POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka (AIR)
SPECJALNOŚĆ: Robotyka (ARR)

PROJEKT INŻYNIERSKI

Metody reprezentacji i śledzenia trasy
dla robota klasy linefollower

Scene representation and line tracking methods
for linefollower robots

AUTOR:
Tomasz Januszewski

PROWADZĄCY PRACĘ:
dr inż. Robert Muszyński, K-7

OCENA PRACY:

WROCŁAW 2017

Kochanym Rodzicom

Spis treści

Rozdział 1

Wstęp

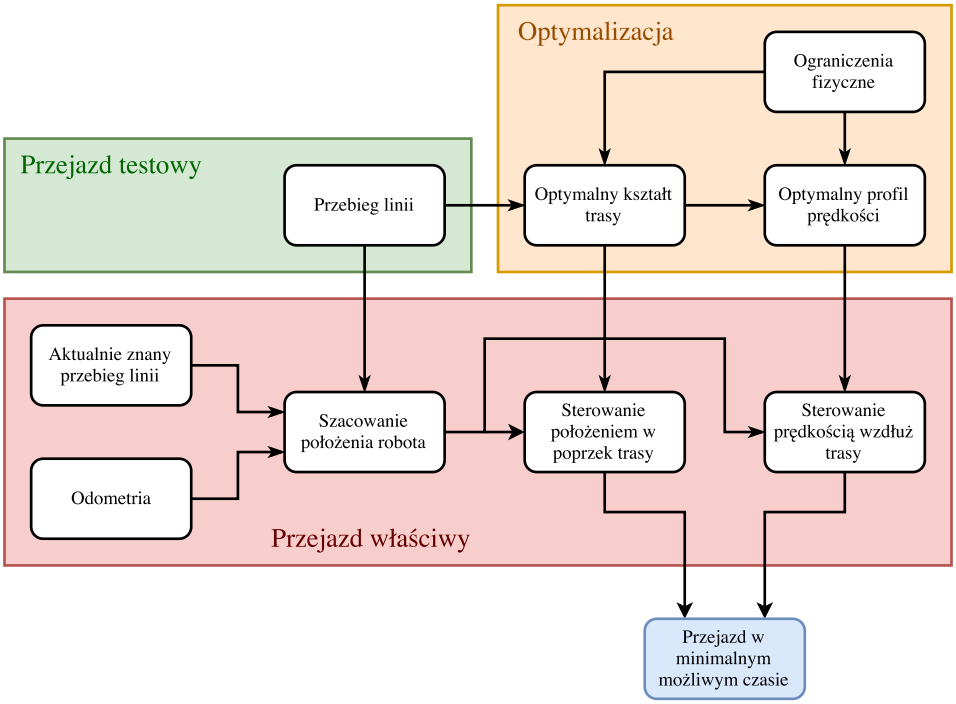
Kategoria robotów typu linefollower jest jedną z najpopularniejszych konkurencji rozgrywanych na zawodach amatorskich robotów. W międzynarodowym konkursie RobotChallenge 2016 na 645 konstrukcji zarejestrowanych w 13 kategoriach, aż 190 z nich należało do kategorii linefollower [4]. Konkurencja ta polega na przejechaniu przez autonomicznego robota mobilnego trasy wyznaczonej czarną linią w możliwie najkrótszym czasie [16].

Typowy robot klasy linefollower jest robotem mobilnym klasy $(2, 0)$ wyposażonym w zestaw czujników pozwalających na wykrywanie czarnej linii. Wśród startujących konstrukcji daje się zauważyć dążenie do poprawy ich osiągów poprzez optymalizację rozkładu masy czy też poprawę przyczepności kół. Popularnym zabiegiem jest również wyposażanie robotów w tzw. turbiny, które poprzez zasysanie powietrza spod robota powodują znaczne zwiększenie siły nacisku na podłoże bez zmiany jego masy, co w efekcie pozwala na uzyskanie bardzo dużych przyspieszeń. Zyski z tego płynące bywają na tyle duże, że w przypadku niektórych zawodów [5, 3] roboty z turbiną i bez niej startują w oddzielnych kategoriach.

Równocześnie wydaje się, że w znakomitej większości konstrukcji przy wyznaczaniu sterowania wykorzystywana jest jedynie informacja o trasie będącej w zasięgu czujników robota, bez rozpatrywania problemu globalnie. Takie postępowanie wymusza bardzo zachowawczą jazdę i skutecznie ogranicza możliwości rozwijania dużych prędkości na trasie. Występowanie na trasie ostrych zakrętów powoduje, że w każdym jej punkcie robot musi mieć możliwość gwałtownego wyhamowania na odcinku rzędu kilkunastu centymetrów, determinowanym długością robota. Regulamin konkurencji pozwala zwykle na kilka przejazdów [16], więc potencjalnie możliwe byłoby zapamiętanie kształtu trasy podczas pierwszego przejazdu z małą prędkością i wykorzystanie tej wiedzy w następnych próbach. Dawałoby to możliwość rozwijania dużych prędkości na odcinkach prostych, pozwalając jednocześnie na rozpoczęcie hamowania przed zakrętami, zanim jeszcze te pojawią się w zasięgu czujników robota.

Celem pracy jest zaproponowanie i próba przetestowania możliwie kompletnego rozwiązania, pozwalającego na przejazd robota typu linefollower po trasie w najkrótszym osiągalnym czasie, z wykorzystaniem uprzednio zdobytej wiedzy o parametrach tej trasy. Zadanie podzielono na mniejsze podproblemy, a każdy z nich sprowadzono do problemu pochodnego i już rozwiązanego. Wśród nich należy wymienić:

- przegląd użytecznych dla analizowanego zadania metod reprezentacji trasy,
- przedstawienie sposobu wykrywania linii i możliwości zwiększenia dokładności określenia jej położenia,
- optymalizację przebiegu trasy robota korzystającą z faktu, że ma on niezerowe wymiary i przejazd nie musi odbywać się dokładnie jej środkiem,



Rysunek 1.1 Idea rozwiązania

- dobór profilu prędkości pozwalającego na możliwie maksymalne wykorzystanie ograniczeń fizycznych, takich jak tarcie,
- zaproponowanie metody śledzenia położenia robota na trasie, odpornej na niedokładności pomiarów i zdarzenia losowe, wynikające na przykład z poślizgów.

Sposób, w jaki wymienione zagadnienia mogą zostać wykorzystane w rozwiązaniu zadania minimalizacji czasu przejazdu zilustrowano na rysunku 1.1. Przeprowadzone w pracy rozważania i przedstawione rezultaty po części dotyczą przypadku robota klasy $(2, 0)$ z zestawem czujników zainstalowanych poza jego osią (jak opisano w dodatku A).

Układ pracy jest następujący. W rozdziale 2 opisano użyte metody reprezentacji trasy. Rozdział 3 pokazuje sposoby wykrywania linii i określania położenia robota na trasie. Rozdział 4 przedstawia technikę wyznaczenia optymalnego profilu prędkości dla zadanej trasy, przybliża algorytm optymalizacji samego kształtu trasy oraz obrazuje potencjalne zyski wynikające z ich zastosowania. Rozdział 5 podsumowuje całość. Umieszczony w pracy dodatek A zawiera opis robota użytego podczas wykonanych eksperymentów. Dodatek B jest uzupełnieniem podrozdziału 4.3 – znajdują się w nim wyniki symulacji przeprowadzonych na większym zbiorze tras testowych.

Rozdział 2

Reprezentacja trasy

Sposób reprezentacji trasy powinien pozwalać na realizację stawianych przed robotem zadań i weryfikację poprawności otrzymywanych wyników, uwzględniając jednocześnie możliwości obliczeniowe i pamięciowe robota. Tak postawione założenie uniemożliwia niestety zastosowanie tylko jednego sposobu reprezentacji, przez co podczas realizacji niniejszej pracy użyteczne okazały się następujące formaty opisu trasy:

- w postaci uszeregowanego zbioru punktów (x, y),
- przez określenie krzywizny w funkcji odległości $\kappa(s)$.

Reprezentacje te są sobie równoważne z dokładnością do warunków początkowych, tzn. istnieją transformacje przekształcające jedną postać w drugą. Użycie konkretnej z nich w rozwiązaniu danego podproblemu wynika wprost z wymaganego formatu danych wejściowych zastosowanego algorytmu.

Przy rozpatrywaniu analizowanych zadań często należy rozróżnić kształt linii po której podąża robot od kształtu trasy, którą pokonuje jego korpus (rysunek 2.1). Pierwszy z nich jest niezależny od sposobu ruchu robota i podczas przejazdu posłuży do określania jego położenia na trasie. Drugi natomiast wynika z kształtu trasy, konstrukcji robota oraz sposobu jego ruchu i będzie poddawany optymalizacji, a otrzymany kształt pozwoli jednocześnie wyznaczyć optymalną prędkość robota wzdłuż trasy.

W przypadku robota klasy $(2, 0)$, wyposażonego w czujniki przyrostowe, takie jak na przykład enkodery, wielkościami mierzonymi i charakteryzującymi ruch robota w danej chwili są zmiany odległości pokonywanych przez lewe oraz prawe koło. Oznaczając je odpowiednio jako p_l oraz p_r, możliwe jest opisanie ruchu robota poprzez rotację r oraz translację t

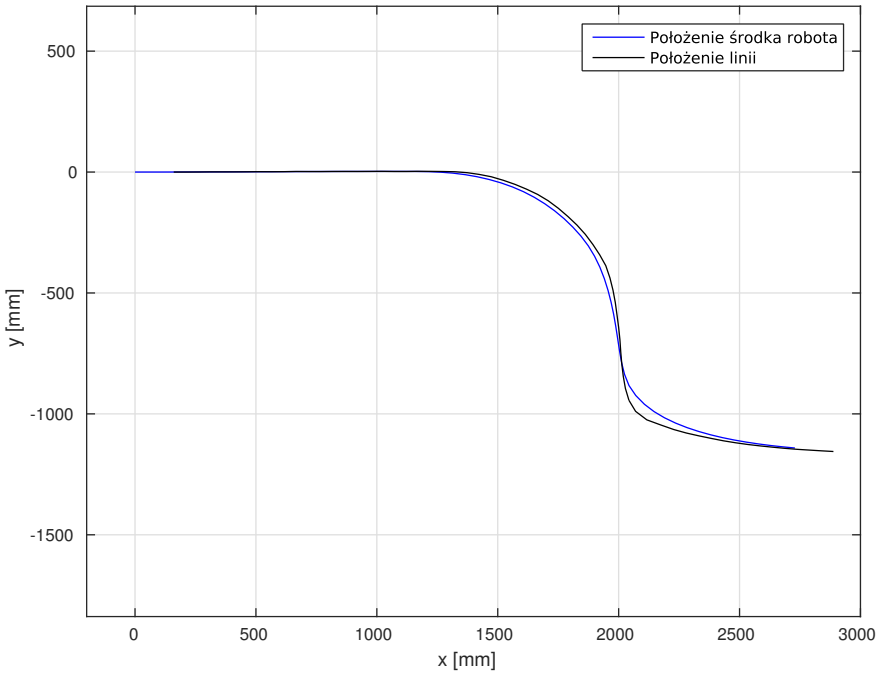
$$t = \frac{p_r + p_l}{2}, \tag{2.1}$$

$$r = \frac{p_r - p_l}{d}, \tag{2.2}$$

gdzie d jest odległością pomiędzy kołami robota. Obliczone wartości r i t posłużą do wyznaczenia kolejnych położeń robota, natomiast jednoczesny pomiar położenia linii względem robota pozwoli na wyznaczenie przebiegu całej trasy.

2.1 Uszeregowany zbiór punktów (x, y)

Reprezentacja trasy w postaci zbioru punktów (x, y) pozwala przede wszystkim na graficzne przedstawienie jej przebiegu, chociażby w celu weryfikacji poprawności pomiarów



Rysunek 2.1 Rozróżnienie kształtu linii i kształtu trasy pokonywanej przez korpus robota

dokonywanych przez robota. Taki format danych jest również używany przez algorytm optymalizujący kształt ścieżki, opisany w podrozdziale 4.2. Przykładową reprezentację trasy w omawianej postaci przedstawia rysunek 2.2.

Obliczenie aktualnego położenia robota $(x_{r,i}, y_{r,i})$ na podstawie translacji t_i i rotacji r_i w danej chwili wymaga wyboru pewnego punktu początkowego, którym może być na przykład

$$(\alpha_0, x_0, y_0) = (0, 0, 0), \tag{2.3}$$

gdzie α_0 oznacza początkową orientację robota. W kolejnych krokach otrzymujemy

$$\alpha_i = \alpha_{i-1} + r_i, \tag{2.4}$$

$$x_{r,i} = x_{r,i-1} + t_i \cos \alpha_i, \tag{2.5}$$

$$y_{r,i} = y_{r,i-1} + t_i \sin \alpha_i. \tag{2.6}$$

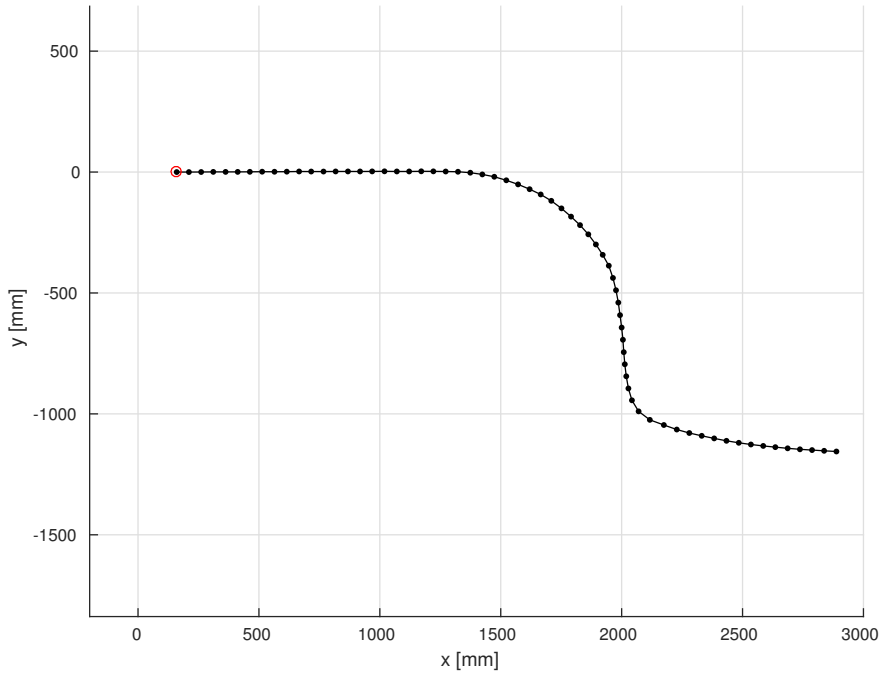
Wyznaczenie aktualnego położenia linii $(x_{l,i}, y_{l,i})$ bazuje na obliczonym wcześniej położeniu robota $(x_{r,i}, y_{r,i})$ i wynosi

$$x_{l,i} = x_{r,i} + l \cos(\alpha_i + \beta_i), \tag{2.7}$$

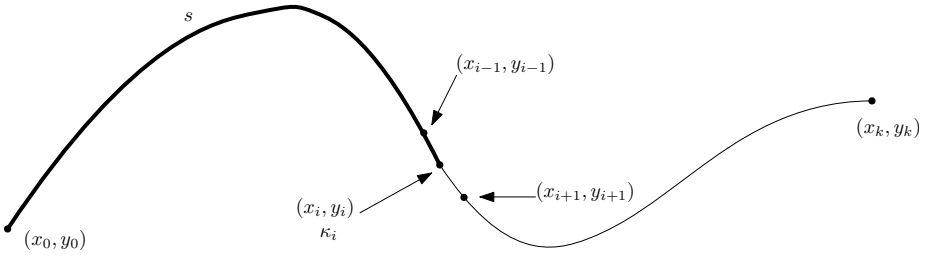
$$y_{l,i} = y_{r,i} + l \sin(\alpha_i + \beta_i), \tag{2.8}$$

gdzie β_i oznacza aktualny kąt położenia linii wyznaczony przez czujniki, natomiast l jest odległością między osią obrotu robota[1] a czujnikami.

