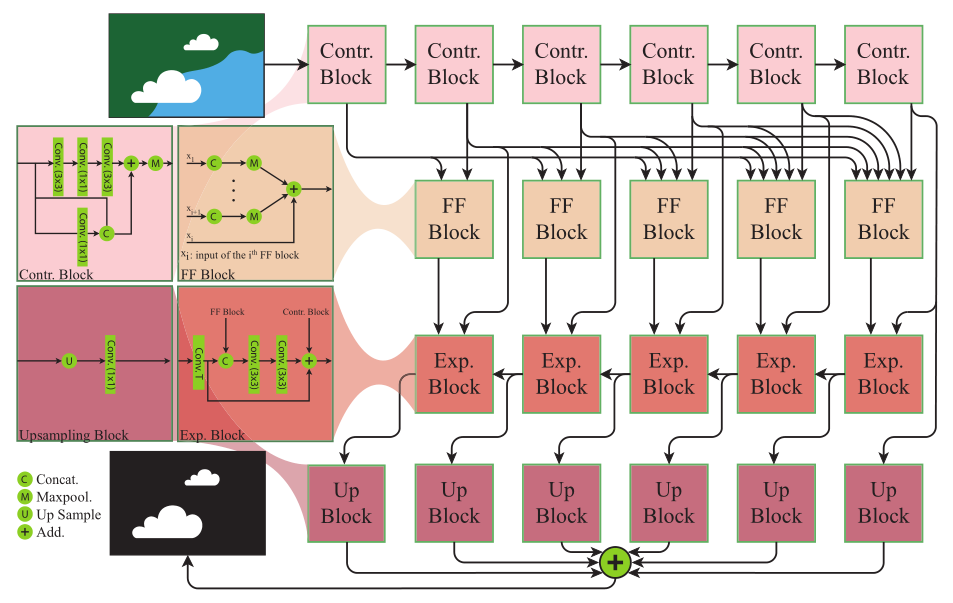
מטרות פרויקט:

* יצירת מערכת לביצוע סגמנטציה סמנטית של עננים על ידי אימון רשת קונבולוציה בעזרת מפת תיוג מלאה.
* אימון הרשת על ידי מפות תיוג חלקיות (Weakly supervised) והשוואת ביצועים למול תיוג מלא.

סקר ספרות:

1. Cloud-Net+: A Cloud Segmentation CNN for Landsat 8 Remote Sensing Imagery Optimized with Filtered Jaccard Loss Function

מאמר זה מציע רשת קונבולוציה מסוג Dense net, ו-Loss Function אשר מתאימות לביצוע משימת הסגמנטציה במקרה של עננים.  
מבנה הרשת מתואר בתמונה הבאה:



ניתן לראות כי הרשת מבצעת כיווץ של התמונה המקורית (Contraction blocks), איסוף כל התוצאות (Feedforward blocks), הרחבת התוצאות חזרה לגודל מקורי (Expansion and Upsampling blocks)

וסכימת התוצאות לקבלת מסכת הסגמנטציה.  
  
שימוש ב-Filtered Jaccard Loss Function:

כאשר t מייצג את תיוג האמת, N מציין את מספר הפיקסלים הכולל, ו-y מייצג את פלט הרשת שלנו.

בנוסף, מייצג את soft Jaccard loss אשר מתוארת בצורה הבאה:

כאשר הינו מספר קטן מאד על מנת להימנע מחלוקה ב-0.

*במקרה בו, כל הפיקסלים עם תיוג 0 (אין עננים בתמונה) נשתמש ב שהוא*

*ושווה ל:*

*כאשר .*

*יש צורך להוסיף הסברים איך מאמר זה מתמודד עם הבעיות בסגמנטציה של עננים*

*ברשת קונבולוציה עמוקה, שכבות עמוקות יותר מזהות דפוסים מופשטים ומורכבים יותר בקלט.*

*לכן, פרטים פשוטים כמו צבעים, קצוות ופינות מזוהים רק על ידי השכבות הראשונות של הרשת.*

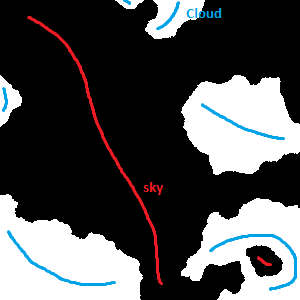
*בניגוד לבעיות אחרות כמו סיווג של תמונות, בסגמנטציה סמנטית צריך לשמור על הפרטים הפשוטים האלה, משום שנדרש זיהוי מדויק של הגבולות של עצמים, ויצירת פלט שדומה במבנה אל הקלט.*

*לשם כך, הרשת שבה השתמשו במאמר (וגם אנחנו השתמשנו בה בפרויקט שלנו), מעבירה את הפלטים של כל השכבות באופן כמעט ישיר אל השכבה שמייצרת את הפלט (על ידי המעבר*

*Contr->FF->Exp->Up), וכך המאפיינים הפשוטים של התמונה עוברים כפי שהם (או על ידי עיבוד מסוים שמזקק מהם את המידע הרלוונטי) מהשכבה שבהם הם נוצרים, אל השכבה שמייצרת את הפלט. זאת בניגוד לארכיטקטורה סדרתית כמו Encoder-Decoder בלי skip connections שבה הפלט של כל שכבה מגיע לסוף רק דרך השכבות העוקבות, אשר מפשטות אותו, כך שלבסוף הפלט של השכבות הראשונות משפיע בצורה הרבה יותר מופשטת על הפלט.*

