REV	DATA	ZMIANY
0.1	10.06.2025	*Jakub Stelmach (stelmachj@student.agh.edu.pl)*

ROBOT ŚLEDZĄCY LINIĘ (LINE FOLLOWER)

Algorytm oraz implementacja czujnika wykrywania linii wraz z symulacją

Autor: inż. Jakub Stelmach
Akademia Górniczo-Hutnicza

Spis treści

Lista oznaczeń

LF	Line Follower
DSP	Digital Signal Processing
PWL	Piecewise Linear
PCB	Printed Circuit Board

1. Wstęp

Niniejszy dokument dotyczy zaprojektowania oraz testów algorytmu wykrywania linii dla robota typu Line Follower. W projekcie wykorzystano istniejącą platformę sprzętową opracowaną w ramach wcześniejszej pracy inżynierskiej (J. Stelmach) [1]. Głównym celem projektu jest rozwój oraz opracowanie detekcji położenia linii, która będzie bardziej dokładna przy interpretacji środka linii niż wcześniejsze rozwiązanie bazujące tylko na metodzie środka ciężkości histogramu. Dodatkowo stworzone zostało narzędzie symulacyjne ułatwiające kolejne optymalizacje wraz z dokumentacją.

2. Wymagania

1. Dostęp do MATLAB (Online) [2]
2. Podstawowa umiejętność korzystania z oprogramowania MATLAB.

3. Funkcjonalność

W skład funkcjonalności projektowanego oprogramowania wchodzi:

1. Detekcja pozycji oraz parametrów linii w oparciu o opracowany algorytm.

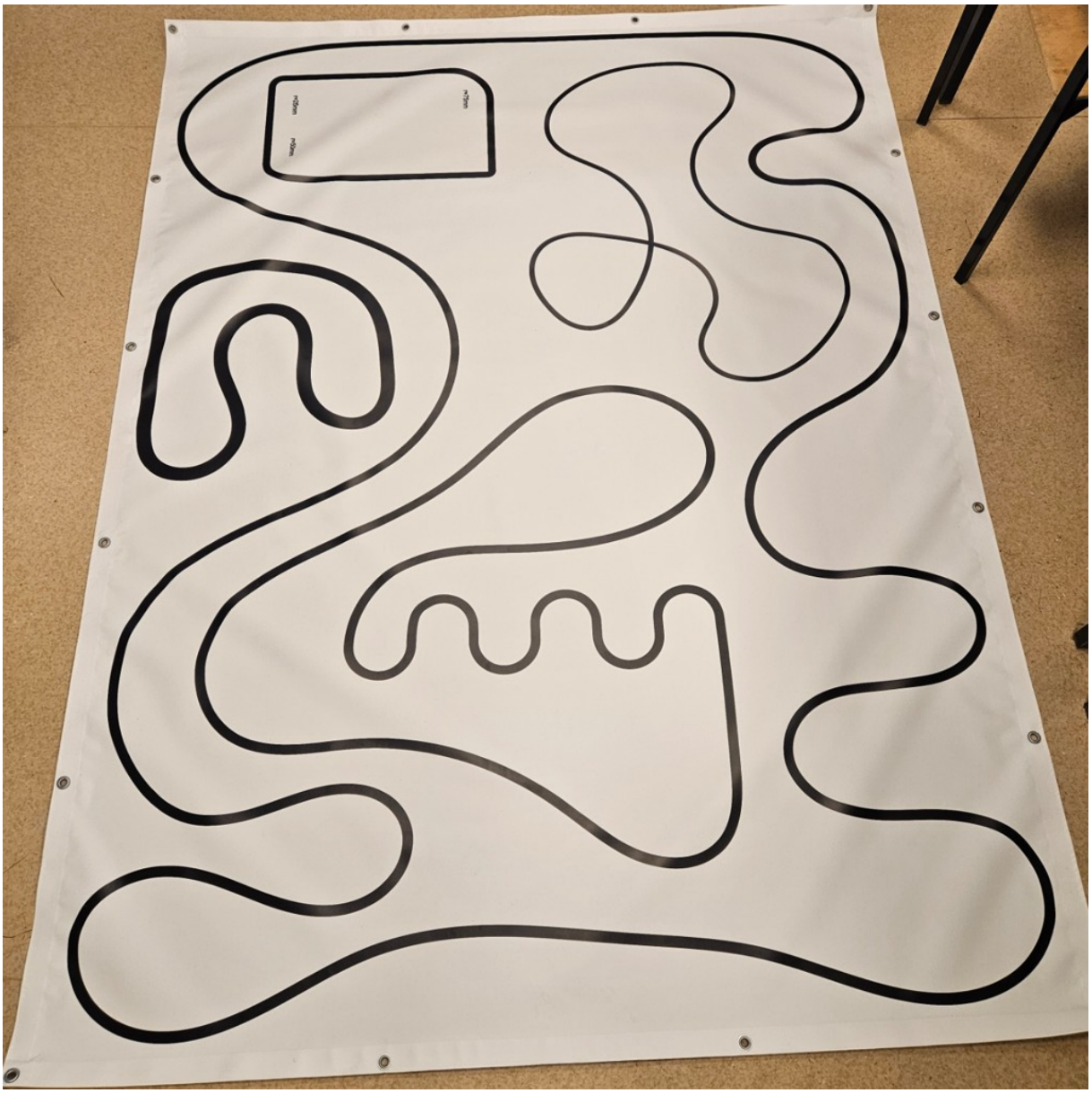
W skład funkcjonalności oprogramowania testującego wchodzi:

1. Możliwość wygenerowania danych testowych (odczytów z pojedynczych czujników w czasie).
2. Możliwość testowania algorytmu na wygenerowanych sztucznych danych.
3. Możliwość porównania działania algorytmu z danymi referencyjnymi.
4. Testowanie poprawności działania poszczególnych funkcji (testy jednostkowe)

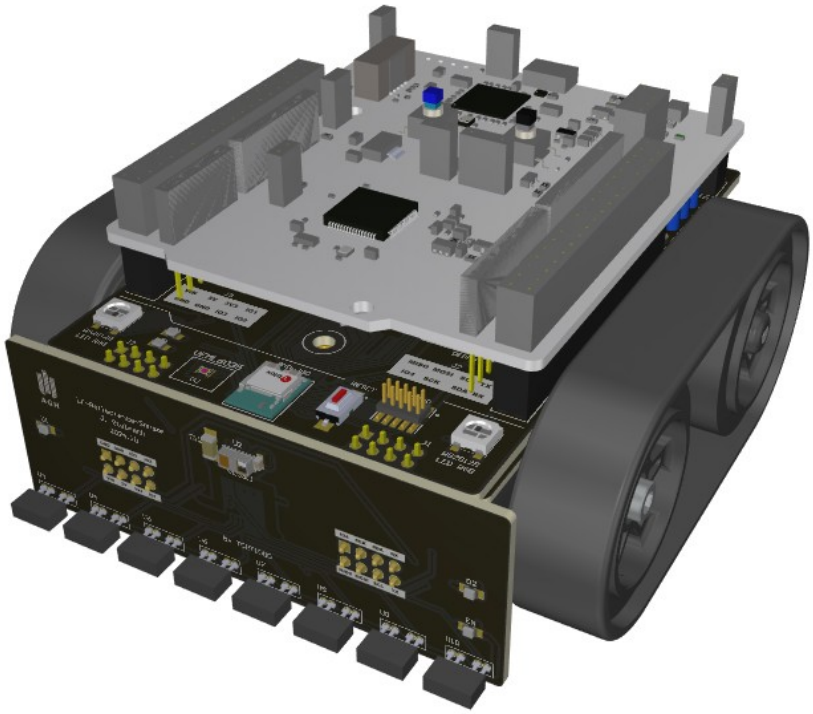
4. Analiza problemu

4.1 Budowa toru jazdy oraz czujnika linii Line-Followera

Podłożem toru jest biały materiał, na którym znajduje się linia w kolorze o wysokim kontraście. Taki wybór wynika z potrzeby zwiększenia dokładności działania czujników – silny kontrast ułatwia ich pracę i poprawia jakość odczytów. W tym przypadku podłoże jest białe a linia czarna, dodatkowo szerokość linii jest stała na całej długości pojedynczego toru.