[1]Dla robota klasy $(2, 0)$ osią obrotu będziemy nazywać punkt leżący w połowie odcinka łączącego środki kół.



Rysunek 2.2 Reprezentacja przykładowej trasy w formie zbioru punktów (x, y) z zaznaczonym punktem początkowym



Rysunek 2.3 Wizualizacja pojęcia krzywizny krzywej w funkcji przejechanej odległości

2.2 Krzywizna krzywej w funkcji długości $\kappa(s)$

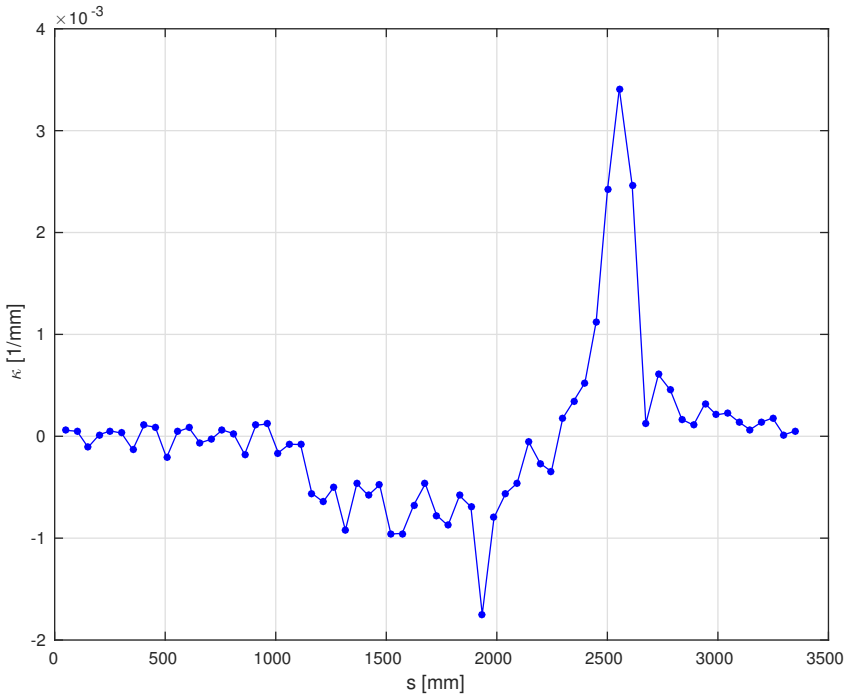
Reprezentacja trasy w postaci jej krzywizny w funkcji odległości $\kappa(s)$ wykorzystywana jest przy obliczenia optymalnego profilu prędkości (podrozdział 4.1) oraz do szacowania położenia robota na trasie podczas finalnego przejazdu (podrozdział 3.2). Wizualizację pojęcia krzywizny w funkcji przejechanej odległości przedstawia rysunek 2.3, natomiast przykładowy przebieg $\kappa(s)$ pokazuje rysunek 2.4.

Obliczenie krzywizny trasy pokonywanej przez robota $(\kappa_{r,i}, s_{r,i})$ możliwe jest wprost na podstawie znajomości chwilowej translacji t_i i rotacji r_i

$$\kappa_{r,i} = \frac{r_i}{t_i}, \tag{2.9}$$

$$s_{r,i} = s_{r,i-1} + t_i, \tag{2.10}$$

przy czym $s_{r,0} = 0$. Obliczenie krzywizny samej linii $(\kappa_{l,i}, s_{l,i})$ dokonywane jest pośrednio,



Rysunek 2.4 Reprezentacja trasy 2.2 w postaci $\kappa(s)$

dzięki znajomości położeń linii w postaci $(x_{l,i}, y_{l,i})$ i transformacji opisanej w podrozdziale 2.3.

2.3 Transformacje pomiędzy reprezentacjami trasy

Przekształcenia zbioru punktów (x, y) w reprezentację $\kappa(s)$ dokonuje się obliczając zmiany pokonanej odległości ds i kąta $d\alpha$ wzdłuż trasy

$$dx_i = x_i - x_{i-1}, \tag{2.11}$$

$$dy_i = y_i - y_{i-1}, \tag{2.12}$$

$$ds_i = \sqrt{dx_i^2 + dy_i^2}, \tag{2.13}$$

$$\alpha_i = \arctan \frac{dy_i}{dx_i}, \tag{2.14}$$

$$d\alpha_i = \alpha_i - \alpha_{i-1}. \tag{2.15}$$

Wartość obliczona z (2.14) powinna uwzględniać znaki dx_i oraz dy_i[2]. Wynik równania (2.15) powinien mieścić się w przedziale $d\alpha \in [-\pi, \pi)$. Pozwala to na obliczenie krzywizny (rysunek 2.5) wprost z definicji

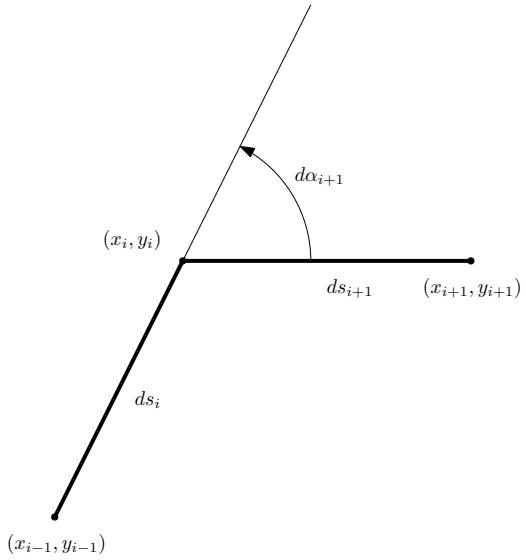
$$s_i = s_{i-1} + ds_i, \tag{2.16}$$

$$\kappa_i = \frac{d\alpha_i}{ds_i}, \tag{2.17}$$

przy czym należy założyć

$$(s_0, \kappa_0) = (0, 0). \tag{2.18}$$

[2]W językach programowania oznacza to zwykle użycie funkcji *atan2*.



Rysunek 2.5 Obliczanie krzywizny krzywej na podstawie współrzędnych punktów sąsiadujących

Transformacja odwrotna z reprezentacji $\kappa(s)$ do zbioru punktów (x, y) oznacza obliczanie kolejnych położeń (x_i, y_i) na podstawie znanych s_i oraz k_i, tzn.

$$ds_i = s_i - s_{i-1}, \tag{2.19}$$

$$d\alpha_i = k_i ds_i, \tag{2.20}$$

$$\alpha_i = \alpha_{i-1} + d\alpha_i, \tag{2.21}$$

$$x_i = x_{i-1} + ds_i \cos\alpha_i, \tag{2.22}$$

$$y_i = y_{i-1} + ds_i \sin\alpha_i. \tag{2.23}$$

Tu także należy założyć pewne położenie początkowe, na przykład

$$(\alpha_0, x_0, y_0) = (0, 0, 0). \tag{2.24}$$

Rozdział 3

Szacowanie położenia robota

Zadania postawione przed robotem implikują konieczność szacowania położenia robota i linii w możliwie dokładny sposób. Precyzyjne zapamiętanie kształtu linii pozwoli na staranne zaplanowanie ruchu robota, a dokładne oszacowanie położenia robota na trasie podczas przejazdu finalnego da możliwość właściwego wysterowania. Wymaga to oczywiście odpowiedniej obróbki i analizy danych sensorycznych.

3.1 Szacowanie położenia linii względem robota

Na potrzeby pracy w celu skupienia się na istocie problemu założono, że trasa jest jednorodna, tzn. jest ciągła, nie zawiera ostrych zakrętów, skrzyżowań, rozgałęzień lub przeszkód. Wprawdzie wykrycie takich sytuacji nie jest szczególnie skomplikowane i powinno być dokonane jeszcze przed jakimikolwiek próbami określenia przebiegu linii, jednakże zagadnienie to wykracza poza zakres niniejszej pracy.

3.1.1 Normalizacja odczytów z czujników linii

Czujniki zbliżeniowe [15] stosowane w typowych robotach klasy linefollower w roli czujników linii są używane przede wszystkim w elektronice konsumenckiej do binarnego wykrywania obiektów i w tej roli spisują się doskonale. Jednakże przy odczycie z nich wielkości analogowej, co ma miejsce w opisywanym zastosowaniu, wykazują duży rozrzut pomiarów pomiędzy poszczególnymi egzemplarzami (rysunek 3.1).

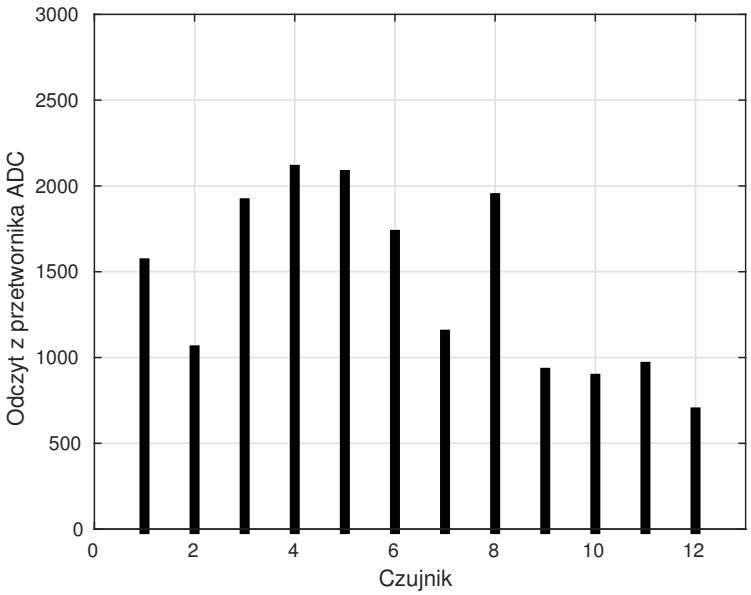
W celu korekty obserwowanego efektu, dla każdego czujnika zdefiniowano parametr czułości λ_i jako

$$\lambda_i = \frac{1}{w_i}, \quad i = 1..n, \tag{3.1}$$

gdzie w_i jest odczytem z i-tego czujnika, uzyskanym po ustawieniu robota na białym podłożu, a n jest liczbą czujników. Parametry λ_i zostają zmierzone raz i są przechowywane w pamięci programu jako stałe charakterystyczne dla każdego czujnika. Obliczona wartość skorygowana odczytu c_i wynosi więc

$$c_i = \lambda_i y_i, \quad i = 1..n, \tag{3.2}$$

gdzie y_i jest aktualnym odczytem z i-tego czujnika, a λ_i jego czułością charakterystyczną (3.1). Dzięki takiemu postępowaniu wartości c_i są praktycznie równe 1 w przypadku, gdy dany czujnik znajduje się nad białą powierzchnią (rysunek 3.2), oraz bliskie 0, gdy sensor znajduje się nad czarną linią (rysunek 3.3). Kontrola wartości zwracanych przez czujniki



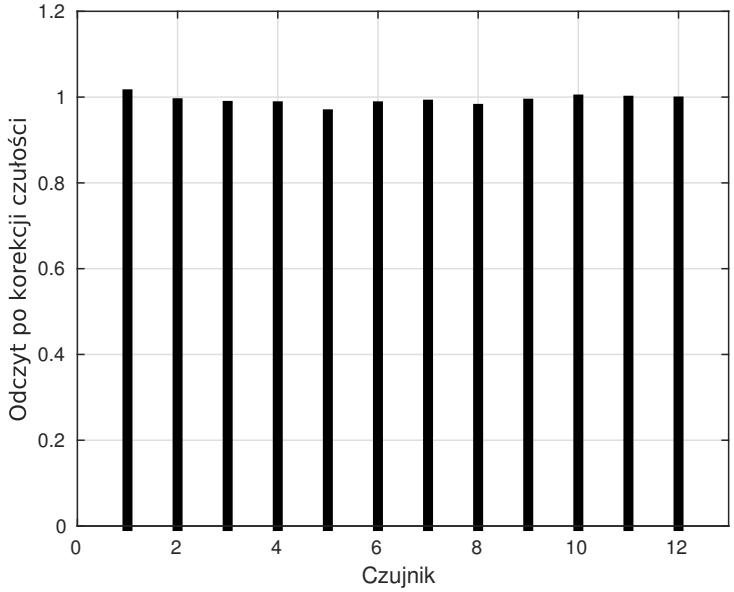
Rysunek 3.1 Przykładowe wartości odczytane z czujników linii – robot umieszczony na białym podłożu

po umieszczeniu robota na czarnym podłożu (rysunek 3.4) pozwala stwierdzić, że w tym przypadku odczyty nie wykazują znacznych rozbieżności.

