-x_{best} החדש.

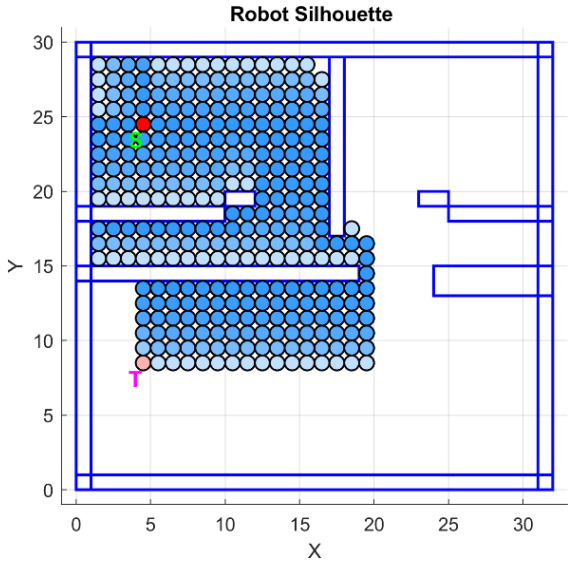
השיטה הזאת משתמשת ב-A* כמו שהוא, ורק מוספיה התחשבות במסלול האמיתי שיש לעשות. עם הפשטות הזאת בא המחיר שזה שיטת ניווט מאוד לא פרקטית, כפי שרואים באיורים 47,48.

המסלול המתקבל מהרצת A* אונליין שיטה 1 על המפה מאיור 43



איור 47: המסלול האמיתי שמבצע הרובוט לפי A* אונליין שיטה ראשונה.

הצללית של המסלול מהרצת A* אונליין שיטה 1 על המפה מאיור 43



איור 48: הצללית של המסלול האמיתי שמבצע הרובוט לפי A* אונליין שיטה ראשונה.

מספר הצמתים במסלול האמיתי הוא 11,849! פי 250 יותר צמתים מאשר במסלול האופטימלי!

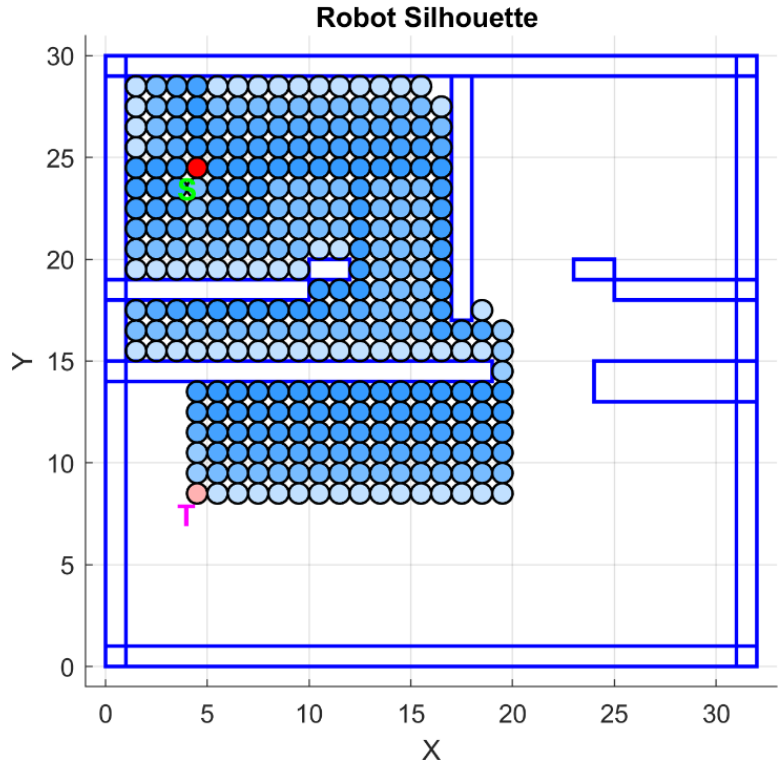
ניווט במפה עם A* במצב online שיטה 2

השיטה השנייה היא שינוי קל לקודמת: שיש חיתוך בין מסלול המצביעים לאחור חזרה ל-S למצביעים קדימה ל-x_{best} החדש, מקצרים את הדרך ע"י המשך במצביעים קדימה ל-x_{best}.

הצמתים שנסגרו הם בדיוק אותם צמתים כמו קודם, ולפי איור 49 הצללית זהה לקודם.

אבל, מספר הצמתים במסלול האמיתי הוא 2,008. פי 43 יותר צמתים מאשר במסלול האופטימלי במקום פי 250, **ירידה של פי 5 בכמות הצמתים של המסלול האמיתי!**

הצללית של המסלול מהרצת A* אונליין שיטה 2 על המפה מאיור 43



איור 49: הצללית של המסלול האמיתי שמבצע הרובוט לפי A* אונליין שיטה שנייה.

סיכום ניווט אונליין

נסדר בטבלה קצרה את ההשוואה בין ניווט offline וניווט אונליין:

שיטת הניווט	מספר צמתים במסלול
A* רגיל, offline	47
A* בשיטת אונליין 1	11,849
A* בשיטת אונליין 2	2,008