1. *הוספת סקירת מאמרים בתחום של weakly ו – scribbles*

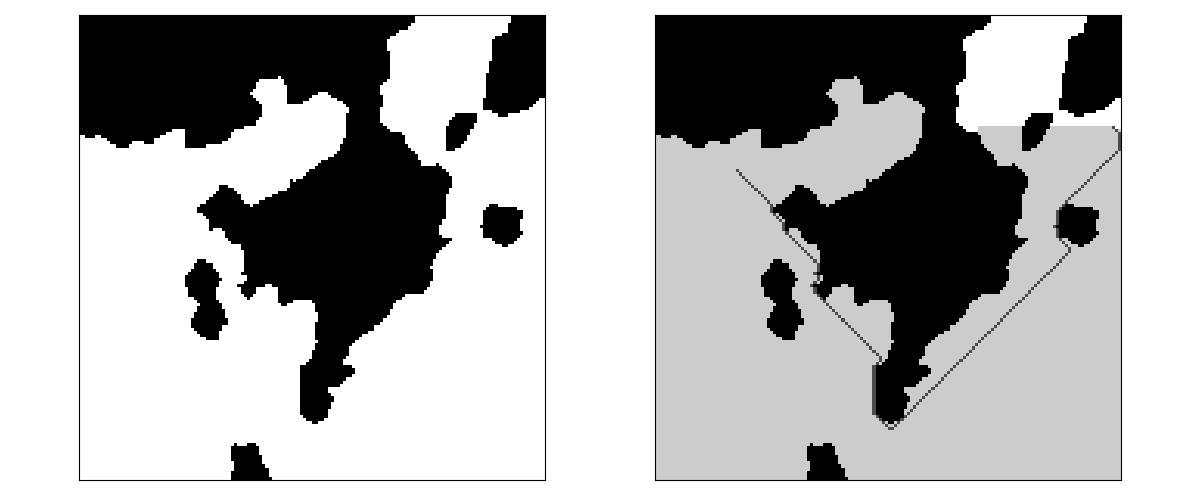
***יצירת שרבוטים עבור אימון weakly supervised מתוך מפות סגמנטציה***

*בחלק זה ניסינו כמה אלגוריתמים שונים על מנת ליצור שרבוטים מספיק טובים, אשר מתארים בצורה נכונה את מפות הסגמנטציה.  
דוגמא שנוצרה ידנית עבור מפת סגמנטציה יחידה:  
*

*ניתן לראות בצבע כחול את השרבוטים עבור כל אחד מהעננים בתמונה, ובצבע אדום את השרבוטים עבור השמיים.  
נחפש אלגוריתם אפשר יאפשר לנו לייצר שרבוטים כאלה בצורה אוטומטית, כעת נציג כמה מהאלגוריתמים שניסינו ותוצאותיהם:*

* *שימוש באלגוריתמי חיפוש:  
  הגרלת שתי נקודות בתוך ענן במפת הסגמנטציה וחיבורן על ידי אלגוריתם חיפוש בשימוש היוריסטיקה של מרחק מנהטן כאשר ניתן לעבור רק בפיקסלים לבנים.*
* ***UniformCost****:*

*תוצאות:*

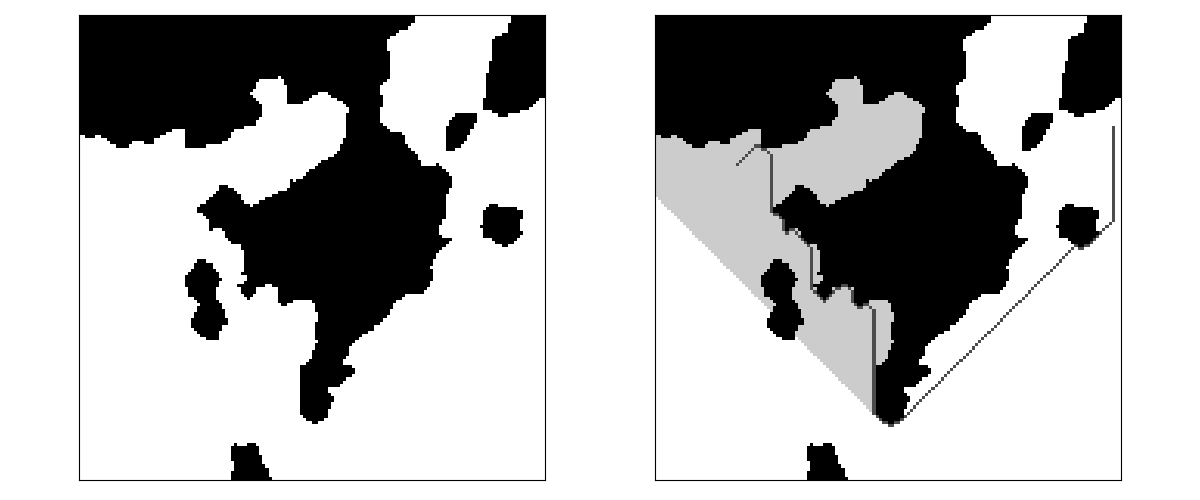
*התמונה מימין מייצגת את התוצאה הסופית, כאשר הפיקסלים בצבע אפור בהיר מסמלים את המצבים בהם עברנו בזמן החיפוש, והפיקסלים בצבע אפור כהה מסמלים את המסלול הסופי שנמצא.*

*בעיות בפתרון:*

1. *זמן החישוב ארוך מאד בגלל פיתוח המון מצבים. (כמה דקות לתמונה)*
2. *המסלול נוטה להיצמד לפינות של העננים – נובע מכך שיש ניסיון למצוא מסלול קצר ביותר.*
3. *לפעמים הנקודות שנבחרות קרובות מידי או בשני עננים שונים ואז מתקבלת תוצאה לא טובה.*