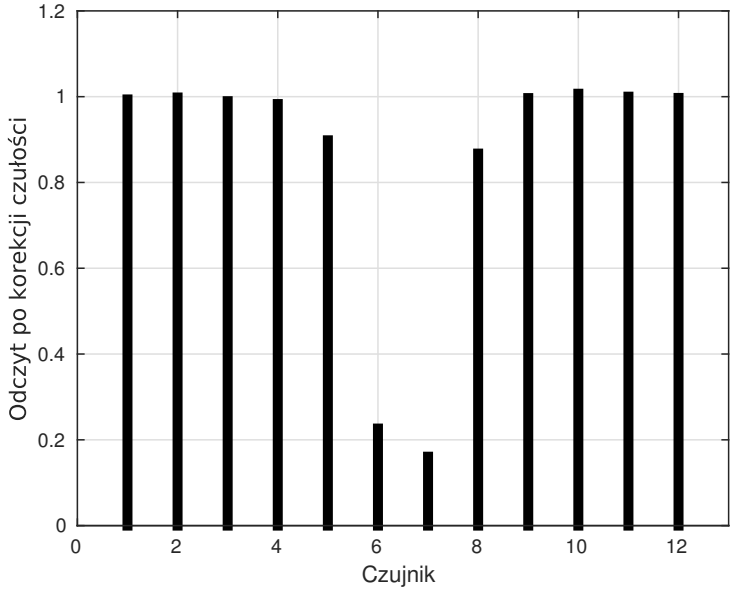
3.1.2 Estymacja dokładnego położenia linii

Proste algorytmy wykrywania linii [11, 20, 7] realizują progowanie (sprzętowe lub programowe) wartości analogowej odczytanej z czujnika, co pozwala na rozróżnienie dwóch stanów – obecności linii pod sensorem lub jej braku. W tym wypadku położenie linii wyznaczają wprost współrzędne czujnika, dla którego odczytana wartość jest najmniejsza. Postępowanie takie nazwano metodą najmniejszego odczytu. Niestety nie jest ono wystarczające do dokładnego oszacowania położenia linii, jako że błąd jej wyznaczenia jest na poziomie połowy odległości między czujnikami. Oprócz tego "schodkowy" charakter zmian wyznaczonego położenia linii powoduje trudności w uzyskaniu sytuacji, gdy robot porusza się po linii w sposób gładki. Stąd, do wyznaczenia położenia linii zaproponowano w tym wypadku zastosowanie odmiennej metody, nazwanej metodą wierzchołka paraboli, w której to określa się zgrubne położenie w sposób identyczny jak w przypadku metody najmniejszego odczytu, a następnie dokonuje się korekcji położenia na podstawie wartości z czujnika o najmniejszym odczycie i jego bezpośrednich sąsiadów [14]. Dane te traktowane są jako wartości pewnej funkcji w punktach -1, 0 i 1. Następnie wyznaczana jest przechodząca przez nie parabola i obliczane jest położenie jej wierzchołka (rysunek 3.5). Oznaczając odczyt czujnika referencyjnego jako β, a sensorów odpowiednio po lewej i prawej stronie jako α i γ, zachodzi

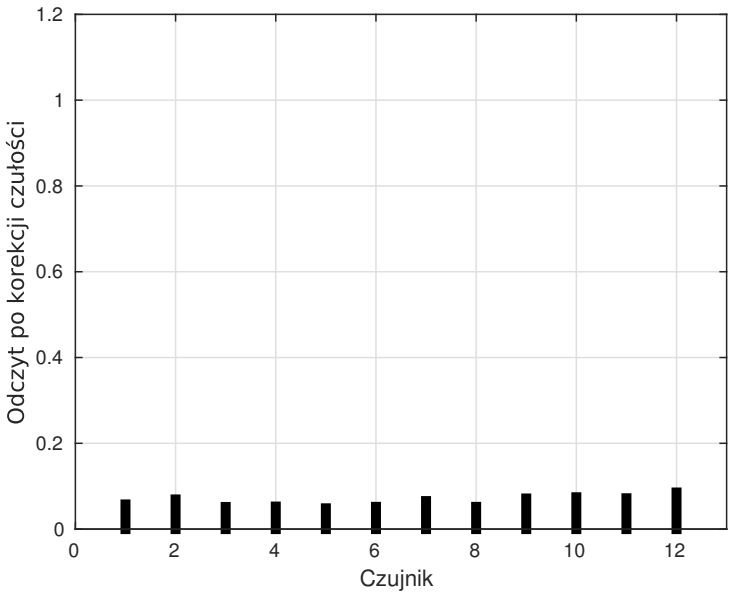
$$\begin{cases} y(-1) = \alpha \\ y(0) = \beta \\ y(1) = \gamma \end{cases}, \tag{3.3}$$



Rysunek 3.2 Odczyt czujników linii uwzględniający czułość – robot umieszczony na białym podłożu



Rysunek 3.3 Odczyt czujników linii uwzględniający czułość – robot umieszczony nad czarną linią nakreśloną na białej powierzchni



Rysunek 3.4 Odczyt czujników linii uwzględniający czułość – robot umieszczony na czarnym podłożu

przez co współrzędna x położenia wierzchołka paraboli przyjmuje wartość

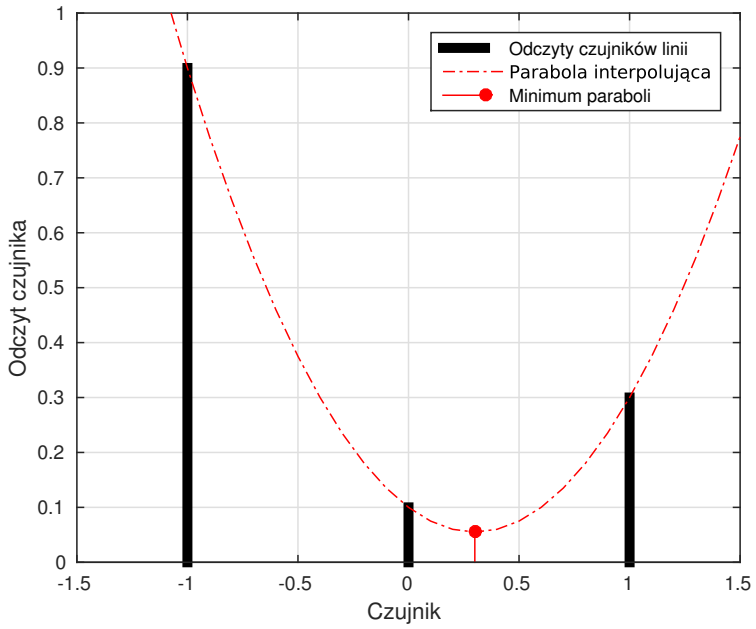
$$p = \frac{1}{2}\frac{\alpha - \gamma}{\alpha - 2\beta + \gamma} \tag{3.4}$$

i zawiera się w przedziale $(-0.5, 0.5)$. Wartość ta przemnożona przez odległość między czujnikami wyznacza przesunięcie, o jakie należy skorygować położenie linii wyznaczone metodą najmniejszego odczytu.

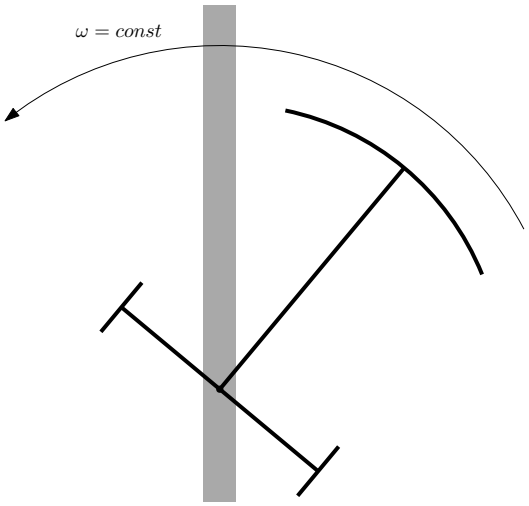
W celu porównania metod wyznaczania położenia linii przeprowadzono eksperyment, w którym robot porusza się ze stałą prędkością kątową ($\omega = 90\frac{^\circ}{s}$) nad fragmentem trasy, a oś obrotu jest umieszczona na jej środku (rysunek 3.6). Czujniki linii rozmieszczone są na łuku co 3.4°, a dokładny ich układ jest przedstawiony w dodatku A. Położenie linii wyznaczane w trakcie eksperymentu metodą najmniejszego odczytu oraz metodą wierzchołka paraboli w funkcji czasu pokazano na rysunku 3.7. Charakterystyki te porównano z linią prostą aproksymującą idealny odczyt położenia linii i graficznie oszacowano maksymalne błędy wprowadzane przez testowane algorytmy (rysunek 3.8). W przypadku metody najmniejszego odczytu wartość błędu jest na poziomie 2°, tj. wynosi połowę odległości kątowej między czujnikami. Zastosowanie metody wierzchołka paraboli daje maksymalną odchyłkę na poziomie 0.2°, co oznacza 10-krotne zwiększenie dokładności wyznaczenia położenia linii. Dodatkowo charakterystyka ta jest gładka, co ułatwia płynne śledzenie trasy.

3.1.3 Wykrywanie obecności linii

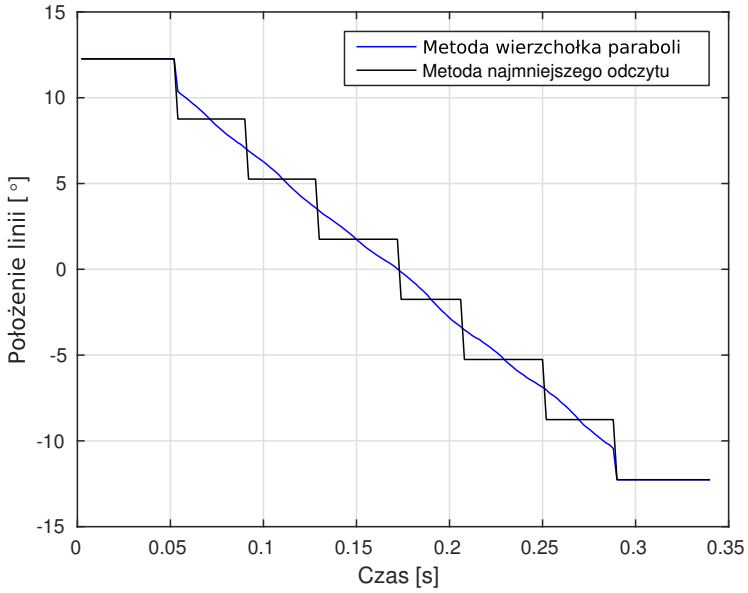
Aby szacowanie położenia linii miało jakikolwiek sens, należy najpierw możliwie wiarygodnie określić, czy ta w ogóle znajduje się pod czujnikami. Próba naiwnego obliczenia położenia linii, gdy tej nie ma pod robotem, kończy się uzyskaniem wyniku, który jest oczywiście poprawny matematycznie, ale nie odzwierciedla rzeczywistej sytuacji. Obliczane położenia linii w takim przypadku są zwykle chaotyczne i szybkozmienne, a próba



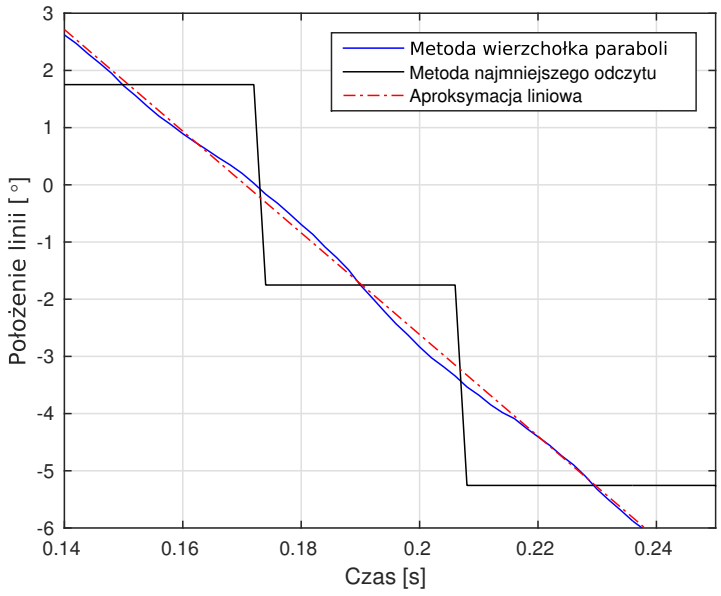
Rysunek 3.5 Algorytm korekcji – wyznaczenie położenia wierzchołka paraboli przechodzącej przez 3 punkty



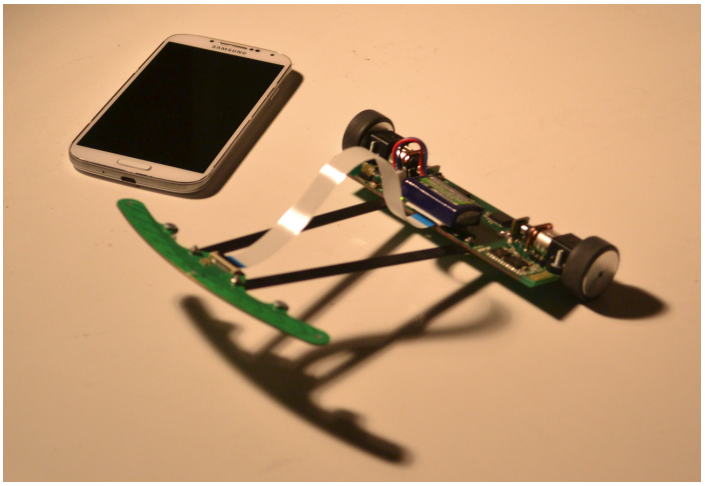
Rysunek 3.6 Wizualizacja przebiegu eksperymentu służącego do porównania metod wyznaczania położenia linii



Rysunek 3.7 Porównanie działania algorytmów obliczających położenie linii



Rysunek 3.8 Aproksymacja wyznaczonej charakterystyki linią prostą w celu oszacowania maksymalnego błędu pomiaru kąta



Rysunek 3.9 Oderwanie się frontu robota od powierzchni spowodowane nadmiernym przyspieszeniem, telefon w tle dla zobrazowania orientacji podłoża

wysterowania robota na ich podstawie może doprowadzić w skrajnym przypadku do spalenia silników.