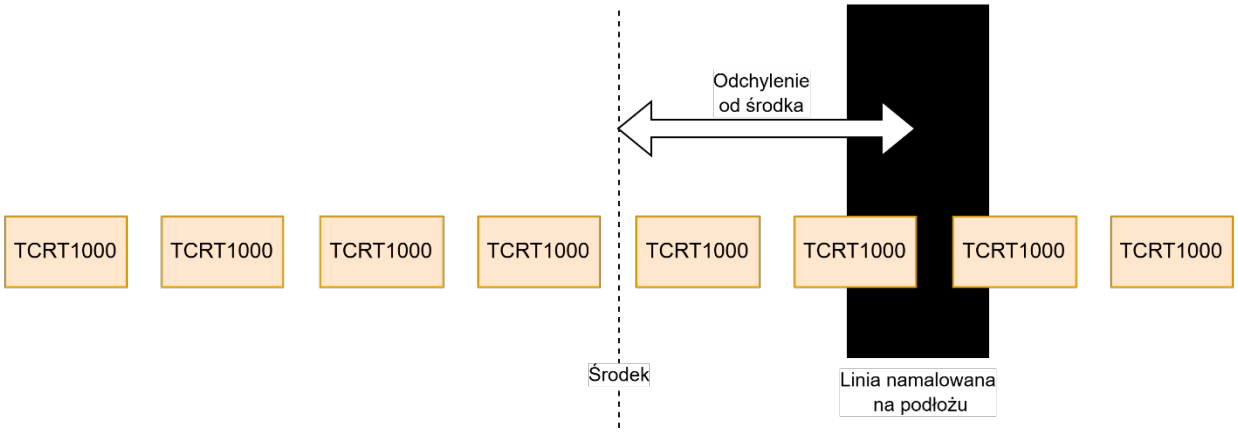


Rysunek 1 Zaprojektowany tor jazdy Line-Followera



Rysunek 2 platforma Line-Follower

Czujnik odpowiedzialny za wykrywanie linii składa się z ośmiu sensorów odbiciowych typu TCRT1000, które zawierają diodę emitującą podczerwień oraz fototranzystor na który pada światło odbite od podłoża. Sygnał w postaci napięcia z kolektora fototranzystora jest następnie odczytywany przez przetwornik analogowo-cyfrowy (ADC) zintegrowany z mikrokontrolerem STM32L051. Odczyt taki jest przechowywany w pamięci procesora jako tablica z ośmioma 12-bitowymi wartościami.



Rysunek 3 Ilustracja działania sekcji czujników TCRT1000 jako dyskretnego czujnika położenia względem linii (skala szerokości linii nie jest zachowana)

4.2 Zasada działania pierwotnego algorytmu

Pierwotny algorytm bazuje na wyznaczaniu środka ciężkości histogramu. Bardzo dobrze sprawdza się on w momencie kiedy cała linia znajduje się w zakresie pomiarowym czujnika (Rysunek 2). Problemy zaczynają się w momencie kiedy linia wychodzi poza zakres pomiarowy. Wtedy do wyznaczenia środka linii wykorzystywana jest niepełna informacja o jej szerokości.

$$x_{cm} = \frac{\sum_{i=1}^{n} x_i \cdot h_i}{\sum_{i=1}^{n} h_i}$$

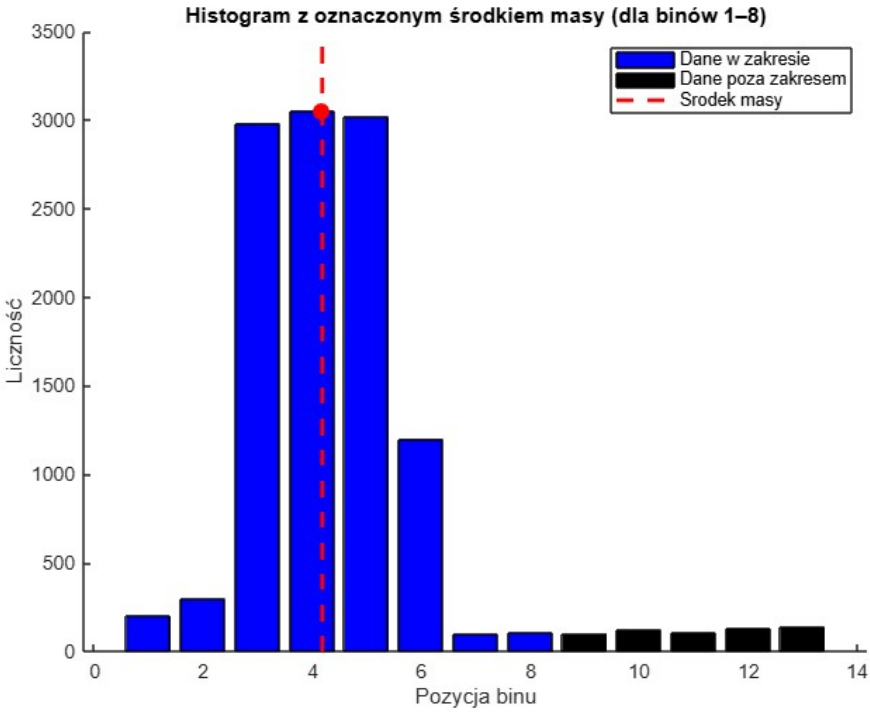
Wzór 1 Wzór na środek ciężkości histogramu

Gdzie:

$\mathbf{x_{cm}}$ to pozycja na osi x histogramu oznaczająca środek masy

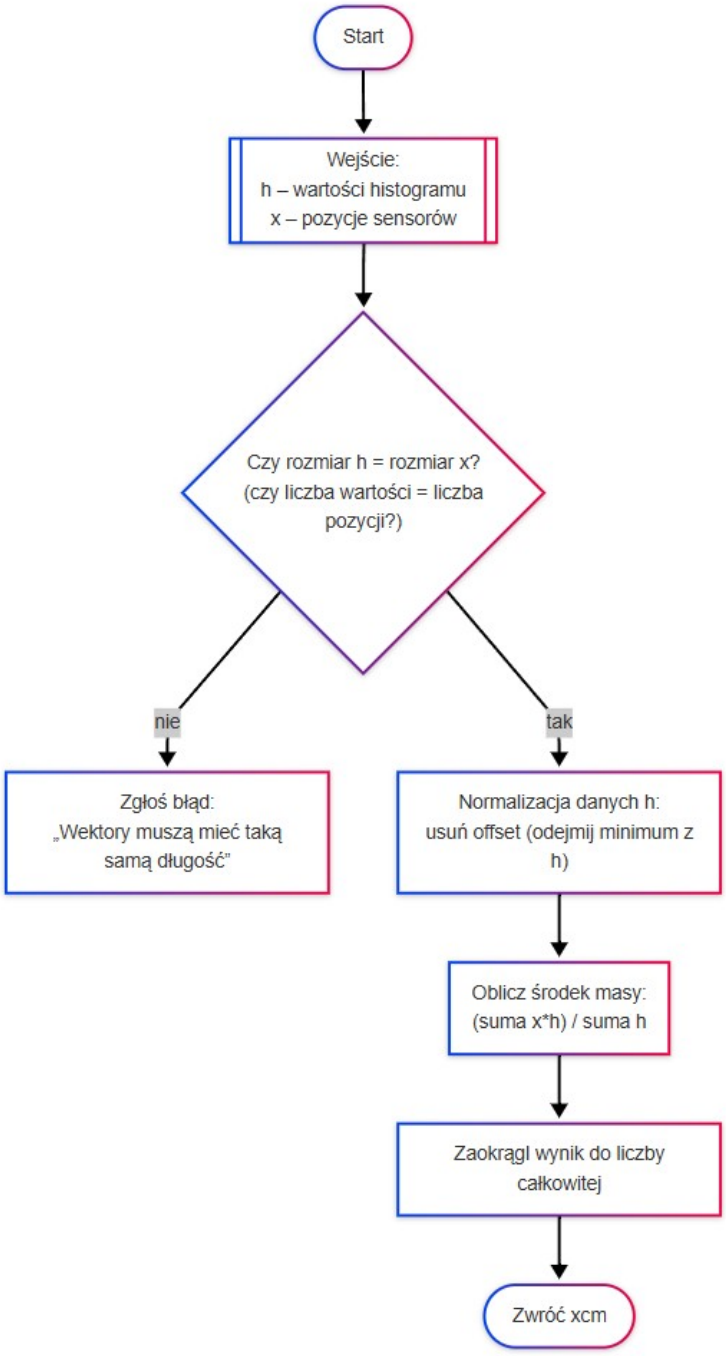
$\mathbf{x_i}$ – pozycja sensora względem początku PCB [mm] (i = 0, 10, 20, …, 70)

