Dokumentacja – thunder sparrows

# Zadanie 1.

Na początku algorytm na podstawie danej mapy tworzy mapę kosztu. Koszt przejazdu do danego piksela można obliczyć użyć wzoru na koszt przejazdu do danego piksela:

Jeżeli tę wartość przemnożymy element-wise przez macierz binarną, czy dany piksel jest drogą otrzymamy mapę kosztu dla wszystkich pikseli-dróg.

Następnie stworzony został własnoręcznie algorytm djikstry – utworzona została tablica wartości tymczasowych – czy dany piksel został odwiedzony, położenie piksela z którego prowadziła droga o najmniejszym koszcie oraz wartość kosztu dotarcia do tego piksela.

Następnie na stos został dodany piksel początkowy o zerowym koszcie dotarcia.

Sam algorytm djikstry polega na

1. pobraniu koordynatów piksela ze stosu
2. Jeżeli piksel był odwiedzony – idź do punktu pierwszego
3. Jeżeli piksel jest pikselem końcowym –zakończ algorytm
4. Oznacz piksel jako odwiedzony, zapisz wierzchołek skąd prowadzi najkrótsza trasa do niego, oraz koszt dostania się do tego piksela
5. Wśród wszystkich sąsiadów danego piksela dodaj do stosu wszystkie wierzchołki nieodwiedzone wraz z kosztem dotarcia do nich (koszt danego piksela + koszt dotarcia do piksela właśnie zdjętego ze stosu)
6. Posortowaniu stosu według kosztu tak, by piksel znajdujący się na jego pierwszym miejscu do zdjęcia miał najmniejszy koszt dotarcia do niego
7. Powrót do punktu 1

Następnie na podstawie tak stworzonej mapy stworzono listę kolejno odwiedzonych pikseli zaczynając od końca trasy według następującego algorytmu zaczynając od piksela końcowego:

1. Dodaj dany piksel na początek listy
2. Jeżeli jest to punkt startowy przerwij algorytm
3. Koordynaty danego piksela zamień na koordynaty piksela skąd prowadzi najkrótsza ścieżka do niego
4. Wróć do punktu 1

Tak zbudowaną listę algorytm zwraca jako wartość wynikową.

Wizualizacja wyniku algorytmu została zaprezentowana w programie „task\_1\_visualize.mlx”.

# Zadanie 2.

W zadaniu drugim została wykorzystana metoda kaskad Haar’a do wykrywania znaku D1. Pomysł polega na wykorzystaniu danych treningowych w celu ekstrakcji cech, a następnie użyciu ich podczas właściwej detekcji. Do początkowego treningu został wykorzystany wstępnie przetrenowany do detekcji znaków model, który następnie był uczony na całym dostępnym zbiorze treningowym z podziałem na próbki negatywne i pozytywne. Gotowa funkcja detekcji pobiera ścieżkę do folderu ze zdjęciami i zwraca dane w wymaganej przez zadanie formie tj. struktury z wymienionymi obrazami oraz wykrytymi ramkami otaczającymi. W planach było również wykorzystanie uogólnionej transformaty Hougha w celu filtrowania detekcji dokonanych metodą kaskad Haar’a, co zwiększyło by jakość rozwiązania, niestety z powodu ograniczeń czasowych nie udało nam się zaimplementować tego algorytmu.

Wynik przykładowej detekcji z wykorzystaniem kaskad Haar’a:Obraz zawierający na wolnym powietrzu, pojazd, Pojazd lądowy, droga

Opis wygenerowany automatycznie

# Zadanie 3.

Sterowanie obsługuje funkcje automatycznej zmiany kierunku wycieraczki, kiedy zbliży się do krawędzi szyby, ograniczenie napięcia do przedziału [-12V, 12V] i natężenia do [-12A, 12A], 3 tryby szybkości oraz stabilizację prędkości wycieraczki.

Zmiana kierunku wycieraczki jest obsługiwana przez blok *Relay*, który przy zbliżeniu się do krawędzi wysyła sygnał sterowania o przeciwnym znaku, do poprzedniego.

Za ograniczenie przedziału napięcia odpowiada blok *Saturation*.

Natężenie przekraczało progi przy szybko zmieniającym się napięciu, co ograniczył blok *Rate limiter*.

Za stabilizację prędkości sterowania odpowiada ręcznie napisany regulator PI.

Regulacja prędkości jest obsługiwana przez blok Switch *case*, który w zależności od trybu prędkości na wyjście daje odpowiedni mnożnik napięcia sterowania oraz parametru P w regulatorze PI.