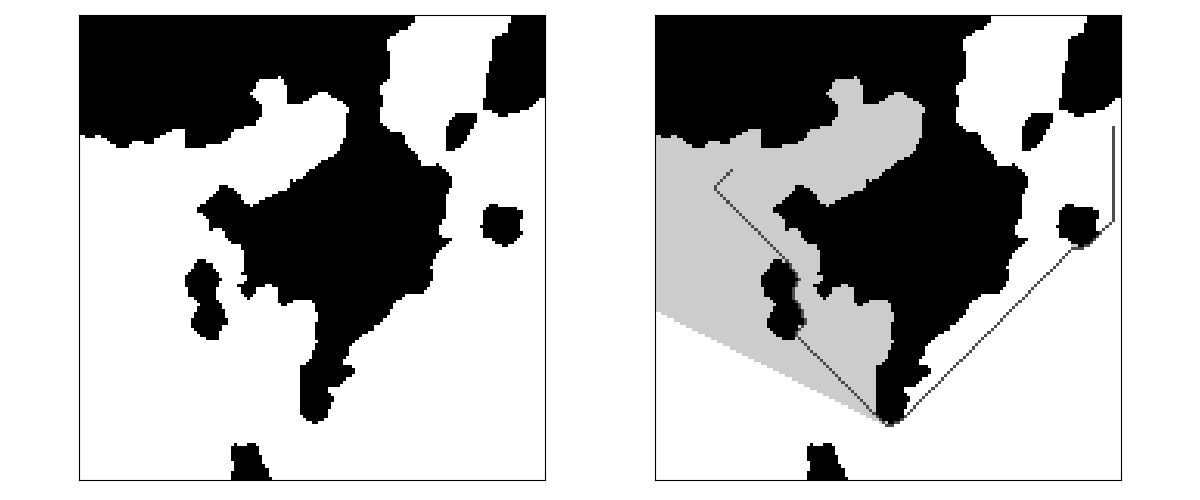
* ***GreedyBest:***

*תוצאות:*

*על מנת לחסוך בכמות המצבים שפותחו ניסינו את אלגוריתם GreedyBest אשר מפתח פחות מצבים ומוותר על מציאת המסלול הקצר ביותר. (עבור הבעיה שלנו אין צורך במסלול קצר ביותר)*

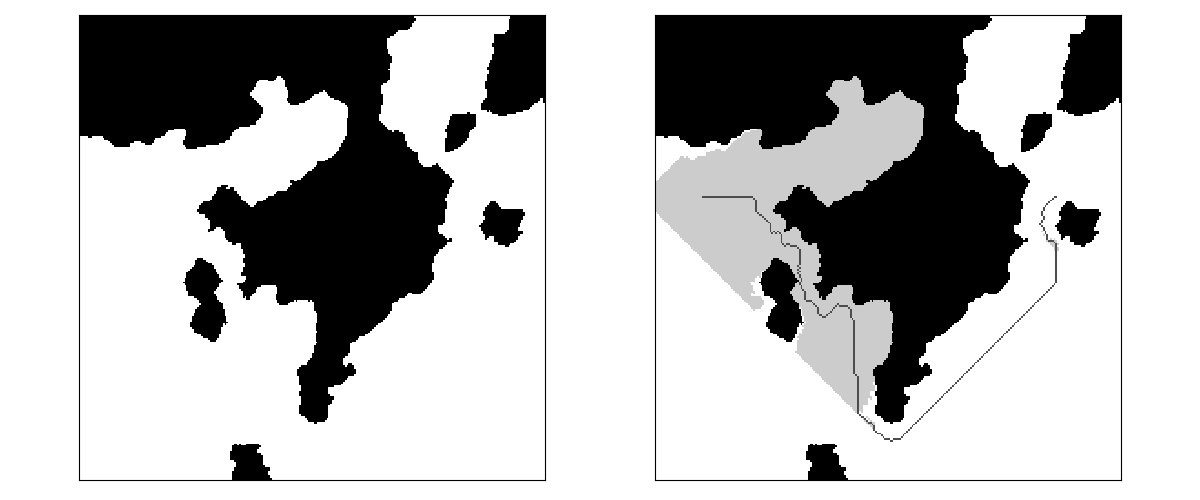
*בעיות בפתרון:*

1. *זמן החישוב עדיין ארוך. (לפחות 15 שניות לתמונה)*
2. *המסלול נוטה להיצמד לפינות של העננים.*
3. *לפעמים הנקודות שנבחרות קרובות מידי או בשני עננים שונים ואז מתקבלת תוצאה לא טובה.*

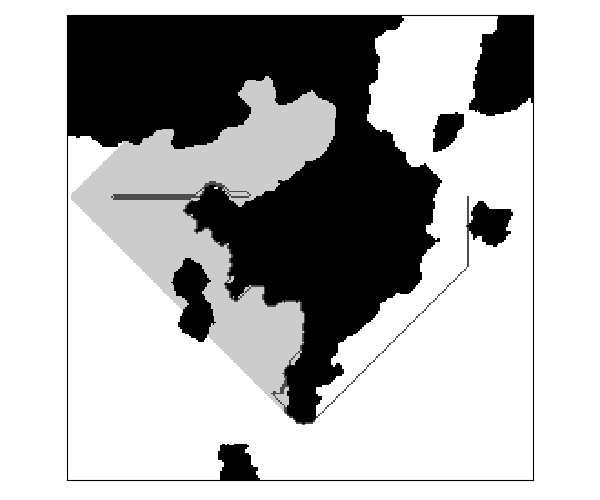
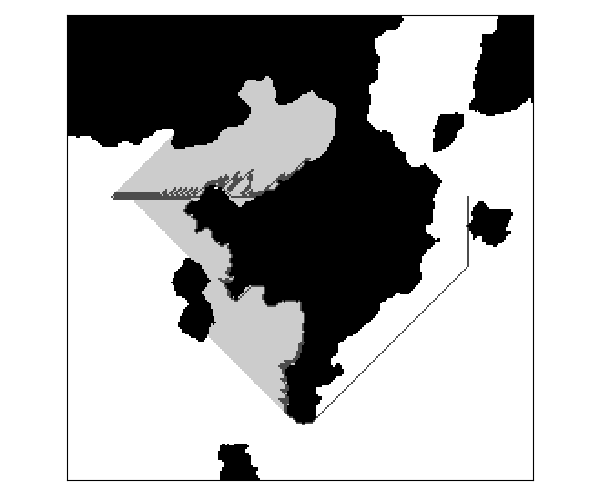
* ***וריאציה של Astar****:  
  תוצאות:*