$\mathbf{h_i}$ – wartość histogramu w danym binie



Rysunek 4 histogram z oznaczonym środkiem masy

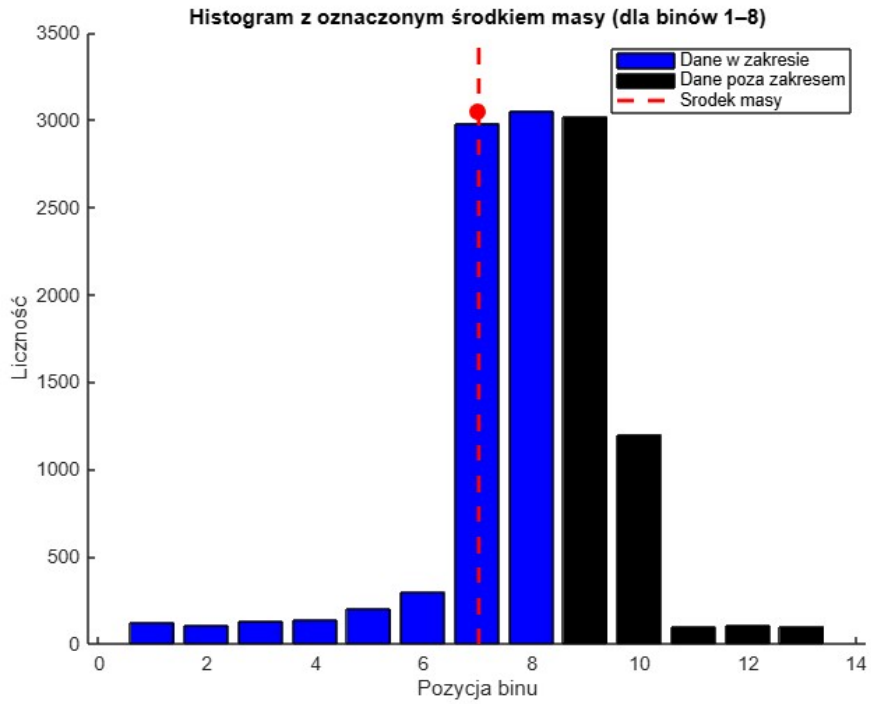
Łatwo zauważyć, że algorytm ten (Wzór 1) to po prostu średnia ważona, która uwzględnia wartości z wszystkich sensorów. Biny od 1 do 8 to wartości sensorów, reszta to „hipotetyczne” odczyty poza czujnikiem.



Rysunek 5 Schemat działania oryginalnego algorytmu

4.3 Wady pierwotnego rozwiązania

Problemem jest pomiar linii w momencie kiedy wychodzi ona poza krańce czujnika. W momencie jak tylko fragment linii jest odczytywany, wtedy jej środek interpretowany jest błędnie z dostępnych danych.



Rysunek 6 Histogram z błędnie określonym środkiem linii

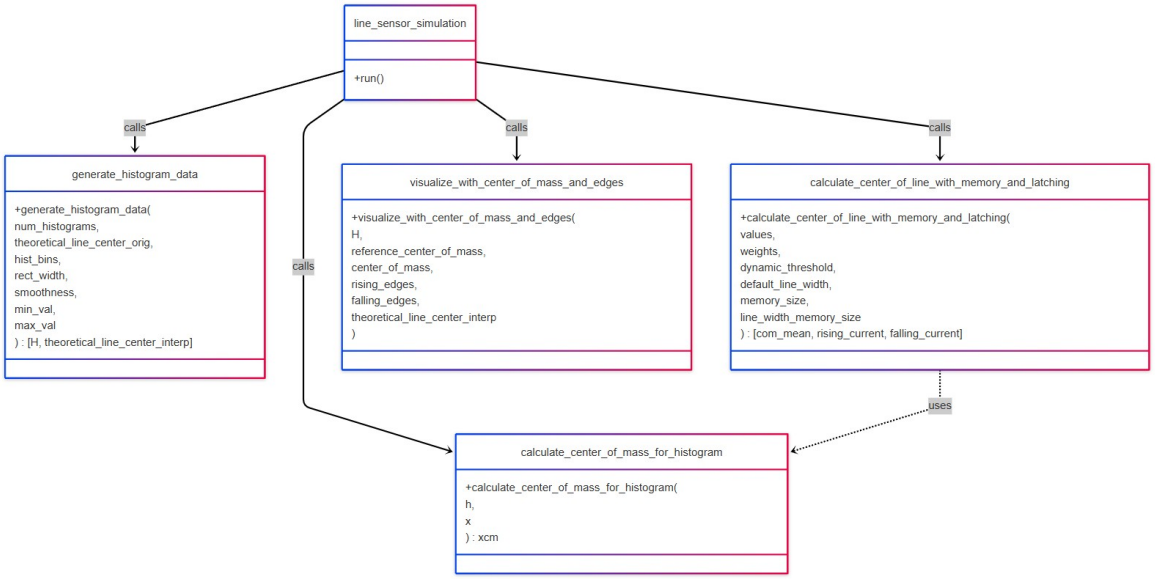
Jasne sprecyzowanie problemu jest niezbędne w celu opracowania optymalnego algorytmu.

5. Projekt techniczny

Projekt nowego algorytmu oraz symulacji opiera się o:

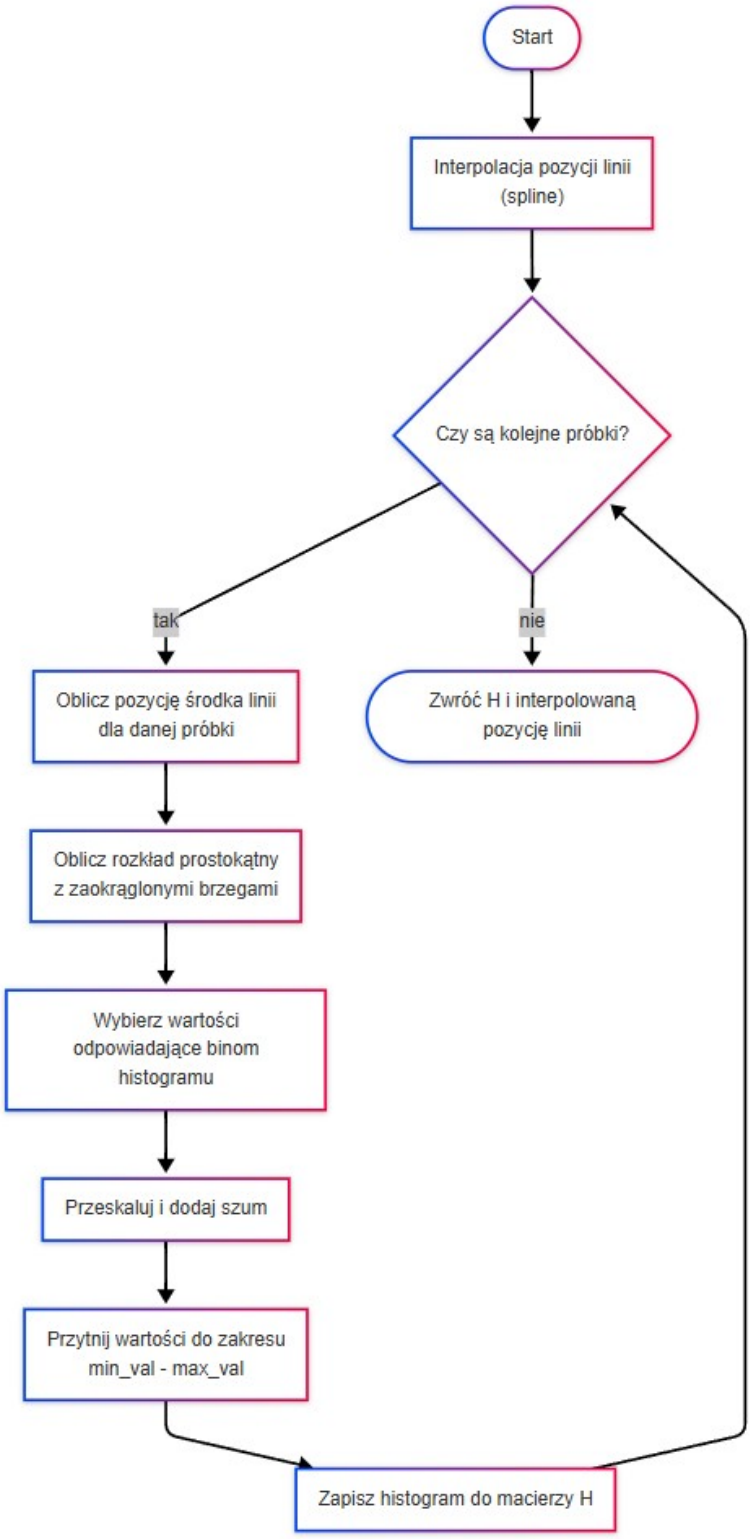
- Implementację oryginalnego algorytmu (schemat zawarty w poprzednim rozdziale)
- Implementacja nowego algorytmu
- Moduł symulujący matematycznie dane odczytywane przez czujnik
- Moduł symulacji uwzględniającej kilka wariantów tras
- Moduł wizualizujący dane w zestawieniu z algorytmem oryginalnym

Całość zaprojektowana, w celu umożliwienia testowania różnych wariantów tras.



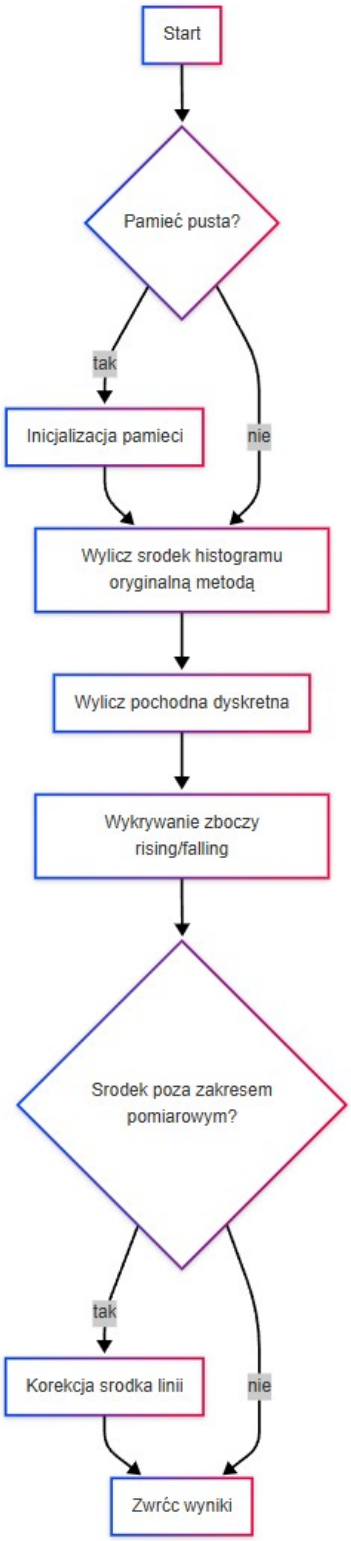
Rysunek 7 Diagram klas przedstawiający zaprojektowany symulator

5.1 Algorytm generujący macierz testową



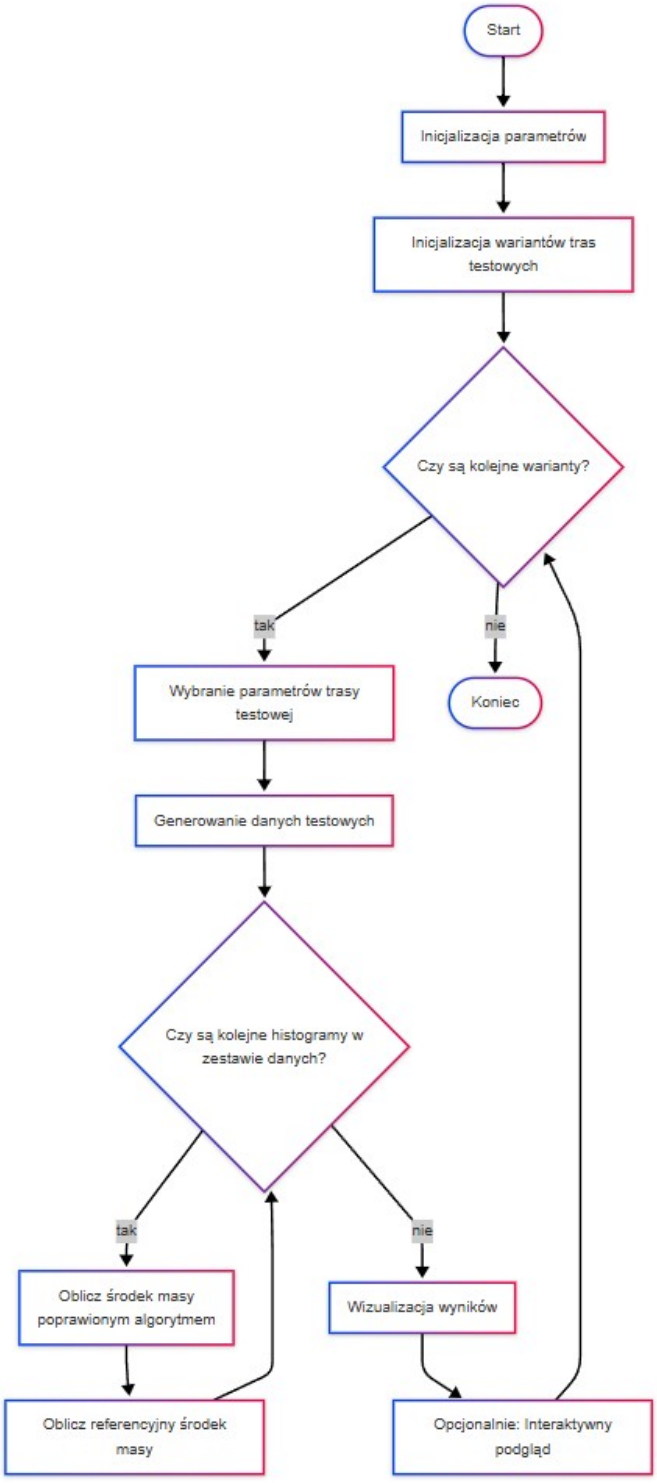
Rysunek 8 Algorytm generujący macierz testową – generate_histogram_data