Zasadniczo zakładamy, że robot znajduje się nad linią wyznaczającą tor przejazdu, jednakże sytuacją dość oczywistą, która może wystąpić, jest wypadnięcie robota z trasy wskutek, na przykład, nadmiernej prędkości na zakręcie. Wówczas odczyt z czujników linii następuje w momencie, gdy te znajdują się nad białą powierzchnią bez linii. Przypadkiem mniej oczywistym jest możliwość dokonania odczytu w momencie, gdy płytka z czujnikami znajduje się zbyt wysoko nad powierzchnią. Może to nastąpić na przykład przy gwałtownym przyspieszaniu (rysunek 3.9) lub podczas najechania na nierówność przy dostatecznie dużej prędkości. Wykrycie obecności linii bywa pomocne również przy wykrywaniu początku i końca trasy (jeśli ta nie jest krzywą zamkniętą). Pamiętając ostatnie wiarygodne położenie linii można rozróżnić przypadki najechania na koniec trasy i wypadnięcia z niej.

W celu określenia sposobu wykrywania obecności linii, zaproponowano wyznaczenie następujących parametrów:

- wartość średnia odczytu z czujników μ,
- wartość minimalna odczytu z czujników y_{min},
- odchylenie standardowe odczytu z czujników σ,
- stosunek wartości minimalnej do średniej $\xi = \frac{y_{min}}{\mu}$.

Wyżej wymienione parametry zmierzono w czterech przypadkach:

- płytka czujników umieszczona na białym podłożu bez linii,
- płytka czujników umieszczona na białym podłożu nad linią, prosto,
- płytka czujników umieszczona na białym podłożu nad linią, pod kątem,
- płytka czujników umieszczona w dużej wysokości nad podłożem.

Uzyskane wyniki zebrano w tabeli 3.1. Można zaobserwować, że parametr σ daje wyraźne rozróżnienie między sytuacjami wykrycia linii, a innymi. Progowanie wartości ξ również pozwala rozgraniczyć te przypadki. Wysoka wartość y_{min} pozwala na wykrycie braku linii.

	μ	y_{min}	σ	ξ
Brak linii	0.971	0.961	0.007	0.989
Obecność linii	0.778	0.139	0.342	0.178
Obecność linii pod kątem	0.419	0.170	0.257	0.405
Czujniki w powietrzu	0.032	0.020	0.013	0.625

Tabela 3.1 Wartości obserwowanych parametrów uzyskane w różnych warunkach

Niska wartość μ oznacza, że moduł czujników jest w powietrzu. na potrzeby pracy przyjęto warunek wykrycia linii jako

$$\begin{cases} \mu > 0.2 \\ y_{min} < 0.5 \\ \sigma > 0.12 \\ \xi < 0.5 \end{cases} . \tag{3.5}$$

3.2 Szacowanie bezwzględnego położenia robota na trasie

Próby uzależnienia zachowania robota od jego położenia na trasie wymagają dokładnej znajomości tego położenia. Wykorzystanie w tym celu tylko i wyłącznie czujników przyrostowych w sposób przedstawiony na początku rozdziału 2, jest obarczone błędem (pomiaru odległości i kąta), który kumuluje się wraz z przejechaną odległością. Przykład takiego zjawiska przedstawia rysunek 3.10. Dodatkowo, jakiekolwiek poślizgi mogą powodować przekłamania na tyle duże, że przy ich obecności obliczone położenie robota kompletnie nie odzwierciedla stanu faktycznego.

Zaproponowano, aby przy próbie oszacowania aktualnego położenia uwzględniać tylko pomiary najbardziej aktualne, a więc obarczone małym błędem, charakteryzujące ostatnio przejechany odcinek trasy. Teraz, znając przebieg całej trasy w reprezentacji $\kappa(s)$ (zapamiętany podczas przejazdu testowego), należy znaleźć taki jej fragment, który w możliwie największym stopniu przypomina ten odcinek. Metoda taka nazywana jest dopasowaniem do wzorca (ang. *template matching, pattern matching*) i jest szeroko używana w przetwarzaniu obrazów i sygnałów [17, 12, 6, 18]. Polega ona na obliczeniu, dla dwóch sygnałów zwanych wzorcem i sygnałem referencyjnym, pewnej metryki [1] przy wszystkich możliwych przesunięciach między tymi sygnałami. Ekstremum tak wyznaczonej funkcji mówi o wartości przesunięcia między sygnałami, które daje najlepsze dopasowanie między nimi. do najpopularniejszych algorytmów realizujących to zadanie należą metody:

- CC *(ang. Cross correlation)* – korelacja krzyżowa

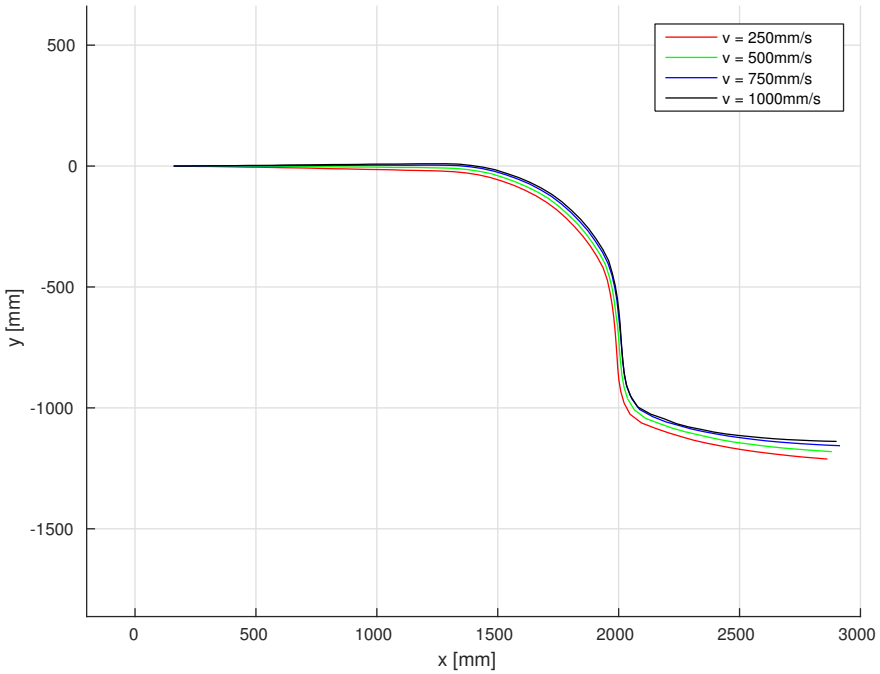
$$CC(\tau) = \sum_t a(t)b(t-\tau), \quad x_s = \arg\max_\tau CC(\tau), \tag{3.6}$$

- SSD *(ang. Sum of squared differences)* – suma kwadratów błędów

$$SSD(\tau) = \sum_t (a(t) - b(t-\tau))^2, \quad x_s = \arg\min_\tau SSD(\tau), \tag{3.7}$$

- SAD *(ang. Sum of absolute differences)* – suma modułów błędów

$$SAD(\tau) = \sum_t |a(t) - b(t-\tau)|, \quad x_s = \arg\min_\tau SAD(\tau), \tag{3.8}$$



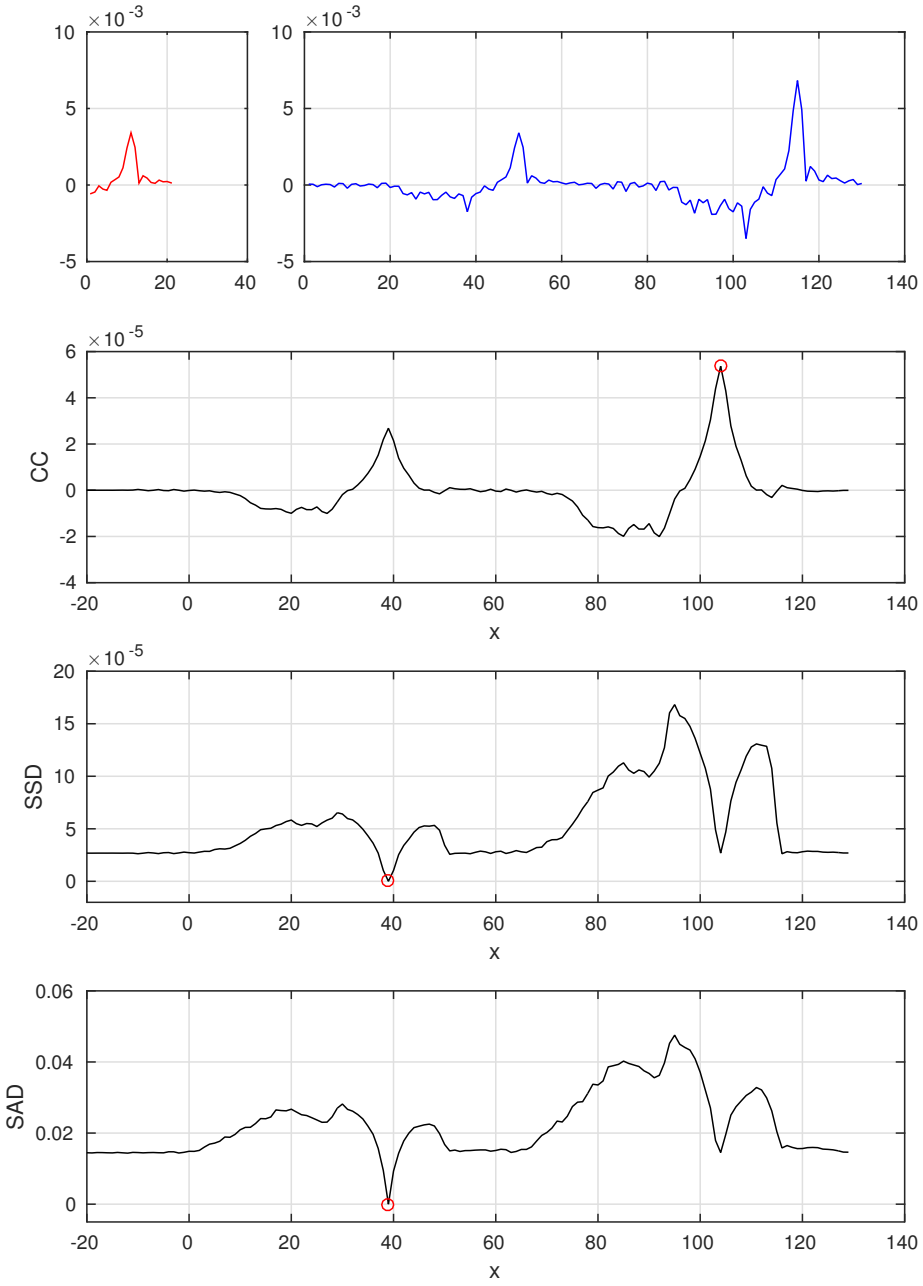
Rysunek 3.10 Położenie linii widziane przez robota w zależności od prędkości liniowej. Odczyty na podstawie samych enkoderów

gdzie a i b są odpowiednio sygnałem poddawanym analizie oraz poszukiwanym wzorcem, t jest domeną analizowanego sygnału (może to być na przykład czas lub długość), τ oznacza przesunięcie sygnału b względem a, natomiast x_s jest taką wartością przesunięcia, dla którego zachodzi najlepsze dopasowanie.

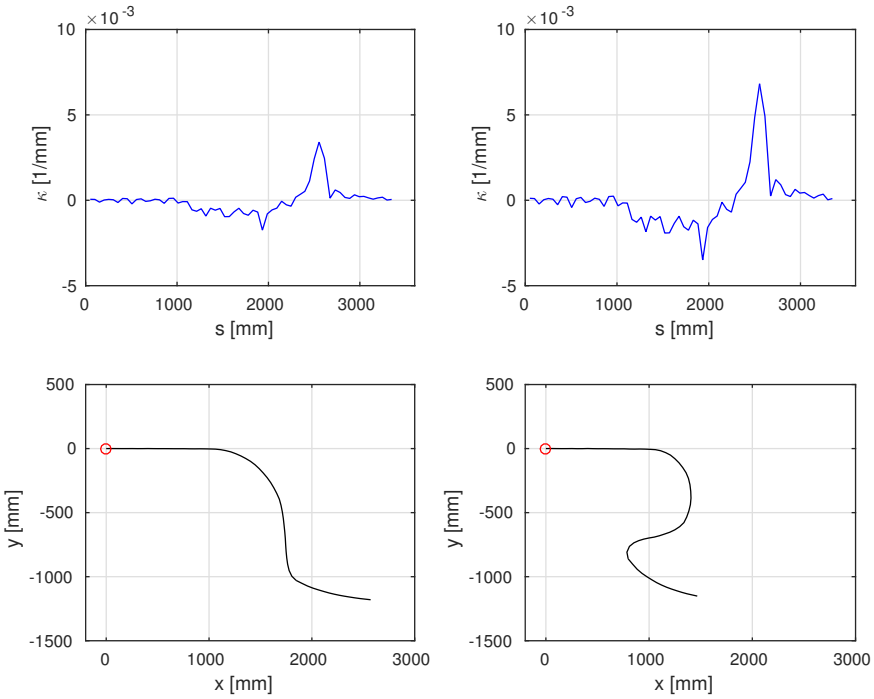
Dla ilustracji na rysunku 3.11 przedstawiono przebiegi uzyskane w efekcie zastosowania wymienionych metod dla przykładowego wzorca i sygnału referencyjnego. Rysunek ten jednocześnie pokazuje pewną, niepożądaną własność algorytmu korelacji krzyżowej, tzn. nieodporność na przeskalowanie sygnału. Wynika ona z faktu, iż algorytm korelacji wykorzystuje operację mnożenia, przez co obecność w sygnale referencyjnym fragmentu podobnego do wzorca, ale przeskalowanego, może wykazać podobieństwo właśnie w tym fragmencie. Poddając analizie sygnał $\kappa(s)$ należy jednak mieć na uwadze, że jego przeskalowanie skutkuje kompletnie inną reprezentacją (x, y) (rysunek 3.12), przez co fragmenty o różnych amplitudach nie powinny być określane jako podobne. Wydaje się więc, że w tym konkretnym przypadku metody SAD i SSD, korzystające z operacji odejmowania, dadzą bardziej wiarygodne wyniki w porównaniu do metody CC.