*על מנת לקבל תמונה מלאה החלטנו לנסות אלגוריתם בסגנון Astar.*

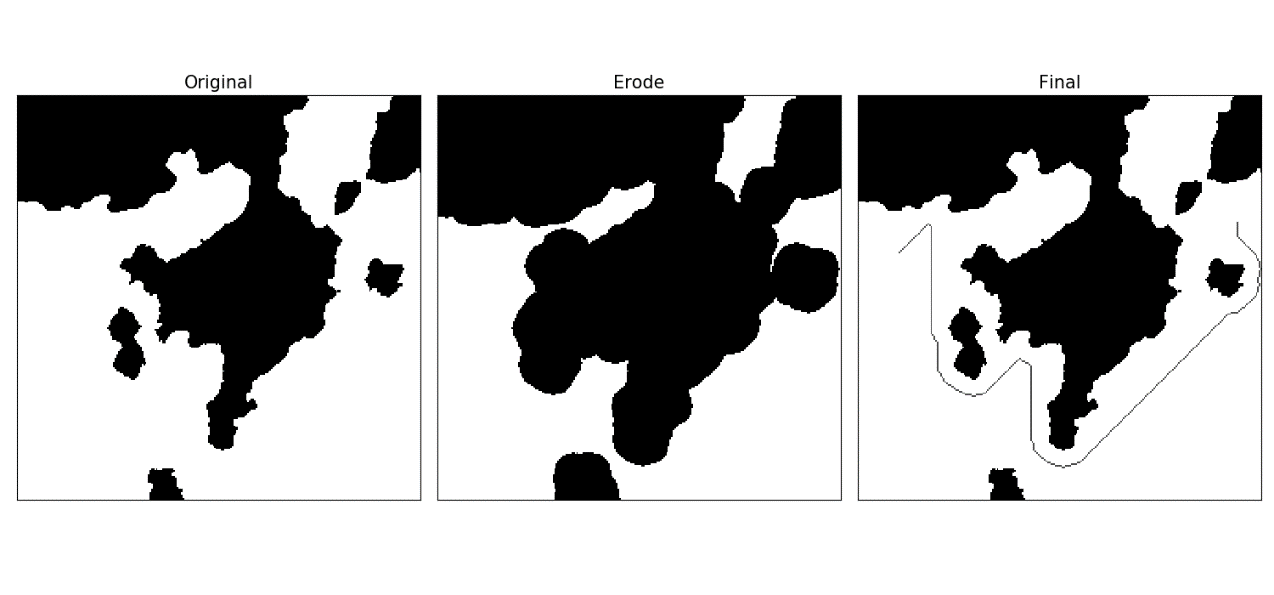
*פתרון זה סובל מבעיות דומות לפתרונות הקודמים.*

*על מנת לנסות להתמודד עם הבעיות שנתקלנו בהן החלטנו כי כעת נמשיך עם אלגוריתם GreedyBest בגלל שהינו המהיר מבין האלגוריתמים.  
תחילה ננסה להתמודד עם הבעיה של הצמדות לפינות על ידי שינוי הפונקציה היוריסטית שלנו, אשר "תעניש" בחירת מסלול הקרוב לפינה.  
תוצאות:  
  
ניתן לראות כי אכן הצלחנו להתרחק מרוב הפינות, אך שילמנו על כך בזמן חישוב היוריסטיקה עבור כל מצב.  
בעיות בפתרון:*

1. *זמן החישוב גדל בגלל חישוב היורסטיקה. (30 שניות לתמונה)*
2. *השרבוט עוקב מידי אחר צורת הפינות בתמונה.*

*על מנת להתמודד עם זמן החישוב, ניסינו אלגוריתמי חיפוש לוקאלים, אשר מפתחים פחות מצבים ולכן יותר מהירים.****BeamSearch****:   
*

*מימין – הגבלת מספר המצבים הפתוחים ל- 20, משמאל – הגבלת מספר המצבים הפתוחים ל- 100.  
בשני המקרים קבלנו תוצאות לא טובות, השרבוט שנוצר חוזר על עצמו ולא נראה טוב (שימוש בהיוריסטיקה מרחק מנהטן).  
שימוש בהיוריסטיקה המענישה קרבה לפינות:  
  
גם במקרה זה קיבלנו תוצאות לא טובות.*

*כעת, לאחר ניסיונות שונים עם אלגוריתמי חיפוש, החלטנו לפנות לכיוון שונה הכולל בתוכו פעולות מורפולוגיות על התמונה.  
כאשר תחילה חשבנו על ביצוע פעולת שחיקה (****erode****) על התמונה ורק לאחר מכן ביצוע חיפוש – כך נוכל להימנע מהצמדות לפינות.  
תוצאות עבור שימוש באלמנט בניה בצורת דיסק עם רדיוס של 10 פיקסלים:  
בעיה שנתקלנו בה:   
לפעמים ביצוע השחיקה גרם לפיצול ענן לשני רכיבים נפרדים, וכך גרם שאזור שלם לא סומן.  
בנוסף, ביצוע השחיקה אינה עקבית מספיק בתמונות שונות ולא ניתן לדעת מה יקרה בתמונה רנדומלית מהסט המקורי.*