5.2 Ulepszony algorytm wyliczający położenie linii



Rysunek 9 Ulepszony algorytm wykrywania linii - calculate_center_of_line_with_memory_and_latching

5.3 Główna symulacja



Rysunek 10 Schemat blokowy sekwencji symulacji line_follower_simulation

5.4 Algorytm wykreślający dane na wykresie temperaturowym

Wykres 1: Referencyjny algorytm

Zawiera:

- Mapę termiczną wizualizacji linii odczytywanej przez czujnik (funkcja imagesc)
- Centrum masy z referencyjnego algorytmu jako linię ciągłą
- Teoretyczne centrum linii jako linia ciągła (funkcja z której wyznaczane są histogramy)
- Oś X: Sensor Bin (1–8) – oznacza poszczególne sesnory
- Oś Y: numer odczytanego histogramu (kolejne odczyty w czasie)
- Legenda i tytuł: "Reference Algorithm"

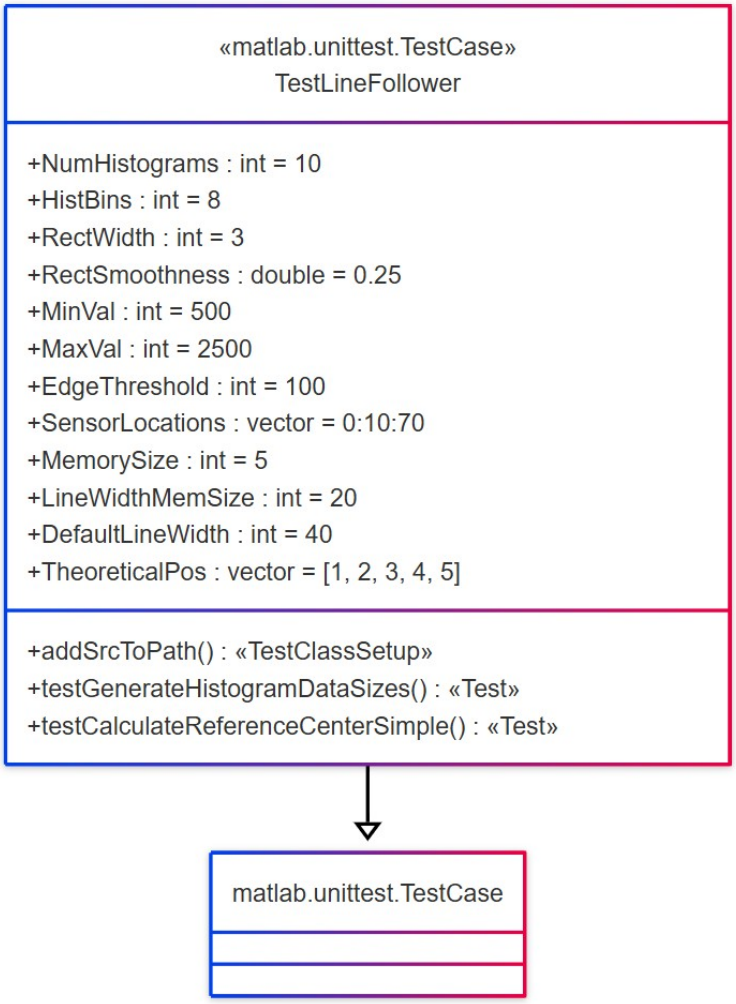
Wykres 2: Zmodyfikowany algorytm

Zawiera:

- Mapę termiczną wizualizacji linii odczytywanej przez czujnik (funkcja imagesc)
- Centrum masy z algorytmu zmodyfikowanego jako linia ciągła
- Krawędzie narastające (rising edges) jako linia ciągła
- Krawędzie opadające (falling edges) jako linia ciągła
- Teoretyczne centrum linii jako linia ciągła (funkcja, z której wyznaczane są histogramy)
- Oś X: Sensor Bin (1–8) – oznacza poszczególne sensory
- Oś Y: numer odczytanego histogramu (kolejne odczyty w czasie)
- Legenda I tytuł: "Modified Algorithm"

Oba wykresy zestawione obok siebie w celu porównania. Kolory dobrane tak aby łatwo można było odróżnić poszczególne elementy – wizualizacja linii na czarno natomiast tło na biało.

5.5 Testy jednostkowe

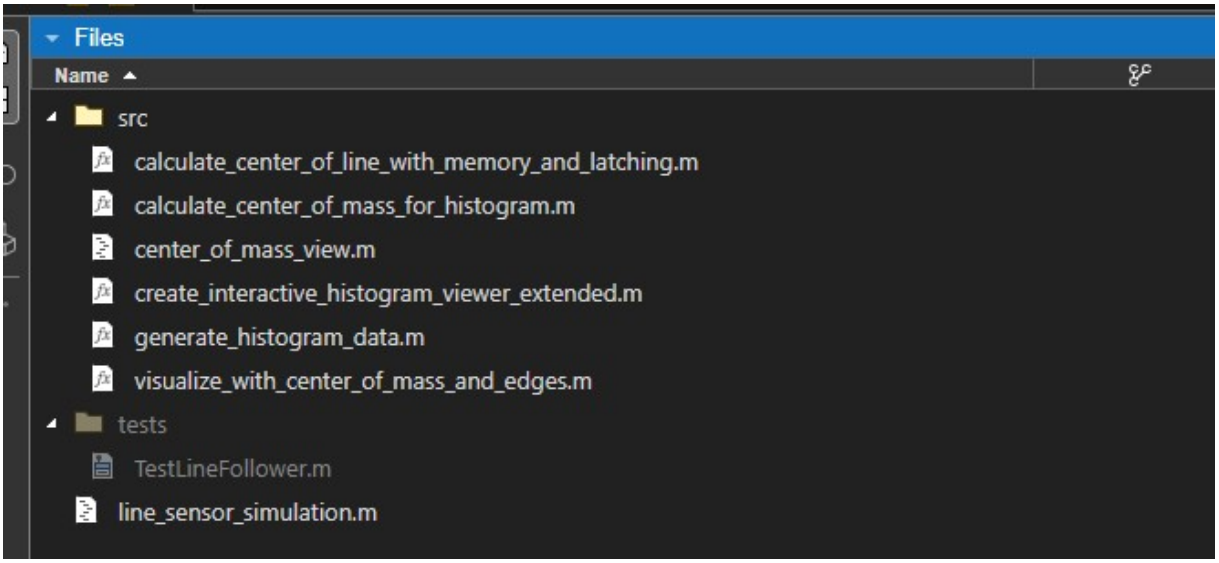


Rysunek 11 Diagram testów jednostkowych

Testy jednostkowe – celem jest analiza czy poprawnie, w ramach zadanych parametrów, tworzone są histogramy testowe symulacji. Funkcjonalność ta jest wbudowana w Matlaba, dzięki czemu jest bardzo prosta w użytkowaniu.