Istotną sprawą jest odpowiedni dobór długości wzorca. Rysunki 3.13 oraz 3.14 pokazują przebiegi funkcji SSD i SAD dla różnych długości sygnału wzorcowego. Daje się zauważyć, że dla obu funkcji zwiększanie długości wzorca powoduje zwiększenie wybitności szukanego ekstremum ponad jego otoczenie. Z drugiej jednak strony, jak zostało powiedziane wcześniej, sygnał wzorcowy jest obarczony błędem rosnącym wraz z jego długością i powinien zawierać tylko najbardziej aktualne pomiary. Dobór odpowiedniej długości wzorca jest więc kompromisem pomiędzy dokładnością wyznaczenia położenia, a odpornością na błędy, wynikające na przykład z zaszumienia sygnałów poddawanych analizie.

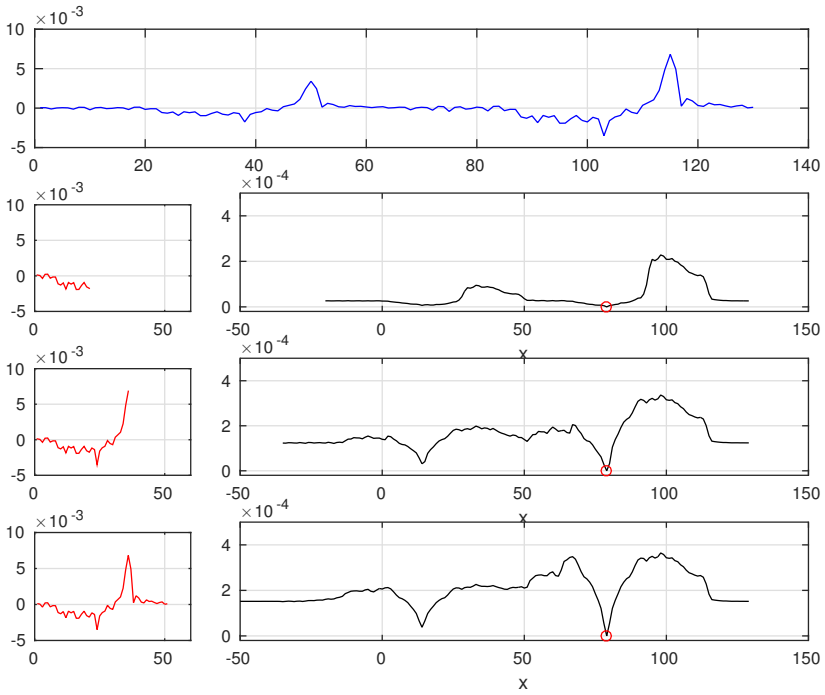
Kolejną rzeczą wartą zastosowania jest wykorzystanie wiedzy o ruchu robota w ce-



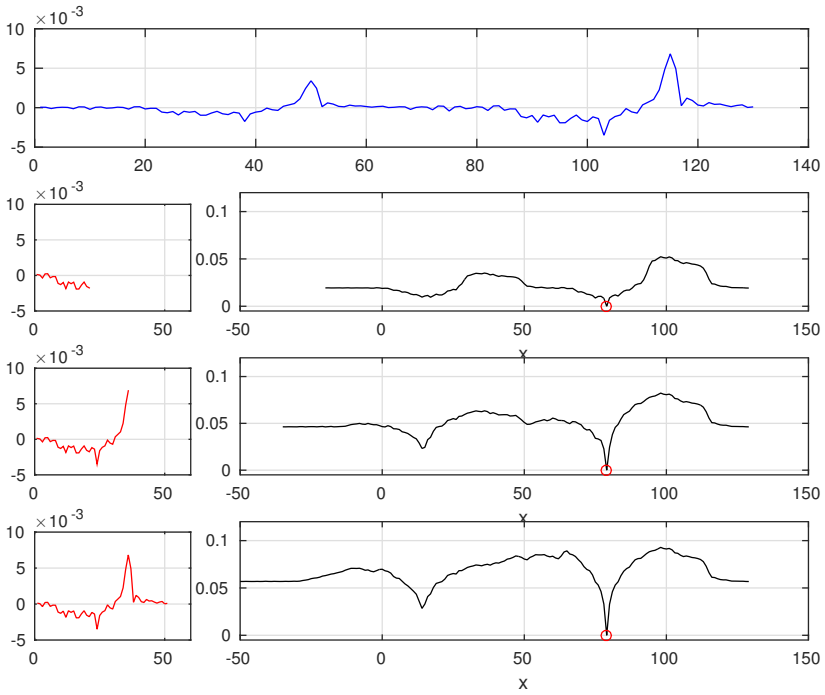
Rysunek 3.11 Przebiegi funkcji CC, SSD i SAD dla pewnego wzorca i sygnału referencyjnego, wraz z zaznaczonymi ekstremami mówiącymi o najlepszym dopasowaniu



Rysunek 3.12 Wpływ przeskalowania przykładowego sygnału $\kappa(s)$ na jego reprezentację (x, y)



Rysunek 3.13 Wartości funkcji SSD dla różnych długości wzorca



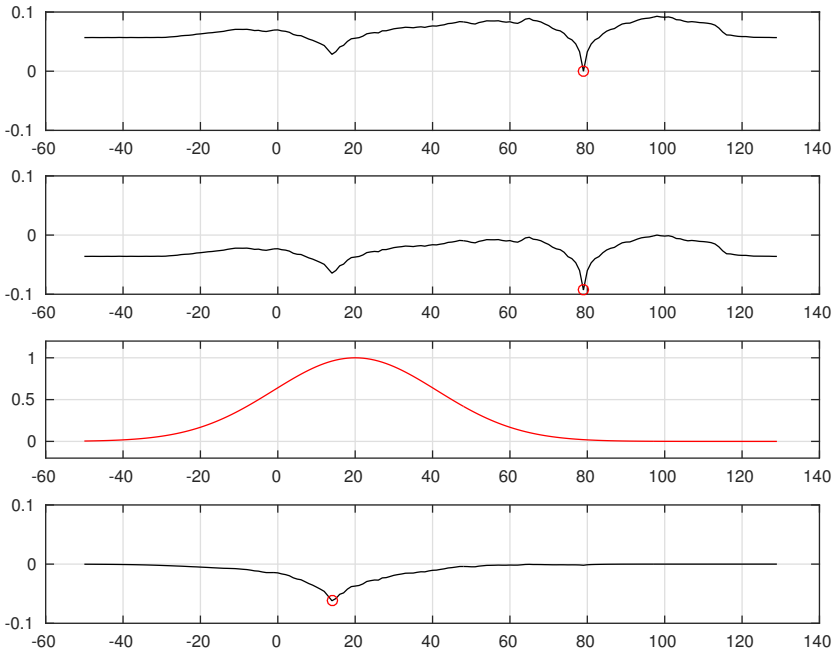
Rysunek 3.14 Wartości funkcji SAD dla różnych długości wzorca

lu zawężenia obszaru poszukiwań, a przez to zwiększenie odporności stosowanej metody na błędy. Znając ostatnie wiarygodne położenie robota na trasie, można założyć, że najprawdopodobniej aktualnie znajduje się on w jego niewielkiej okolicy. W efekcie, pierwszym krokiem, jaki należy podjąć, jest taka modyfikacja sygnału obliczonego metodami SAD/SSD, że punkt o najmniejszym dopasowaniu ma wartość 0. Można tego dokonać odejmując od sygnału wartość elementu maksymalnego. Tak przygotowany sygnał należy następnie pomnożyć przez znormalizowane okno Gaussa

$$G(x) = e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \tag{3.9}$$

gdzie μ wynika z zakładanego najbardziej prawdopodobnego położenia robota na trasie, a σ określa szerokość obszaru poszukiwań. Teraz minimum tego sygnału określa najlepsze dopasowanie wzorca do sygnału referencyjnego, z uwzględnieniem prawdopodobieństwa wynikającego ze znanego wcześniej położenia robota na trasie. Przykład podobnego rozwiązania opisany jest w [13].

Należy zauważyć, że algorytmy SAD/SSD cechują się złożonością obliczeniową $O(n^2)$ i czas ich obliczeń dla długich tras może być rzędu setek milisekund, co pozwala na dokonywanie lokalizacji co najwyżej kilka razy na sekundę. W celu zapewnienia większej częstotliwości wyznaczania położenia na trasie, co może być potrzebne do zapewnienia gładkiego sterowania robotem, proponuje się więc użycie fuzji sygnałów obliczonych z SAD/SSD oraz zmierzonych z enkoderów. W tym wypadku algorytm SAD/SSD zapewniłby poprawne wyznaczenie bezwzględnego położenia robota na trasie z małą częstotliwością, a odczyty z enkoderów, liczone od ostatnio obliczonego położenia metodą SAD/SSD, korygowałyby położenie zapewniając wymaganą częstotliwość pomiarów.



Rysunek 3.15 Przykład użycia okna Gaussa do zamodelowania prawdopodobieństwa wyznaczenia położenia robota na trasie. od góry: surowy sygnał SAD, sygnał SAD po odjęciu wartości maksymalnej, zastosowane okno Gaussa, zmodyfikowana wartość SAD uwzględniająca prawdopodobieństwo położenia robota

Rozdział 4

Optymalizacja ruchu robota

Znajomość kształtu trasy i chęć wykorzystania tej wiedzy podczas przejazdu finalnego pociąga za sobą konieczność zaplanowania tego przejazdu. Zadanie to oznacza określenie prędkości wzdłuż trasy oraz modyfikację kształtu ścieżki, którą pokona robot.

4.1 Wyznaczenie optymalnego profilu prędkości

Znając przebieg trasy określony jej krzywizną w funkcji odległości $\kappa(s)$, konieczne jest wyznaczenie takiego profilu prędkości $v(s)$, który pozwoli na przejechanie trasy w minimalnym czasie. Sposób wyznaczenia $v(s)$ bazuje na [21, 22] i opiera się na założeniu, że w każdym punkcie trasy wykorzystywane jest całe „dostępne" (wynikające ze współczynnika tarcia kół robota) przyspieszenie, tzn. zachodzi

$$\|\mu g\| = \|a_{lat} + a_{long}\|, \tag{4.1}$$

gdzie μ jest współczynnikiem tarcia, g wektorem grawitacji, natomiast a_{long} i a_{lat} oznaczają odpowiednio wektory przyspieszenia wzdłużnego i dośrodkowego. Stąd wartości przyspieszeń spełniają relację

$$(\mu g)^2 = a_{lat}^2 + a_{long}^2. \tag{4.2}$$

Jednocześnie z definicji

$$a_{lat} = v^2|\kappa|, \tag{4.3}$$

$$a_{long} = \frac{dv}{dt}, \tag{4.4}$$

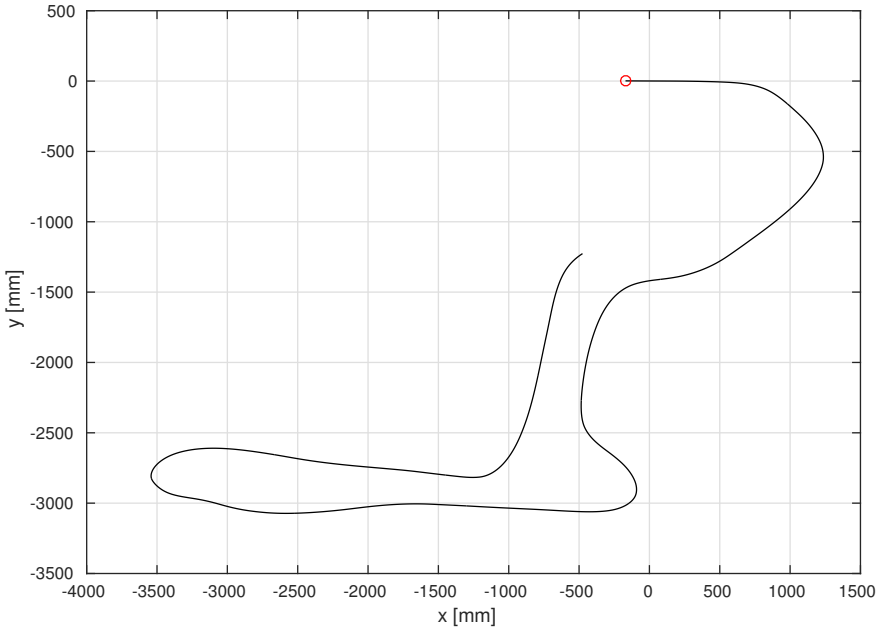
$$v = \frac{ds}{dt}. \tag{4.5}$$

Pozwala to na zapisanie warunku (4.2) w formie

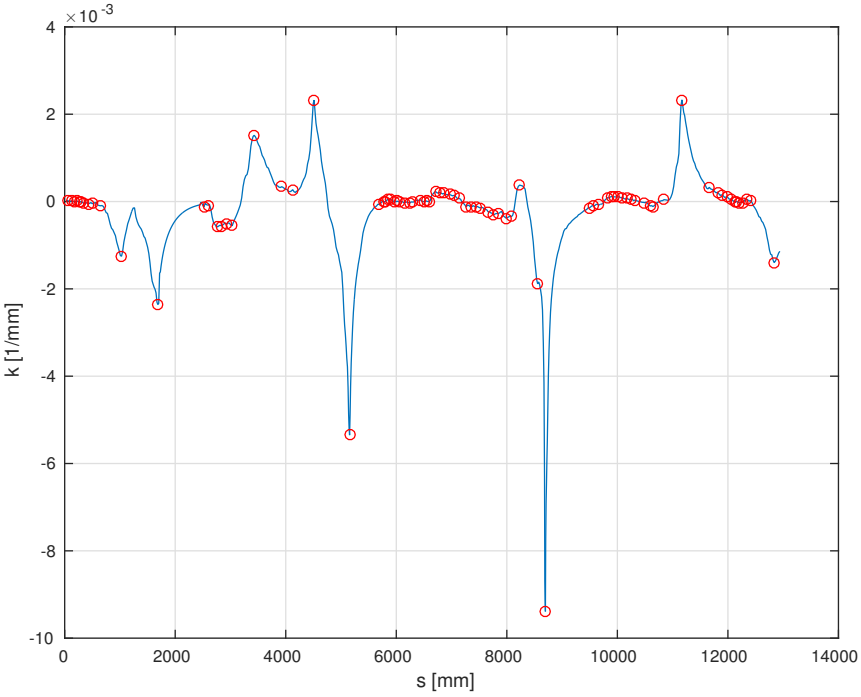
$$\mu^2 g^2 = v^4|\kappa|^2 + v^2\left(\frac{dv}{ds}\right)^2. \tag{4.6}$$