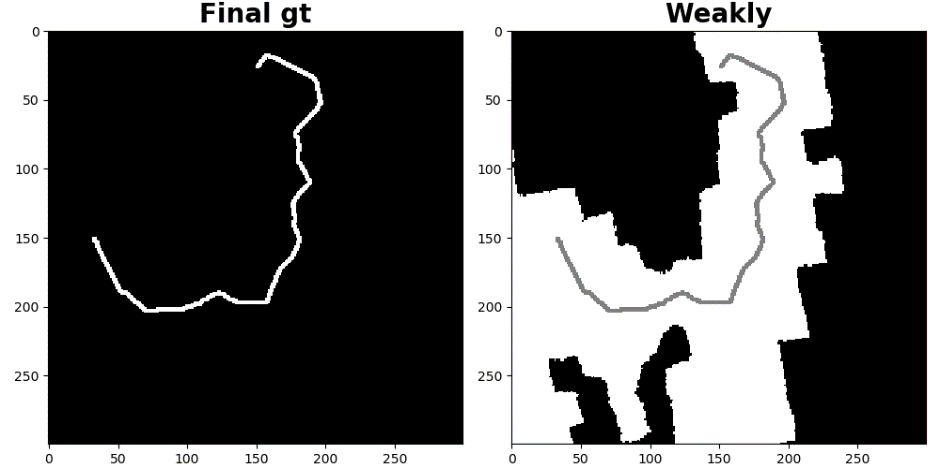
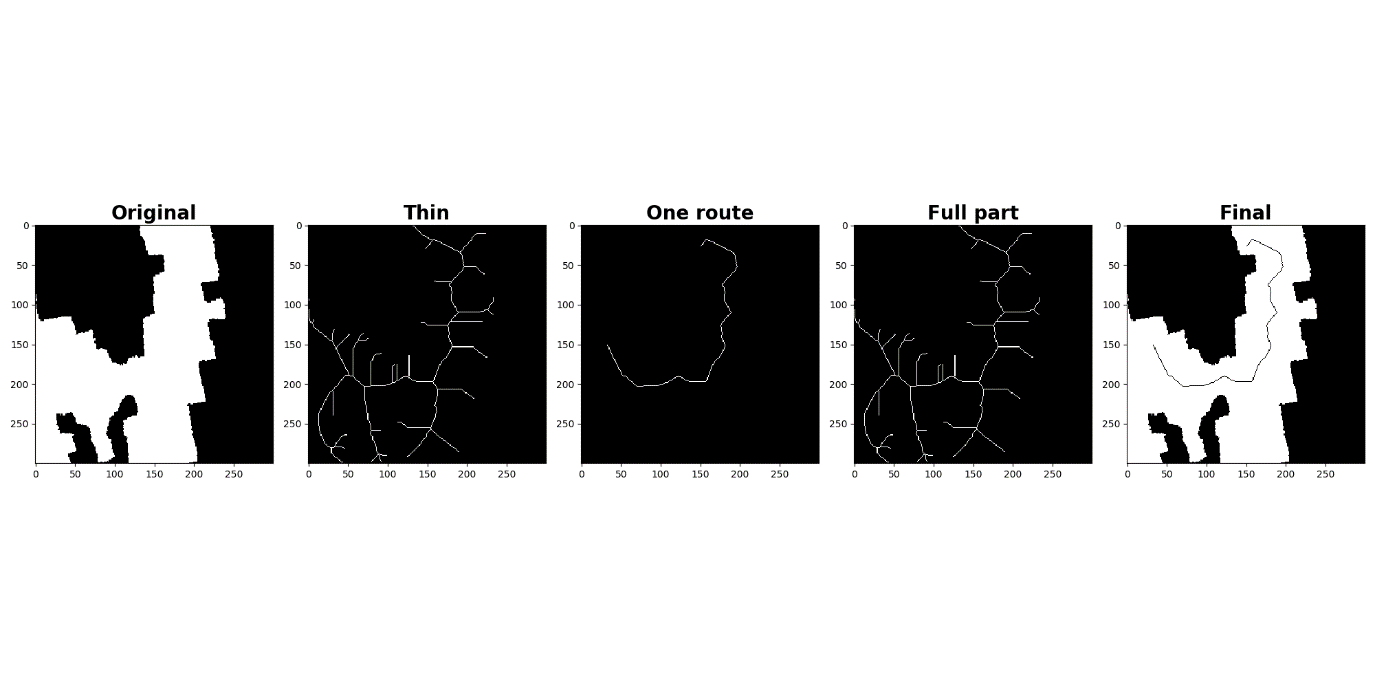
*פעולה מורפולוגית* ***thinning:****פעולה זו הינה פעולה המבצעת איטרציות של אלגוריתם hit or miss (ביצוע התאמת תבניות על ידי רכיבי בנייה שונים באמצעות שחיקה והרחבה) בתוספת פעולות נוספות, על מנת לקבל את "שלד" התמונה.  
דוגמא לתוצאות הרצת thinning על תמונת סגמנטציה בודדה:*