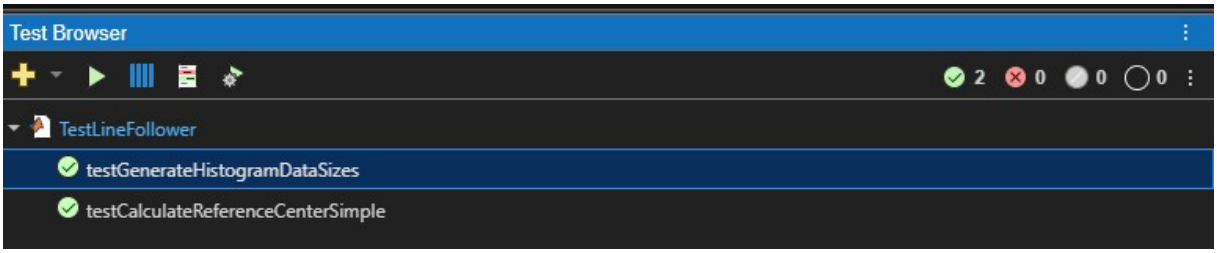
6. Opis realizacji

W celu zwiększenia uniwersalności, całość projektu została zapisana w kodzie MATLAB. Implementacje poszczególnych funkcjonalności zapisane są w postaci osobnych plików, aby zwiększyć czytelność kodu, a poszczególne segmenty są opisane zwięzłym komentarzem, w języku angielskim. Do uruchomienia wystarczy dostęp do środowiska MATLAB Online, a pełny kod jest dostępny na repozytorium git (https://github.com/JakubSt01/Line-Follower).



Rysunek 12 Pliki zawarte w projekcie

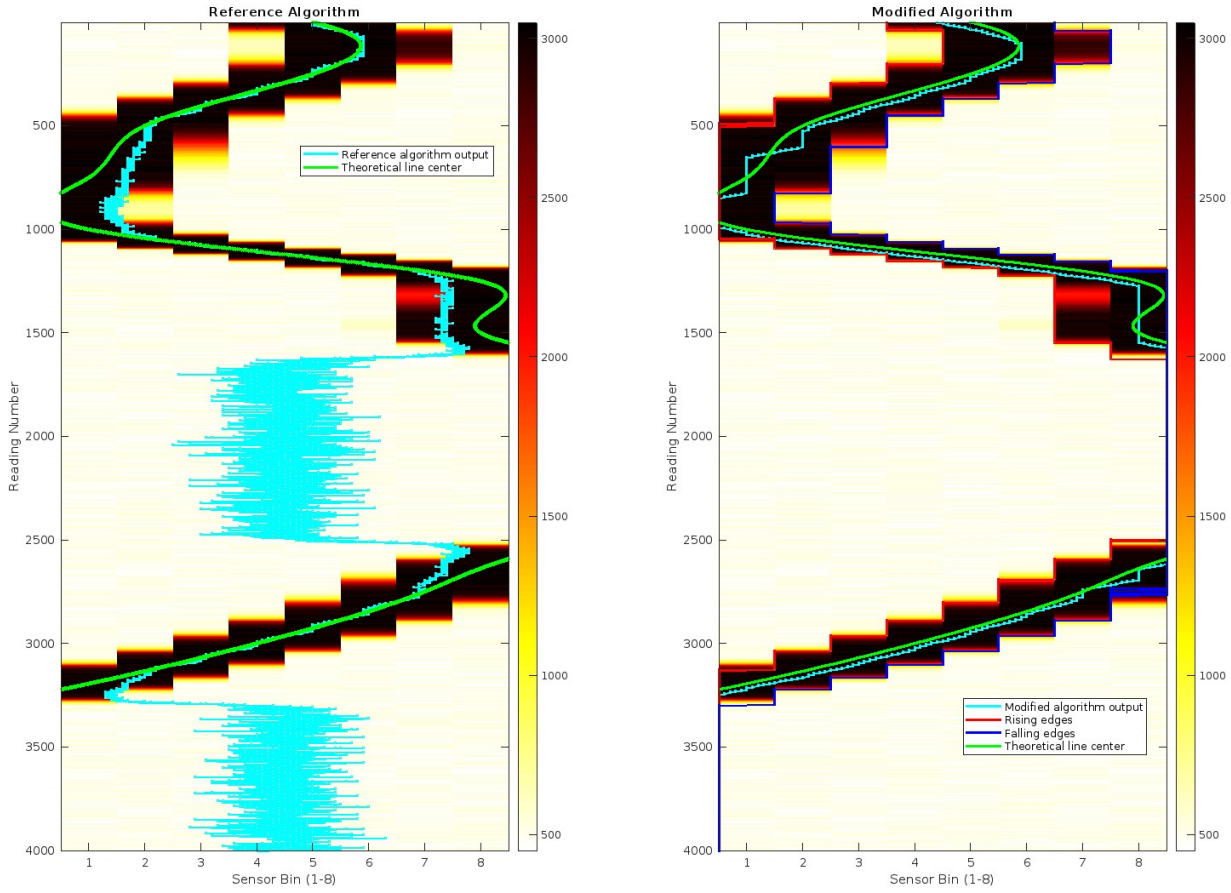
Ze względu na to, że rozwijana jest platforma edukacyjna – wykorzystanie tego środowiska jest pożądane, ponieważ studenci mają do niego bezproblemowy dostęp.



Rysunek 13 Wynik uruchomienia testów jednostkowych

6.1 Wizualizacja danych wyjściowych

Najbardziej kluczowym elementem projektu jest wizualizacja danych wyjściowych algorytmów, dzięki której jesteśmy w stanie porównać która z implementacji lepiej radzi sobie z wyznaczaniem środka linii.



Rysunek 14 Wizualizacja danych

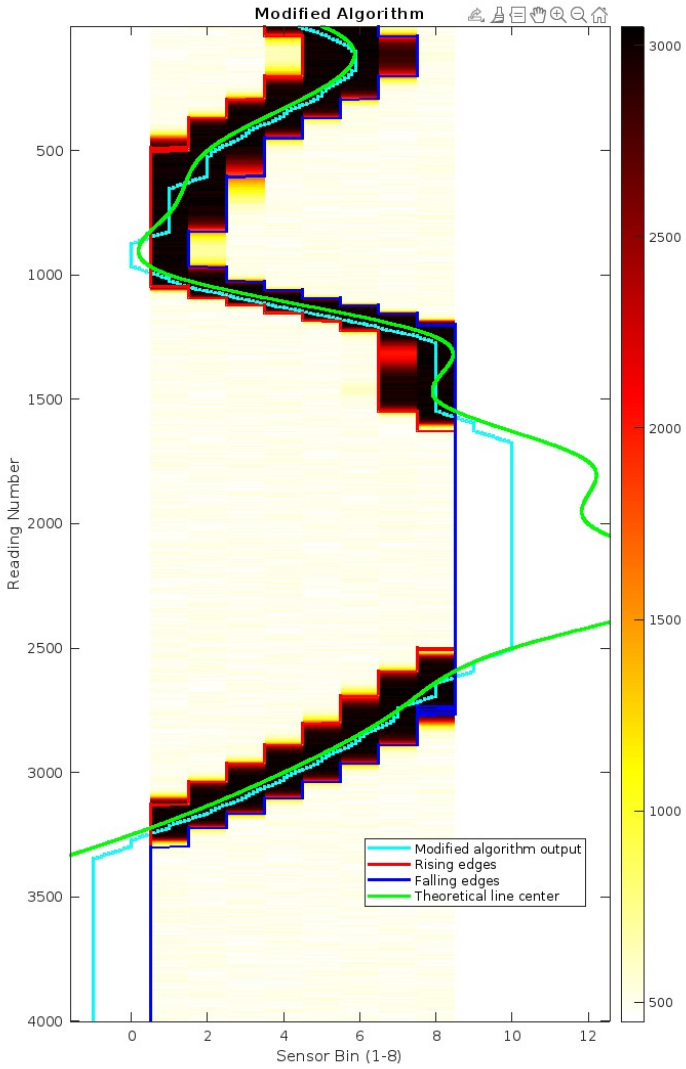
Dane wyświetlane na wyjściu obejmują:

1. Mapy temperaturowe wartości histogramu – na osi x jest numer binu, na osi y numer pomiaru (przebieg w czasie), po lewej efekt działania oryginalnego algorytmu a po prawej efekt działania poprawionego algorytmu.
2. Wykreślony wejściowy środek linii na którego bazie są generowane histogramy „Theoretical Line Center”
3. Wykreślony wyjściowy wyliczony środek linii „Algorithm output”
4. Wykryte zbocza na mapę temperaturową – Rising Edges, Falling Edges

Celem działania algorytmu jest określenie jak najprecyzyjniej rzeczywistego położenia linii. Na powyższych wykresach widzimy że oryginalny algorytm radzi sobie do momentu kiedy linia zbliża się do jego krawędzi lub całkowicie znika z pola widzenia. Dodatkowo nie radzi on sobie z szumem i wartość położenia oscyluje wokół docelowej wartości. Dodatkowo widać efekt kompresji spowodowany wyliczaniem środka z niepełnej linii.