Równanie (4.6) posłuży do wyznaczenia wartości dv, a to pozwoli na obliczenie profilu prędkości $v(s)$. W równaniu tym brakuje jednak informacji o aktualnym znaku przyspieszenia, a przez to również o znaku dv. Konieczne jest więc poczynienie dodatkowych założeń. I tak w każdym punkcie, gdzie krzywizna ma swoje ekstremum lokalne, rozumiane jako maksimum dla wartości dodatnich i minimum dla wartości ujemnych (rysunek 4.2), przyjmuje się chwilową niezmienność prędkości wzdłużnej, co można zapisać jako

$$a_{long} = 0. \tag{4.7}$$



Rysunek 4.1 Trasa testowa z zaznaczonym punktem startowym



Rysunek 4.2 Krzywizna trasy w funkcji odległości $\kappa(s)$ obliczona na podstawie zebranych danych (rysunek 4.1) oraz ekstrema, od których rozpoczyna się obliczanie profili prędkości

Oznacza to, że całe „dostępne" przyspieszenie przyjmuje rolę przyspieszenia dośrodkowego

$$(\mu g)^2 = v^2|\kappa|, \tag{4.8}$$

dzięki czemu możliwe jest jednoznaczne obliczenie wartości prędkości w tym punkcie

$$v = \sqrt{\frac{\mu g}{|\kappa|}}. \tag{4.9}$$

Znając prędkość w takim punkcie zakłada się następnie, że przed nim występuje hamowanie, a za nim – przyspieszanie. Dzięki temu możliwe jest obliczenie prędkości w punktach sąsiadujących

$$v_{i-1} = v_i + dv, \tag{4.10}$$

$$v_{i+1} = v_i + dv. \tag{4.11}$$

Zmiana prędkości dv obliczana jest na podstawie (4.6) jako

$$dv = \sqrt{\frac{(\mu^2 g^2 - v^4|\kappa|^2)ds^2}{v^2}}, \quad dv \geqslant 0. \tag{4.12}$$

Obliczając kolejne prędkości odpowiednio przed i za punktem ekstremum, korzystając z równań (4.10), (4.11) i (4.12), możliwe jest wyznaczenie kompletnego profilu prędkości $v_i(s)$ dla i-tego ekstremum. Czynność tą należy powtórzyć dla wszystkich ekstremów na trasie, a profil ostateczny $v^*(s)$, uwzględniający wszystkie ograniczenia, wyznaczają prędkości minimalne w każdym punkcie trasy, tzn.

$$v^*(s) = \min_{\forall i \ \in E}(v_i(s)), \tag{4.13}$$

gdzie E jest zbiorem wszystkich ekstremów lokalnych krzywizny w reprezentacji $\kappa(s)$, rozumianych jako maksima dla wartości dodatnich i minima dla wartości ujemnych.

W kwestiach implementacji wartym uwagi jest fakt, że nie ma konieczności zapamiętywania wszystkich profili $v_i(s)$, których może być bardzo dużo. Możliwe jest obliczanie aktualnego profilu optymalnego $v_i^*(s)$ jako minimum profilu poprzedniego i profilu wyznaczonego dla aktualnego ekstremum, tzn.

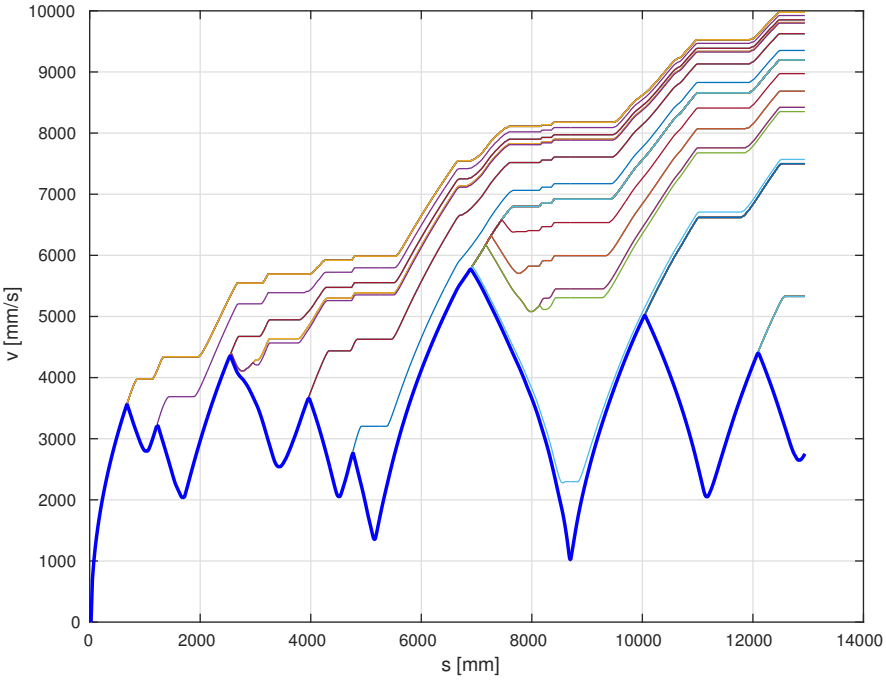
$$v_i^*(s) = \min(v_{i-1}^*(s), v_i(s)). \tag{4.14}$$

Taki sposób postępowania prezentuje rysunek 4.3.

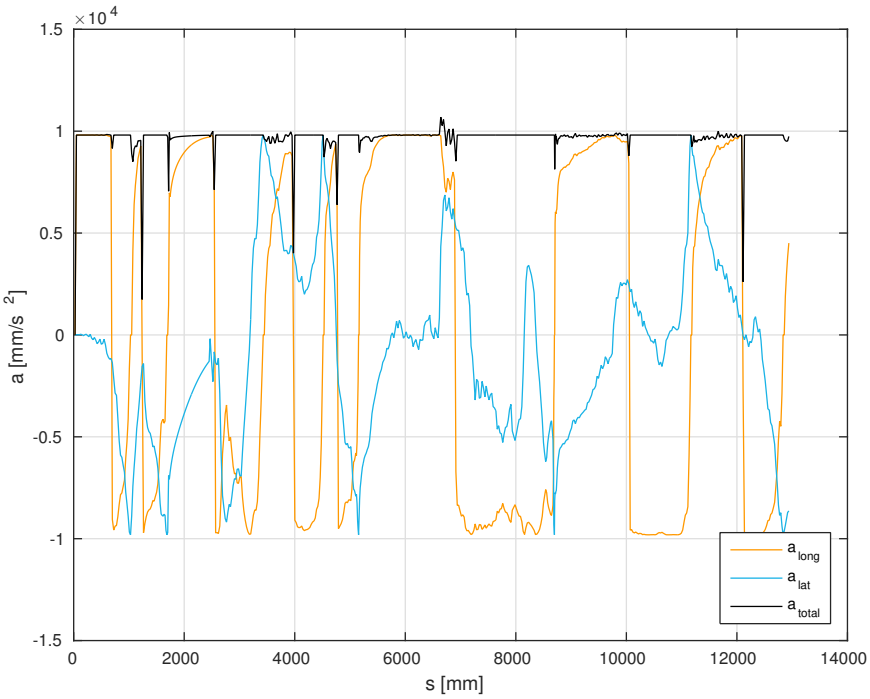
Ciekawą obserwacją może być również to, że występująca w równaniu (4.12) wartość μ nie musi być stała. Przykładowo możliwe byłoby wyznaczenie wartości $\mu(s)$ i uwzględnienie brudzenia się opon robota wraz z przejechaną odległością. W przypadku robotów linefollower zjawisko to może powodować zauważalne efekty nawet po przejechaniu kilkunastu metrów.

Sposób wyznaczania profilu prędkości (4.13) pozwala na łatwe dodawanie kolejnych ograniczeń prędkościowych. Generując nowy profil prędkości zaczynający się w punkcie $v(0) = 0$, możliwe jest zamodelowanie zerowej prędkości na początku trasy zgodnie z wzorem (4.11). Uzupełniając natomiast zestaw profili profilem $v(s) = const.$, możliwe jest uwzględnienie prędkości maksymalnej charakterystycznej dla robota. Oczywiście w tym przypadku nie ma gwarancji, że warunek (4.6) będzie spełniony w każdym punkcie trasy.

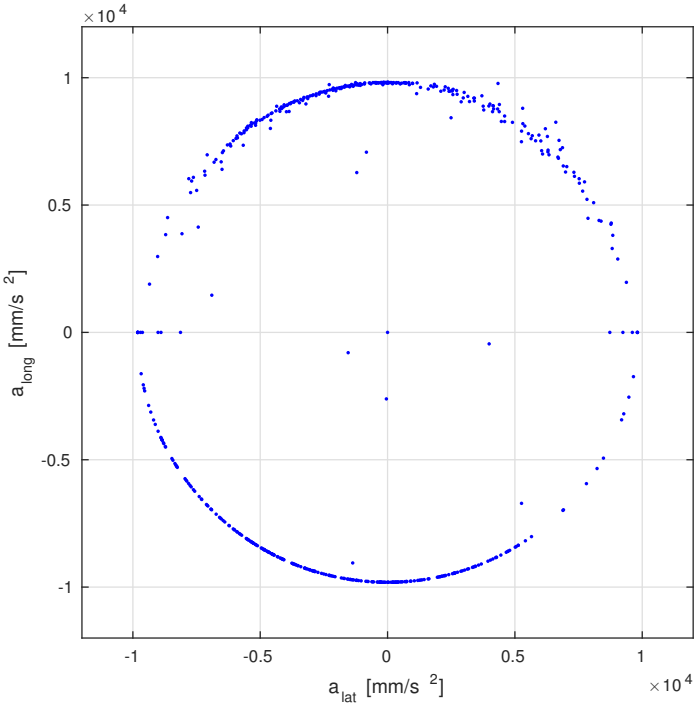
Rysunek 4.4 prezentuje rozkład przyspieszeń wzdłużnego a_{long}, dośrodkowego a_{lat} i całkowitego a_{total} wzdłuż trasy 4.1 dla obliczonego profilu prędkości 4.3 przy $\mu = 1$. Rysunek



Rysunek 4.3 Profile prędkości wyznaczane w kolejnych iteracjach algorytmu oraz profil finalny (niebieski)



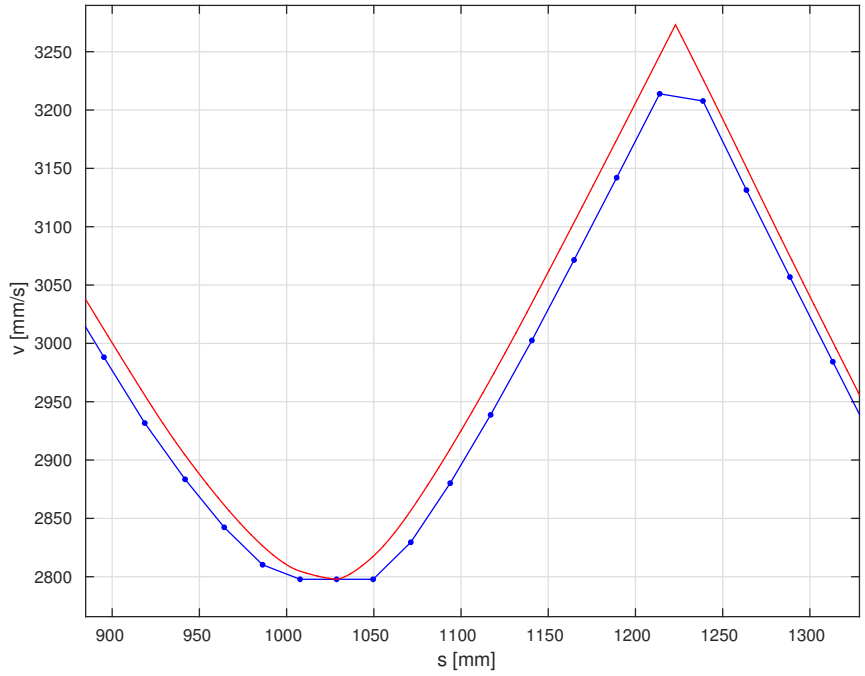
Rysunek 4.4 Przyspieszenia w funkcji przejechanej odległości dla obliczonego profilu prędkości



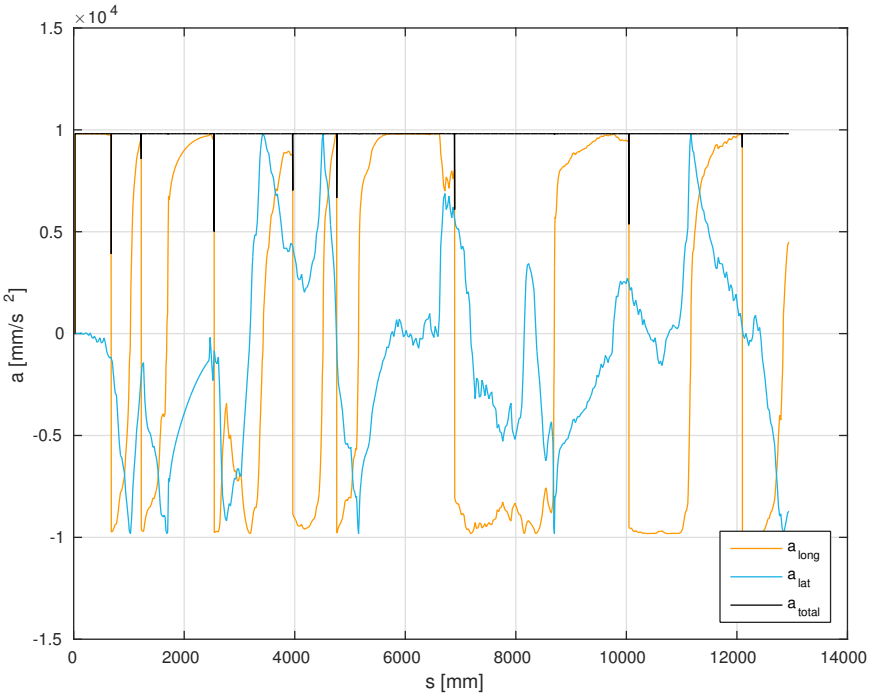
Rysunek 4.5 Rozkład przyspieszeń wzdłużnych i dośrodkowych dla obliczonego profilu prędkości

4.5 pokazuje natomiast rozkład przyspieszeń wzdłużnych i dośrodkowych na płaszczyźnie XY. Wątpliwości może wzbudzać fakt występowania nieciągłości przyspieszenia całkowitego w punktach, gdzie krzywizna ma swoje ekstrema. Może to być spowodowane tym, że przyspieszenie dośrodkowe jest obliczane wprost na podstawie krzywizny i prędkości w danym punkcie, natomiast przyspieszenie wzdłużne jest przybliżane ilorazem różnicowym na podstawie prędkości dwóch punktów sąsiadujących. Chcąc potwierdzić te przypuszczenia sprawdzono zachowanie algorytmu obliczającego optymalny profil prędkości dla $ds \to 0$. W tym celu sygnał wejściowy $\kappa(s)$, mający w opisywanym przypadku 580 próbek, poddano znacznemu nadpróbkowaniu, uzyskując sygnał $\kappa^*(s)$ o długości 100000. Wynik działania algorytmu dla takiego wektora wejściowego prezentują rysunki 4.6, 4.7 oraz 4.8.