*תמונה שמכילה מפה

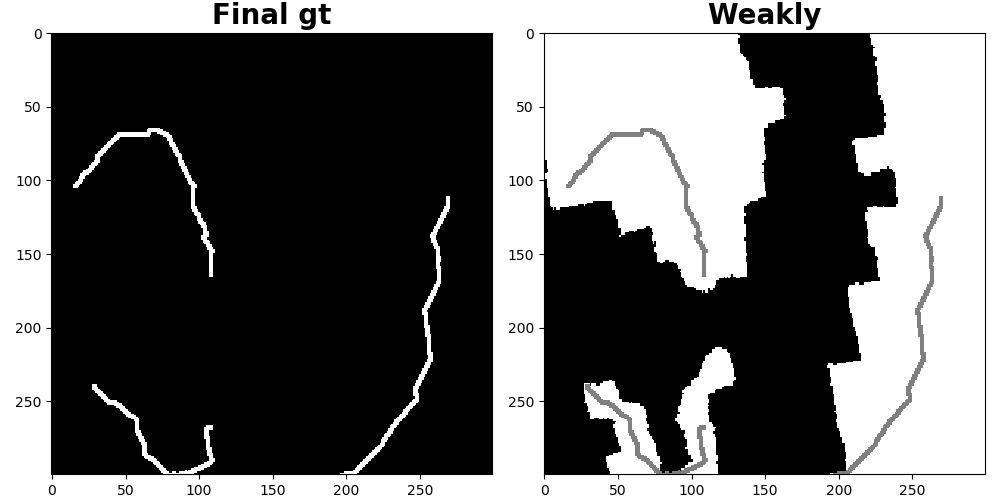
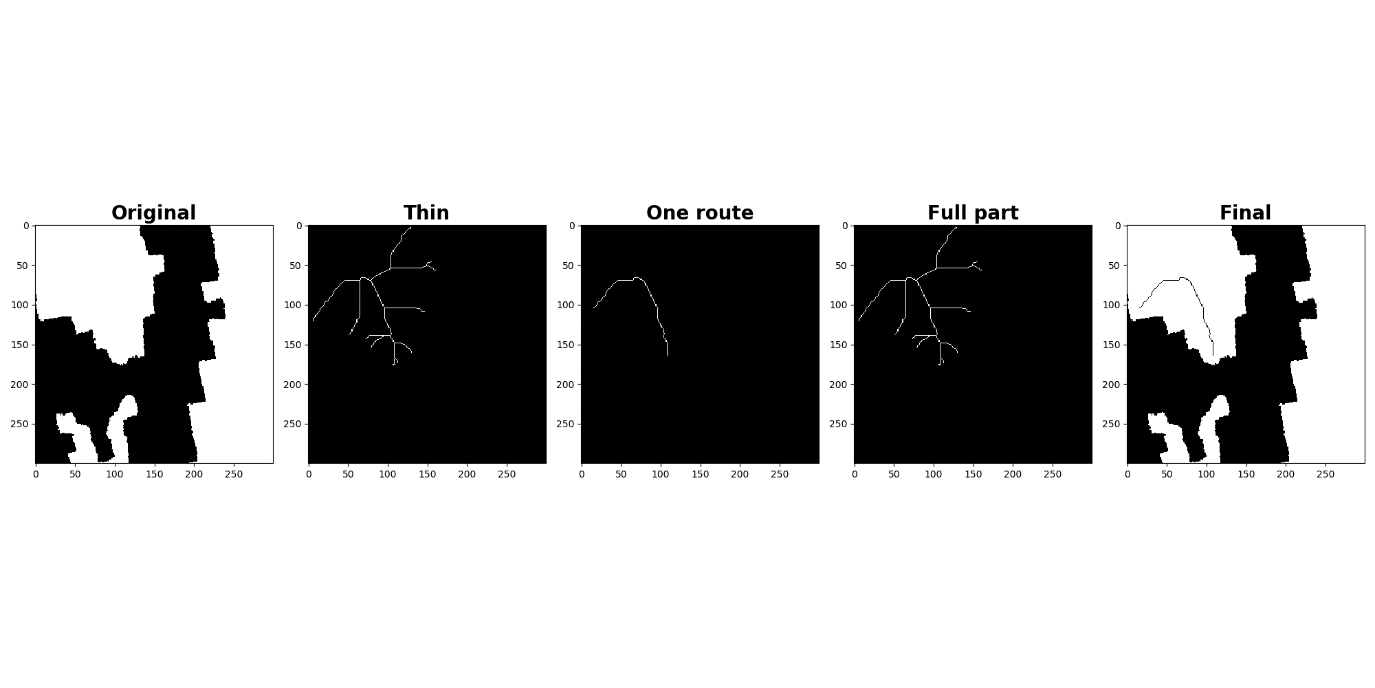
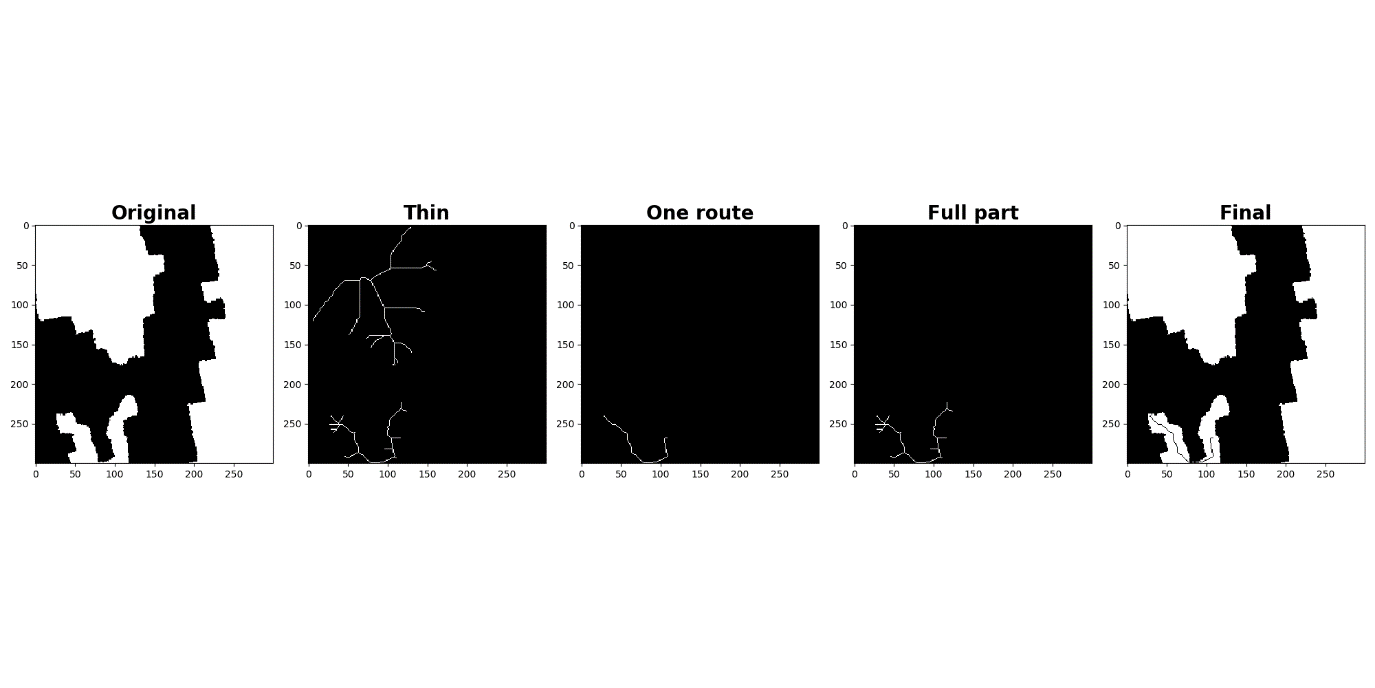
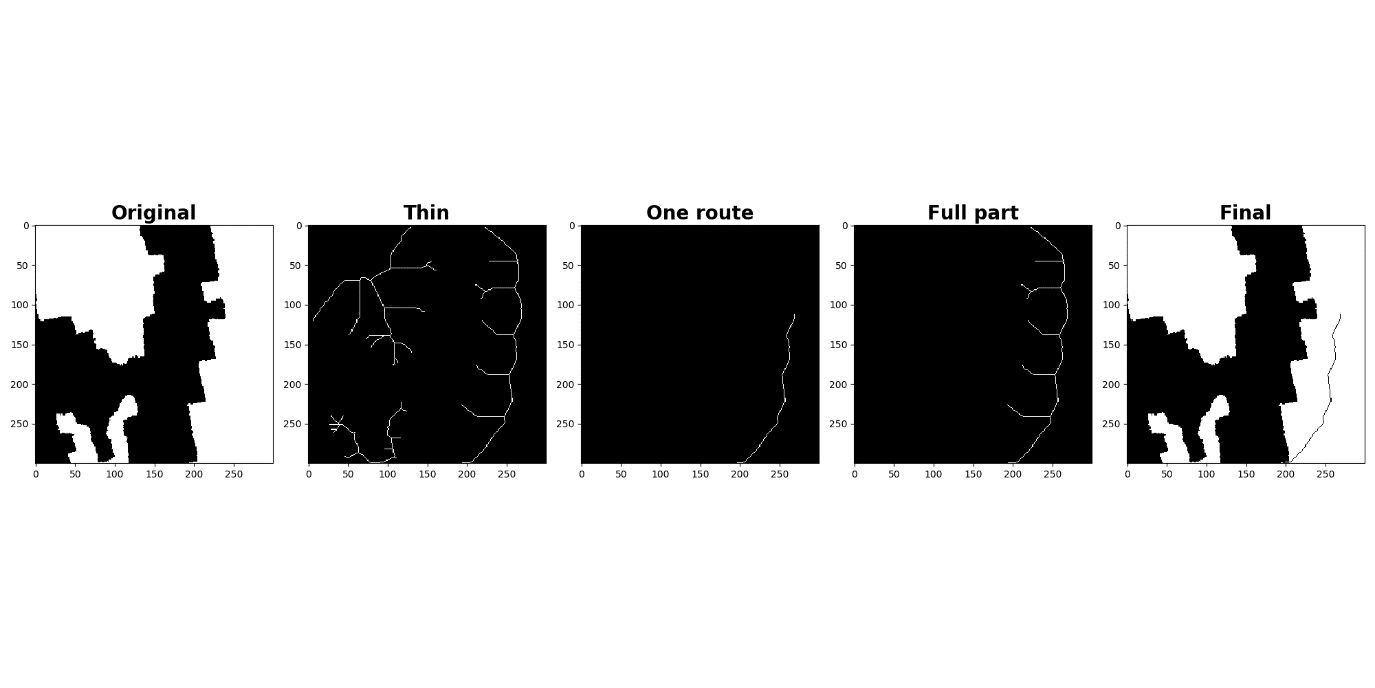
התיאור נוצר באופן אוטומטי  
ניתן לראות כי פעולה זו מצמצמת לנו את תיאור הסגמנטציה לצורה של גרף עם כמה רכיבי קשירות לפי מספר העננים הנפרדים בתמונה.  
על פי בדיקות נוספות של אלגוריתם זה, החלטנו להתקדם בעזרתו וליצור את השרבוטים על ידי שימוש בשלד שהוא יוצר.*