W nowym algorytmie dzięki zastosowaniu wykrywania zboczy i szacowaniu położenia linii jesteśmy w stanie uzyskać precyzję pomiaru niemal w całym zakresie pomiarowym czujnika. Dodatkowo dzięki pamięci poprzednich stanów (średnia krocząca) następuje efekt filtrowania danych co diametralnie zmniejsza poziom szumów.

Kolejnym z problemów było zgubienie linii w trakcie jazdy. Pojazd, ze względu na szum interpretował brak linii jako oscylacje wokół środka czujnika co objawiło się jego jazdą w bliżej nieokreślonym kierunku. Wada ta została wyeliminowana dzięki śledzeniu zbocza.



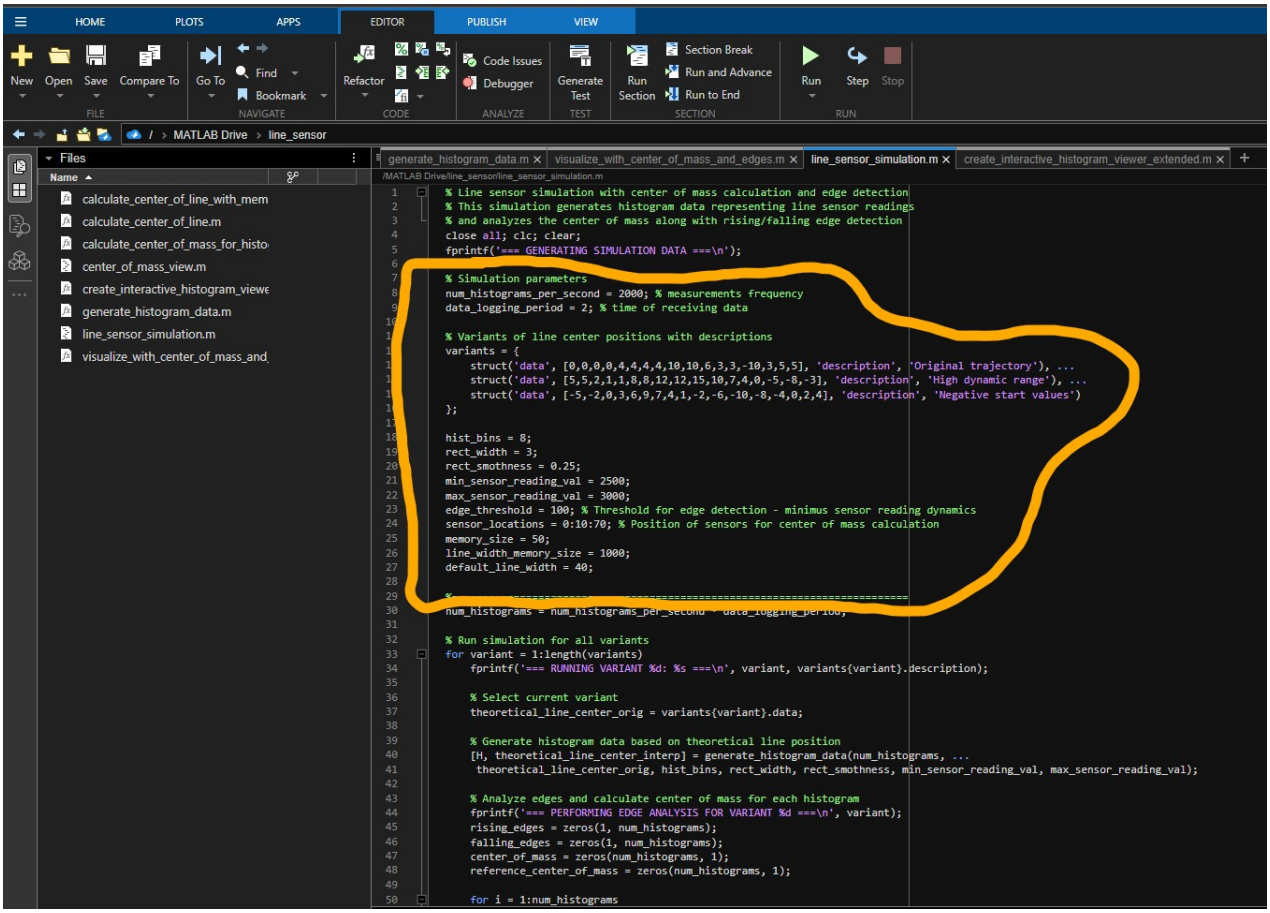
Rysunek 15 Zalety ulepszeń algorytmu

Dodatkowo dzięki określaniu środka linii przy wykorzystaniu informacji o jej szerokości i położeniu zboczy, jesteśmy w stanie wychylenie wyznaczać nawet poza rejestrowanym zakresem pomiarowym sensora.

7. Podręcznik użytkownika

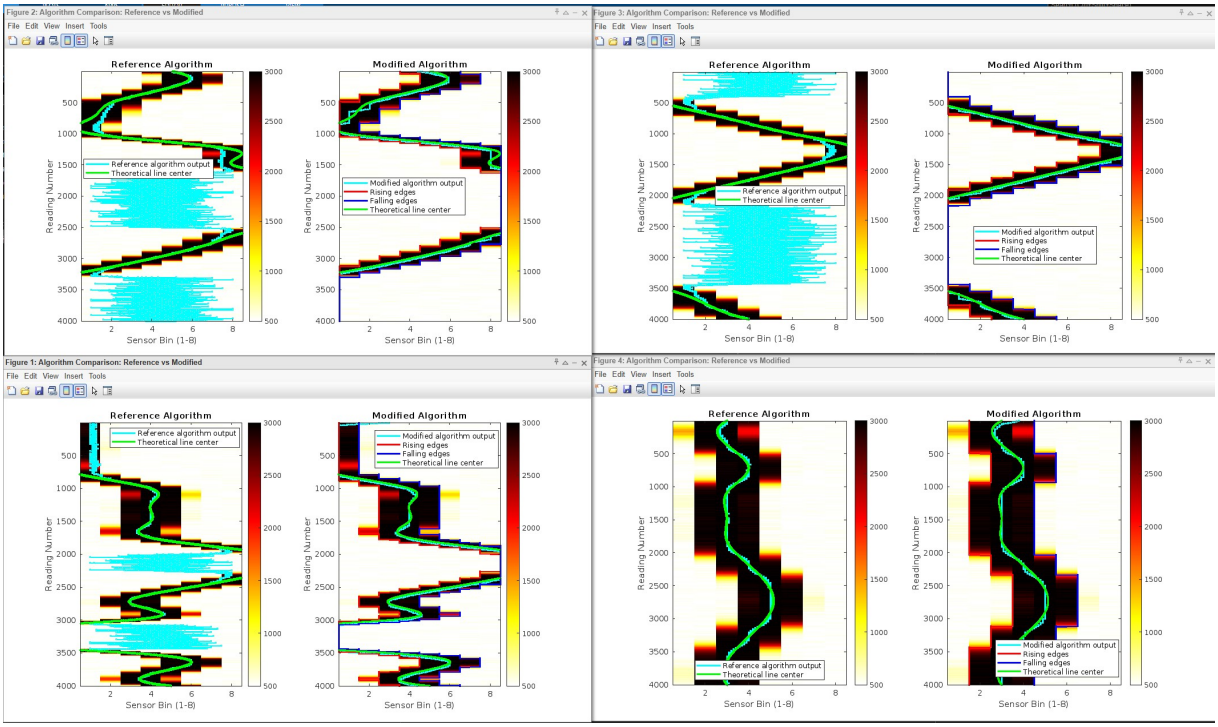
7.1 Uruchomienie aplikacji w MATLAB Online