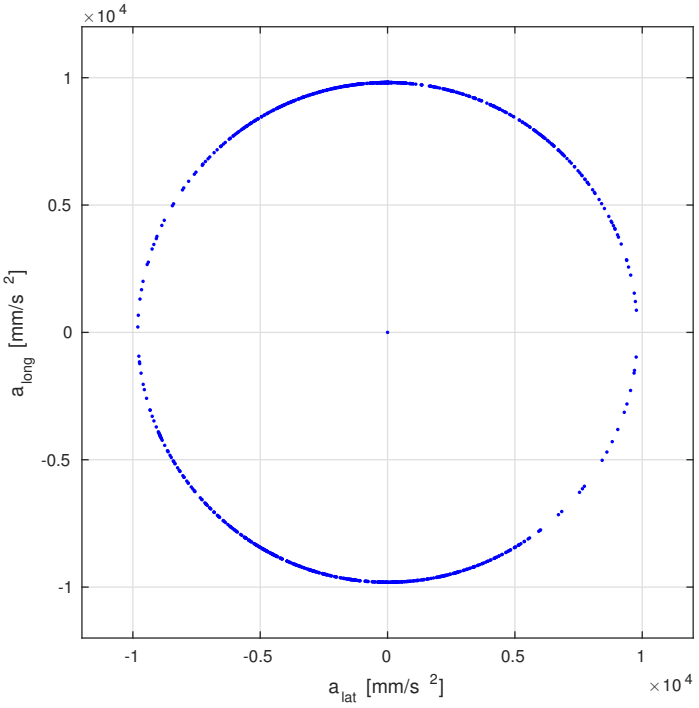
Zastosowanie znacznego nadpróbkowania sygnału $\kappa(s)$ daje niewielką, aczkolwiek zauważalną, poprawę wyznaczonej charakterystyki prędkości i rozkładu przyspieszeń wzdłuż trasy. Ze względu na duży koszt obliczeniowy zostało jednak przedstawione głównie jako dowód poprawności rozumowania, a nie jako niezbędna część algorytmu wyznaczającego optymalny profil prędkości. Zarówno dla rozkładów przyspieszeń 4.4, jak i 4.7, daje się zauważyć, że całkowite przyspieszenie w przeważającej części trasy osiąga wartość zbliżoną do tej wyznaczonej warunkiem (4.1). Mimo, iż dla pierwszego profilu występują punkty o minimalnie większym, niż założone maksimum, przyspieszeniu, to należy mieć przede wszystkim na uwadze fakt, że wyznaczenie profilu prędkości pierwotnie bazuje na z góry założonym współczynniku tarcia, który będąc wyznaczanym eksperymentalnie, musi uwzględniać pewien margines bezpieczeństwa. Niewielkie i chwilowe przekroczenie wartości przyspieszenia ponad założony limit nie powinno więc powodować problemów w jeździe robota.



Rysunek 4.6 Fragmenty optymalnych profilów prędkości obliczonych na podstawie wektorów wejściowych $\kappa(s)$ (niebieski) oraz $\kappa^*(s)$ (czerwony)



Rysunek 4.7 Przyspieszenia w funkcji przejechanej odległości dla optymalnego profilu prędkości obliczonego na podstawie $\kappa^*(s)$



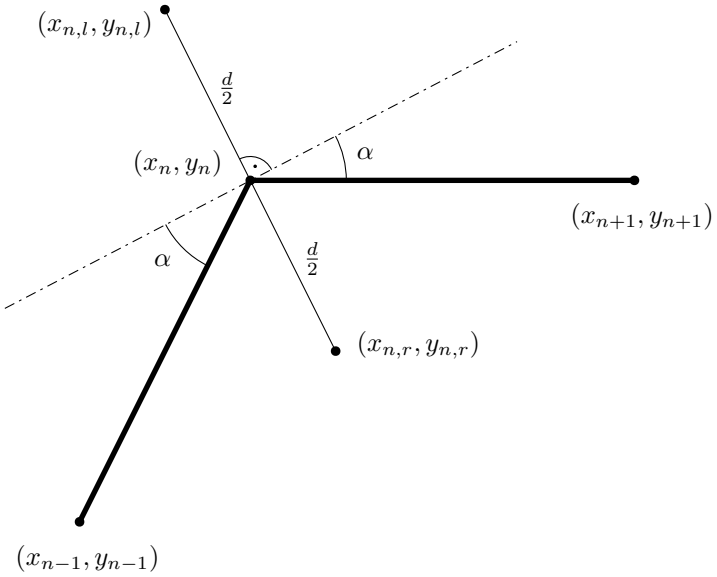
Rysunek 4.8 Rozkład przyspieszeń wzdłużnych i dośrodkowych dla optymalnego profilu prędkości obliczonego na podstawie $\kappa^*(s)$. Wykres przedstawia tylko 1% danych ze względu na ogromną ich ilość

4.2 Optymalizacja kształtu ścieżki

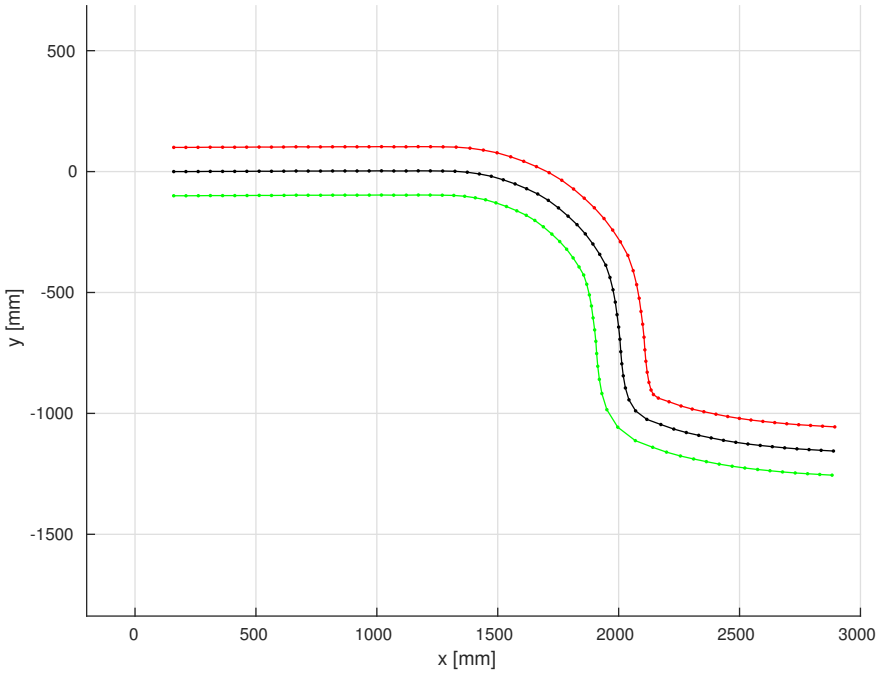
Zgodnie z algorytmem zaprezentowanym w rozdziale 4.1, dla danej trasy i przy założonych ograniczeniach (np. współczynnik tarcia, prędkość początkowa), istnieje pewien profil prędkości $v(s)$, który w efekcie daje minimalny czas przejazdu t_{min}. Oznacza to, że aby dodatkowo zmniejszyć czas przejazdu, konieczna jest modyfikacja kształtu trasy. Fakt, iż robot posiada pewną niezerową szerokość d , oraz zasady konkurencji linefollower mówiące o tym, że robot swoim obrysem nie może wyjechać poza linię, dają pewne możliwości w tym obszarze. Ściślej rzecz ujmując w każdym punkcie trasy możliwe jest przesunięcie robota maksymalnie o odległość $\frac{d}{2}$ w lewo lub prawo. Zadanie polega więc na wyznaczeniu takiego przesunięcia $l(s)$, $l \in [-\frac{d}{2}, \frac{d}{2}]$ w każdym punkcie trasy, aby nowo powstały kształt toru przejechanego przez robota dał minimalny czas przejazdu, mniejszy od wyznaczonego wcześniej t_{min}.

Poprzez redukcję rozmiarów robota do zera, przy jednoczesnym powiększeniu obszaru dopuszczalnych jego położeń, możliwe jest zapisanie wyżej postawionego problemu w równoważnej formie. Korzystając ze znanego kształtu linii, zapisanego jako zbiór punktów (x_n, y_n), należy wyznaczyć punkty graniczne $(x_{n,l}, y_{n,l})$ oraz $(x_{n,r}, y_{n,r})$ w taki sposób, że leżą one odpowiednio po jej lewej i prawej stronie w odległości $\frac{d}{2}$ od niej. Odcinek łączący te punkty powinien być jednocześnie prostopadły do stycznej w punkcie (x_n, y_n). Sposób wyznaczenia punktów ilustruje rysunek 4.9, natomiast wyznaczone granice dla przykładowej trasy są pokazane na rysunku 4.10.

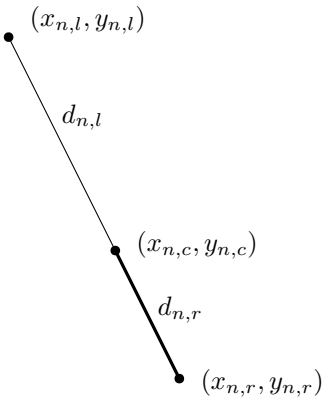
Zadanie postawione w takiej formie okazuje się już rozwiązane w kontekście wyznaczenia optymalnej trasy przejazdu samochodu wyścigowego na torze [10, 8]. Dostępna jest również analiza ruchu robota klasy linefollower na trasie przy założeniu rozdzielności



Rysunek 4.9 Wyznaczenie punktów $(x_{n,l}, y_{n,l})$ i $(x_{n,r}, y_{n,r})$, używanych jako granice obszaru dopuszczalnych położeń robota przy wyznaczeniu optymalnej trasy przejazdu



Rysunek 4.10 Wyznaczenie lewej i prawej granicy obszaru dopuszczalnych położeń dla danego kształtu trasy



Rysunek 4.11 Interpretacja graficzna parametru α. Punkt kontrolny $(x_{n,c}, y_{n,c})$ dzieli odcinek między punktami granicznymi $(x_{n,l}, y_{n,l})$ i $(x_{n,r}, y_{n,r})$ na dwie części o długościach odpowiednio $d_{n,r}$ i $d_{n,l}$, przy czym $d = d_{n,r} + d_{n,l}$ jest odległością między punktami granicznymi. Parametr α to stosunek długości odcinka $d_{n,r}$ do odcinka d, tzn. $\alpha = \frac{d_{n,r}}{d}$

kolejnych zakrętów [19]. Wynikiem dającym minimalny czas przejazdu po trasie ograniczonej daną szerokością, jest ścieżka o minimalnej długości, minimalnej krzywiźnie, lub też pewna suma ważona dwóch poprzednich. Przy dominujących ograniczeniach związanych z prędkością maksymalną pojazdu rozwiązanie jest bliskie trasie o najmniejszej długości, natomiast jeśli czynnikiem przeważającym są limity przyspieszeń (np. współczynnik tarcia, dostępna moc silników), to rozwiązanie optymalne będzie przypominało trasę o minimalnej krzywiźnie.

Przykładowe rozwiązanie [9] sprowadza problem do zadania optymalizacji kwadratowej i opisuje sposoby wyznaczenia ścieżki zarówno o minimalnej długości, jak też o minimalnej krzywiźnie. Trasa reprezentowana jest przez zbiór par punktów (x_l, y_l), (x_r, y_r) wyznaczających granice poszukiwań. Wprowadzane jest również pojęcie punktu kontrolnego (x_c, y_c), tzn. takiego miejsca, w którym ścieżka przecina odcinek między lewą a prawą granicą. Dla każdej takiej pary wyznaczany jest pewien współczynnik $\alpha_n \in [0, 1]$ mówiący o tym, w którym miejscu pomiędzy punktami granicznymi leży punkt kontrolny. Interpretację graficzną punktu kontrolnego oraz współczynnika α przedstawia rysunek 4.11. Problem sprowadzony jest do znalezienia takiego wektora współczynników α, że ścieżka wyznaczona przez punkty kontrolne spełnia warunek minimalnej długości lub krzywizny, i daje się zapisać w postaci

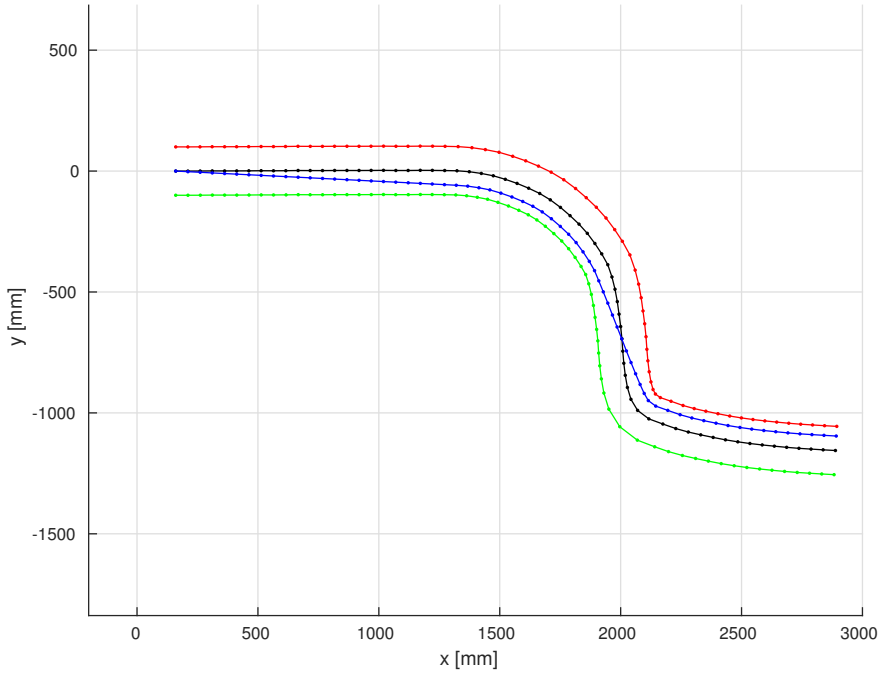
$$\min_{\alpha} \alpha^T H_S \alpha + B_S \alpha, \tag{4.15}$$

gdzie H_S oraz B_S są odpowiednio pewną macierzą i wektorem zależnymi tylko od położeń punktów granicznych, natomiast α jest poszukiwanym wektorem współczynników, na podstawie którego możliwe jest wyznaczenie punktów kontrolnych, a tym samym kształtu zoptymalizowanej trasy. Sposób wyliczenia H_S oraz B_S został opisany dokładnie w [2].