***אלגוריתם סופי ליצירת שרבוטים לתמונת סגמנטציה:*** *עבור קלט של seg תמונת הסגמנטציה המקורית:*

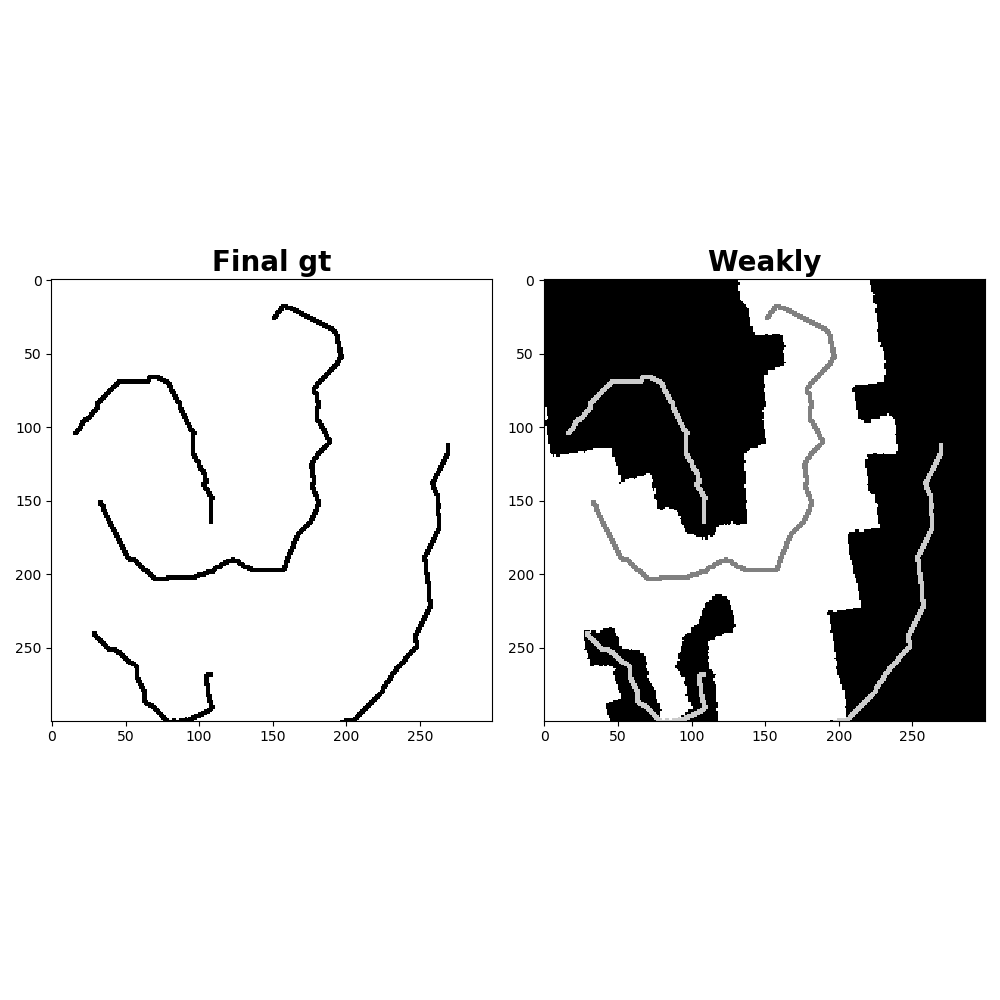
1. *ביצוע פעולת thinning על seg לקבלת שלד המתאר את הסגמנטציה, נסמן את התוצאה ב thin.*
2. *הגרלת 2 נקודות לבנות מתוך thin.*
3. *שימוש באלגוריתם חיפוש כלשהו (בקוד שלנו השתמשנו ב GreedyBest), על מנת לחבר בין הנקודות בתוך thin. במקרה בו לא ניתן לחבר את הנקודות (בשני רכיבי קשירות שונים), נחזור לשלב 2.  
   נסמן את המסלול בין שתי הנקודות ב route.*
4. *שימוש באלגוריתם BFS על thin החל מאחת הנקודות בסעיף 2, על מנת למצוא את רכיב הקשירות המלא המכיל את route. נסמן רכיב זה ב comp.*
5. *נגדיר: thin = thin – comp  
   על מנת לקבל את שלד הסגמנטציה ללא הרכיב שסימנו.*
6. *אם thin שונה מתמונה שחורה לגמרי, חזור לסעיף 2 (תמונה שחורה לגמרי = עברנו על כל העננים בתמונה).  
   אחרת, סכום את כל route, נסמן סכום זה ב c\_scribbles.*
7. *בצע הרחבה (dilate) באמצעות רכיב בנייה של ריבוע עם אורך צלע = 3 פיקסלים על c\_scribbles, על מנת להרחיב את השרבוט.*
8. *בצע c\_scribbles = c\_scribbles \* seg , על מנת למחוק פיקסלים שיצאו מגבולות הסגמנטציה בסעיף 7.*
9. *הגדר seg = 1 – seg , על מנת לקבל את התמונה המשלימה לסגמנטציה (כעת השמיים בלבן והעננים בשחור).*
10. *חזור על שלבים 1-8 עם seg החדשה. נסמן את תוצאות סכום ה route המתקבלת ב s\_scribbles.*
11. *הגדר תמונה חדשה המכילה 255 בכל פיקסל נסמנה ב final (אינדקס פיקסלים לא מסומנים יהיה 255).  
    final[c\_scribbles==1] = 1 (אינדקס עננים יהיה 1)  
    final[s\_scribbles==1] = 0 (אינדקס שמיים יהיה 0)*
12. *החזר את final.*

***דוגמת הרצה:*** *שלבים 1-8:  
*

*שלבים 9-10:  
כל תמונה מכילה הרצה אחת של שלבים 1-8*

**

*שלבים 11-12:  
(ערכי השרבוטים עבור העננים ועובר השמיים שונים)*

**