1. Otwieramy folder z projektem w MATLAB Online, konfigurujemy parametry symulacji a następnie uruchamiamy symulację zielonym przyciskiem **Run.**



Rysunek 16 Sekcja parametrów symulacji, linii oraz algorytmu czujnika wraz z wariantami

2. Po uruchomieniu symulacji widzimy symulacje z wszystkimi wariantami przewidzianymi na etapie parametryzacji.

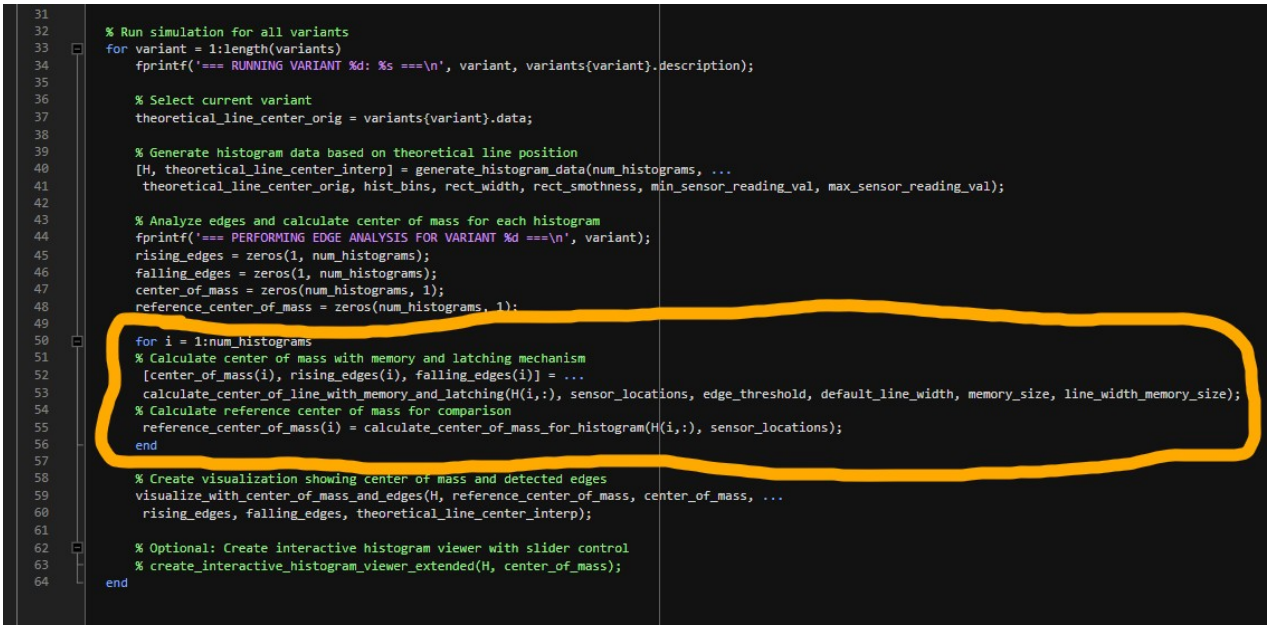


Rysunek 17 Wyniki symulacji dla kilku wariantów testowych

Na powyższym obrazku widzimy jak sprawdza się algorytm, symulacja dzięki MATLAB Online jest w pełni przenośnia, a po zapisaniu na MATLAB Drive możemy ją modyfikować w dowolnym momencie.

Parametry jakie są w niej ustawione domyślnie, pokrywają się z tymi obserwowanymi na fizycznym czujniku.

7.2 Dodawanie nowych algorytmów wykrywania linii



```
% Run simulation for all variants
for variant = 1:length(variants)
    fprintf('=== RUNNING VARIANT %d: %s ===\n', variant, variants{variant}.description);

    % Select current variant
    theoretical_line_center_orig = variants{variant}.data;

    % Generate histogram data based on theoretical line position
    [H, theoretical_line_center_interp] = generate_histogram_data(num_histograms, ...
     theoretical_line_center_orig, hist_bins, rect_width, rect_smothness, min_sensor_reading_val, max_sensor_reading_val);

    % Analyze edges and calculate center of mass for each histogram
    fprintf('=== PERFORMING EDGE ANALYSIS FOR VARIANT %d ===\n', variant);
    rising_edges = zeros(1, num_histograms);
    falling_edges = zeros(1, num_histograms);
    center_of_mass = zeros(num_histograms, 1);
    reference_center_of_mass = zeros(num_histograms, 1);

    for i = 1:num_histograms
    % Calculate center of mass with memory and latching mechanism
     [center_of_mass(i), rising_edges(i), falling_edges(i)] = ...
     calculate_center_of_line_with_memory_and_latching(H(i,:), sensor_locations, edge_threshold, default_line_width, memory_size, line_width_memory_size);
    % Calculate reference center of mass for comparison
     reference_center_of_mass(i) = calculate_center_of_mass_for_histogram(H(i,:), sensor_locations);
    end

    % Create visualization showing center of mass and detected edges
    visualize_with_center_of_mass_and_edges(H, reference_center_of_mass, center_of_mass, ...
     rising_edges, falling_edges, theoretical_line_center_interp);

    % Optional: Create interactive histogram viewer with slider control
    % create_interactive_histogram_viewer_extended(H, center_of_mass);
end
```

Rysunek 18 Lokalizacja fragmentu odpowiedzialnego za testowanie algorytmu

Aby przetestować inny algorytm, należy podmienić wywołanie funkcji:

calculate_center_of_line_with_memory_and_latching(…)

Należy pamiętać że do poprawnego wyświetlania potrzebne jest minimum wyliczenie środka linii, krawędzie linii są tylko w celu analizy na wykresie.

Najbardziej wygodnym rozwiązaniem może być jednak modyfikacja samej implementacji tej funkcji.

8. Podsumowanie

W ramach realizacji tego projektu skupiono się na udoskonaleniu jednego z kluczowych komponentów Line-Followera – algorytmu analizującego linię. Jego poprawne działanie jest niezbędne do prawidłowego poruszania się robota po wyznaczonej trasie. Dotychczasowe, najprostsze podejścia okazały się niewystarczające i nieskuteczne. Problem ten wymagał bardziej zaawansowanego i nieszablonowego podejścia. Zastosowanie środowiska MATLAB, doskonale przystosowanego do tego typu analiz, pozwoliło na opracowanie rozwiązania, które będzie służyło również do weryfikacji zbieżności zachowania symulowanego robota z jego rzeczywistym działaniem w przyszłości. Jest to kamień milowy w rozwijaniu projektów, który jest i będzie udoskonalany.

Niestety w trakcie tworzenia uszkodzeniu uległ czujnik linii, a przez brak części nie udało się zaimplementować zaprojektowanego algorytmu z Matlaba na C (uszkodzeniu uległ procesor STM32L051)

Spis ilustracji

Bibliografia

[1] J. Stelmach, Dydaktyczna platforma pojazdu autonomicznego typu „Line Follower” oparta o płytkę rozwojową NUCLEO-STM32G491RE, kraków, 2025.

[2] „MATLAB Online,” Mathworks, [Online]. Available: https://matlab.mathworks.com/. [Data uzyskania dostępu: 15 06 2025].