פרויקט זיהוי נתיבים (Python + OpenCV)

הקלט יהיה תמונת כביש שמצולמת מקדמת הרכב לעבר כיוון הנסיעה.

את הקלט יש להמיר מתמונה מלאת צבעים לתמונה לניתוח בעזרת canny edge detector:

Canny -> The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.



A picture containing way, road, scene, sky

Description automatically generated

Gradient -> מייצג את השינוי צבעים בין אובייקטים בתמונה -> כלומר על ידי מדידת השינויים ניתן לזהות ״מסגרת״

הגישה:

על מנת שהCanny יוכל לעבוד כמו שצריך הוא צריך תמונה אפורה. ולכן יש להמיר את התמונה לשחור לבן.

צריך להפוך 3 ערוצים של ירוק אדום וכחול וערוץ יחיד של אפור בטווח של [0,255] -> 0 שחור.

צבעי הנתיבים:

צבעי קווי ההפרדה בדרך כלל לבנים או צהובים.

עם הצבע הלבן אין לנו בעיה ממשית אבל הצבע הצהוב קצת קשה להפרדה כשאנחנו מדברים על צבעים מבסיס הRGB – מסיבה זאת נפריד לבסיס צבעים HSV (גוון/רוויה/בהירות).

מעבר לGaussian Filter:

המעבר לפילטר נועד על מנת להפחית רעשים בתמונה ומתן יכולת ניתוח גבוהה יותר -> יש לשים לב כמה שCanny רגיש לרעשים ולכן ננסה למנוע כמה שיותר.

שימוש בספריית Canny:

לאחר שימוש בתיקיית Canny יתקבל פלט של השינויים הבולטים בתמונה -> מה שמזוהה כסימון הנתיבים.

בנוסף לפלט יתקבל צירים (X,Y) ובהם מתאפשר לסמן 3 נקודות אשר מסמנות את כיוון או צורתו הכללית של הכביש (כולל לקחת שוליים רחבים)

![Chart

Description automatically generated]()



יצירת משולש העניין:

יצירת משולש העניין יעשה לפי נקודות המשולש שסומן למעלה וזה על מנת שניסיון הזיהוי יתבצע אך ורק על האזור התחום שמעניין את הרכב.

החישוב הבינארי:

בגלל שהמטריצה מייצגת את גוון הצבעים שבין שחור (0) ללבן (255) -> כל תא במטריצה יהיה ביט יחיד (אשר יכול להכיל מספרים מ-0 עד 255 לפי החישוב הבינארי).

החישוב עצמו נעשה בשיטת bitwise והחישוב הינו לפי & (וגם) -> אפשר למצוא הסבר פשוט ביוטיוב.

השוואת ה״וגם״ בין המשולש שנוצר לתמונה שעברה Canny תוביל לתוצאה שכל פיקסל שאינו צבוע בלבן ב2 האובייקטים יהיה שחור לחלוטין -> אך ורק האזור ה״מעניין״ בתמונת הCanny יכלול סימונים לבנים.

(bitwise של 255 משאיר ללא שינוי בעוד bitwise של 0 משנה לשחור לחלוטין).

מציאת הישר:

התמונה שהתקבלה עד כה מראה בבירור לעין אנושית מה הנתיבים שעל הכביש – ניתן לזהות אותם על פי צורת הקו הישר. על מנת שהמחשב יוכל לזהות את הקו הישר ניתן לסמן את כלל הנקודות הלבנות בכל ״כתם״ לבן על התמונה ולמצוא את הקו הישר המקורב שהנקודות נמצאות עליו.

ניתן לעשות את זה לפי משוואת הישר Y=aX+b אך במשוואה זאת לא ניתן למצוא ישר מאוזן ולכן משוואת הישר שתתאים במקרה זה הינה P=Ysin0 + Xcos0 שמייצגת את המרחק בזווית 0 (טטא) מהראשית.

חישוב ומציאת הישר יתבצעו בצורה הבאה:

* השמת כל הנקודות הלבנות בGRID.
* מציאת המשבצת בעלת מספר הישרים הגבוהה ביותר.
* יצירת ישר בעל הזווית והמרחק המתאימים (לפי מיקום המשבצת).

כמובן שככל שהמשבצות יהיו קטנות יותר (יכללו טווח נתונים קטן יותר) האלגוריתם יהיה סלחן פחות וכלל שהמשבצות יהיו גדולות יותר (יכללו טווח נתונים רחב יותר) האלגוריתם יהיה סלחן יותר אך פחות מדויק.

הערה: יש להגדיר threshold שיהיה המינימום הנדרש של קווים במשבצת בודדה.

Important links:

<https://medium.com/analytics-vidhya/building-a-lane-detection-system-f7a727c6694>

<https://www.youtube.com/watch?v=eLTLtUVuuy4>

<https://github.com/rslim087a/road-image/blob/master/test_image.jpg>

<https://github.com/rslim087a/road-video>