W opisywanym przypadku poszukiwane wartości α_n mieszczą się w przedziale $LB_n \leqslant \alpha_n \leqslant UB_n$, gdzie

$$\forall_n \begin{cases} LB_n = 0 \\ UB_n = 1 \end{cases} . \tag{4.16}$$

Wprowadzając dodatkową zmienną $b \in [0, 0.5]$, nazwaną marginesem bezpieczeństwa,



Rysunek 4.12 Optymalizacja kształtu poprzez znalezienie ścieżki o minimalnej długości (niebieski) dla danego kształtu trasy (czarny) i przy dodatkowym określeniu punktu początkowego oraz marginesu bezpieczeństwa $b = 0.2$

możliwe jest zawężenie obszaru poszukiwań, co w efekcie daje ograniczenia postaci

$$\forall_n \begin{cases} LB_n = 0 + b \\ UB_n = 1 - b \end{cases} . \tag{4.17}$$

Efektem zwiększania b jest oddalanie poszukiwanej trasy od punktów ograniczających, dzięki czemu niewielkie zaburzenia ruchu robota nie skutkują wypadnięciem z trasy. Oprócz tego możliwe jest dokładniejsze ograniczenie obszaru poszukiwań dla konkretnych punktów. Przykładowo założenie wartości

$$\begin{cases} LB_0 = 0.5 \\ UB_0 = 0.5 \end{cases} \tag{4.18}$$

spowoduje umieszczenie punktu początkowego zoptymalizowanej trasy dokładnie pomiędzy punktami ograniczającymi $(x_{0,l}, y_{0,l})$ i $(x_{0,r}, y_{0,r})$.

Rysunek 4.12 przedstawia wynik działania algorytmu minimalizującego długość trasy dla $b = 0.2$ oraz przy założeniu, że początek trasy leży dokładnie pomiędzy początkowymi punktami granicznymi ($LB_0 = 0.5 \ \wedge \ UB_0 = 0.5$). Gwoli ścisłości należy podkreślić, że algorytm ten minimalizuje sumę kwadratów odległości odcinków, z których składa się trasa, co nie jest równoważne minimalizacji długości. Graficzna interpretacja uzyskanych wyników pozwala jednak śmiało stwierdzić, że otrzymany wynik w bardzo dobry sposób przybliża trasę o najmniejszej długości i jest zgodny ze zdroworozsądkowymi przewidywaniami.

4.3 Wyniki optymalizacji

Na podstawie zadanych profili $v(s)$ przeprowadzono obliczenia czasów przejazdów robota dla kilku przykładowych tras. Założono, że sterowanie prędkością robota wzdłuż trasy może odbywać się w sposób:

- zachowawczy, gdzie wiedza o bezwzględnym położeniu robota na trasie nie jest wymagana, a prędkość przejazdu determinuje jej fragment o największej krzywiźnie,
- optymalny, gdzie prędkość na trasie jest dana obliczonym optymalnym profilem $v(s)$,
- optymalny, gdzie dodatkowo prędkość maksymalna jest ograniczona pewną wartością v_{max}.

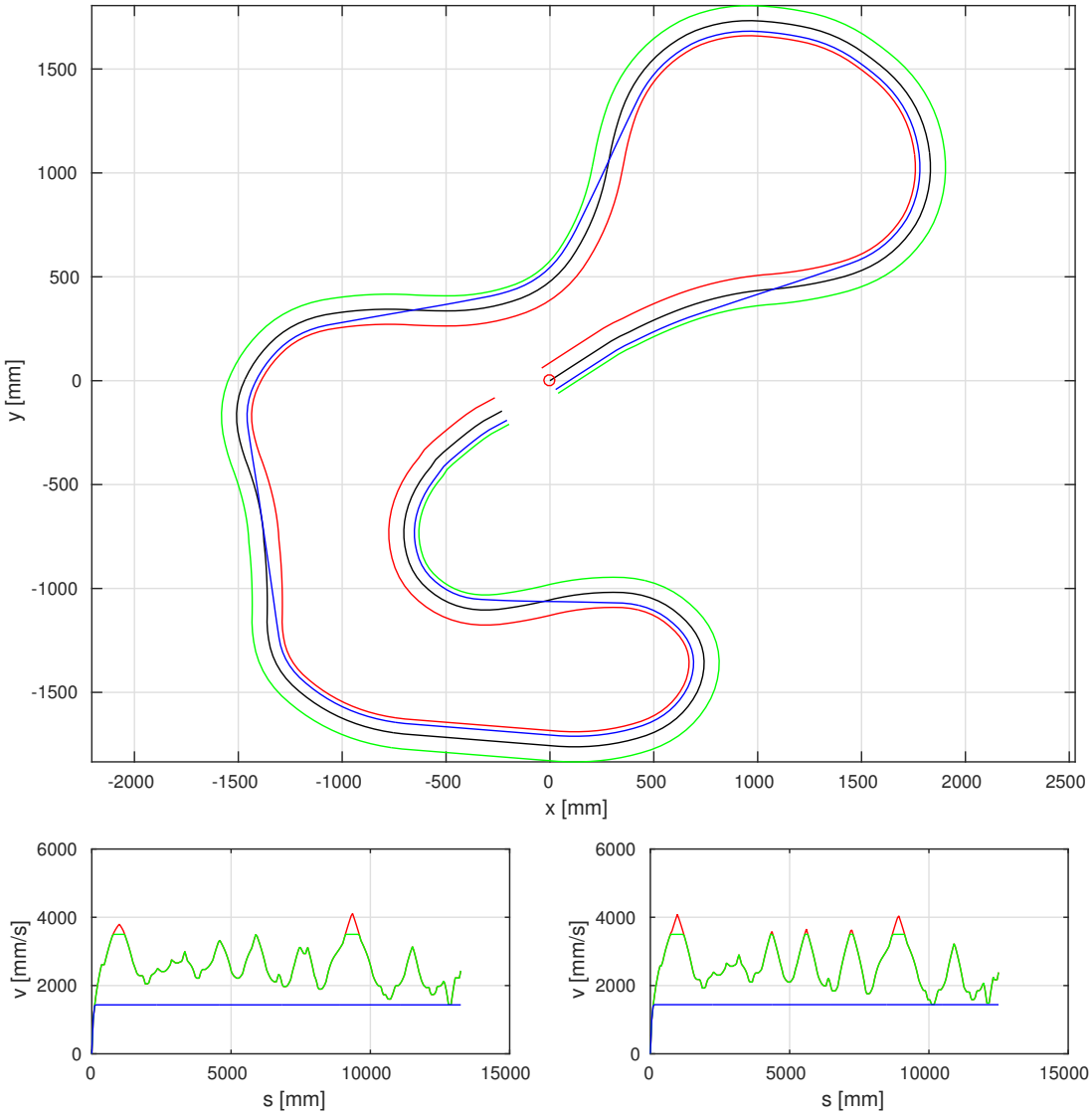
Przeprowadzone symulacje obejmowały sytuacje, gdy:

- robot porusza się po ścieżce odpowiadającej kształtem zadanej trasie,
- trasa realizowana przez robota jest wynikiem minimalizacji jej długości.

Założono przy tym współczynnik tarcia $\mu = 1$, prędkość początkową $v_0 = 0$, prędkość maksymalną $v_{max} = 3.5\frac{m}{s}$ i margines bezpieczeństwa przy wyznaczaniu najkrótszej ścieżki $b = 0.15$. Przyjęto szerokość robota jako $d = 145mm$. Do celów symulacji założono, że metody szacowania położenia linii i robota, opisane w rozdziale 3, dostarczają wiarygodnych wyników, przez co pozwalają na poprawne sterowanie prędkością robota wzdłuż trasy. Daje to możliwość obliczenia czasu przejazdu trasy wprost z wyznaczonego profilu prędkości.

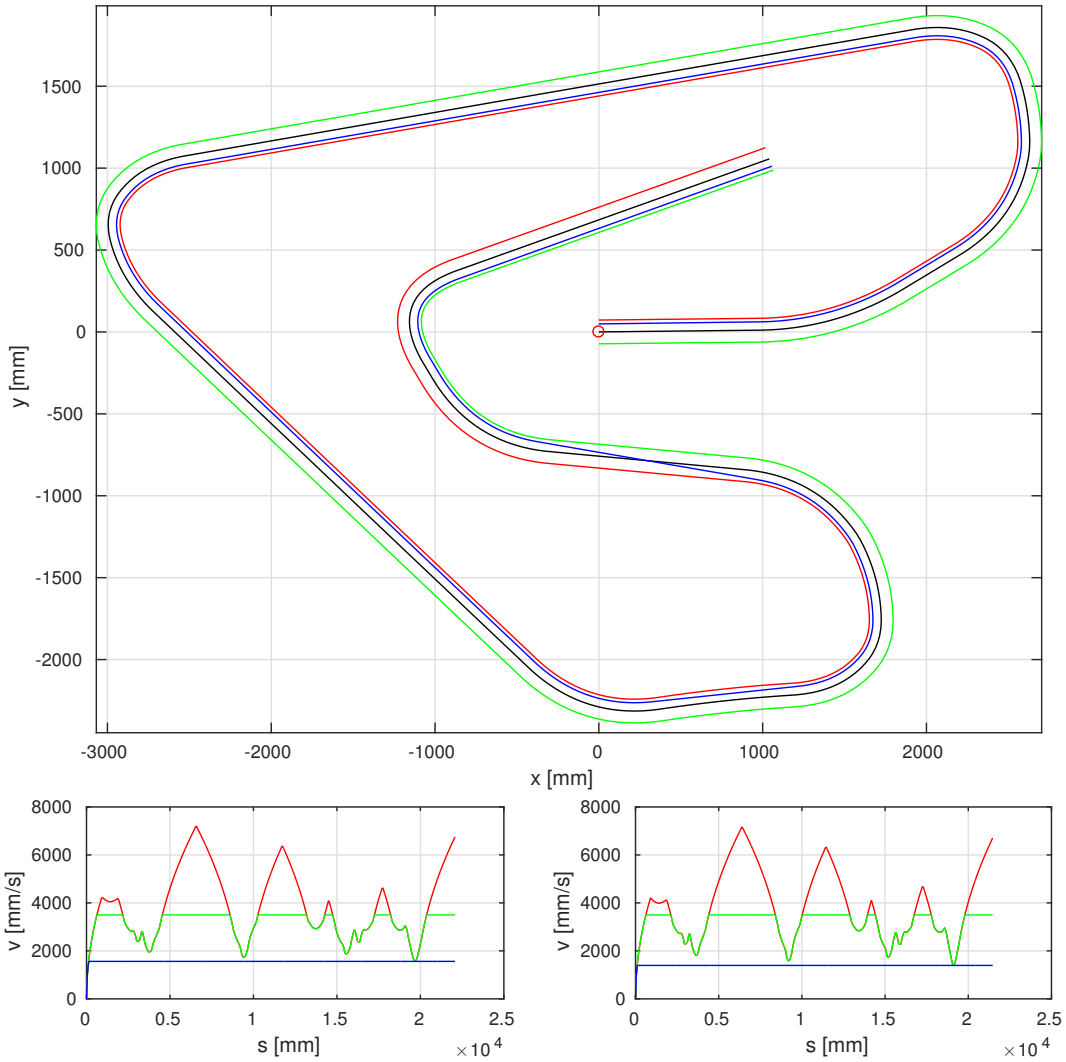
Wybrane wyniki symulacji przedstawiają rysunki 4.13 – 4.17. W dodatku B umieszczono rezultaty uzyskane dla większego zbioru tras testowych. Zawarty na rysunkach wykres na górze przedstawia kształt trasy oraz ścieżkę będącą wynikiem minimalizacji jej długości. Wykres po lewej stronie pokazuje przebieg prędkości wzdłuż trasy przy poruszaniu się po linii, natomiast ten po prawej ukazuje prędkości podczas przejazdu po trasie o minimalnej długości. Kolorem niebieskim oznaczono prędkość przy jeździe zachowawczej, czerwonym – przy jeździe optymalnej, a zielonym – przy jeździe optymalnej z ograniczeniem v_{max}. Tabela na dole podsumowuje wynik symulacji. Analiza otrzymanych danych pozwala stwierdzić, że:

- minimalizacja długości ścieżki daje pozytywne rezultaty tylko w połączeniu z optymalnym sterowaniem prędkością, natomiast przy jeździe bez wiedzy o położeniu robota może powodować drastyczne ograniczenie prędkości na całej trasie,
- optymalne wysterowanie prędkości może skutkować znaczącą poprawą czasu przejazdu,
- minimalizacja długości ścieżki, przy jednoczesnym optymalnym sterowaniu prędkością, daje niewielkie, chociaż zauważalne, zmniejszenie czasu przejazdu,
- maksymalna prędkość osiągana przez robota jest czynnikiem mającym realne znaczenie tylko w przypadku tras, na których występują dostatecznie długie odcinki proste.



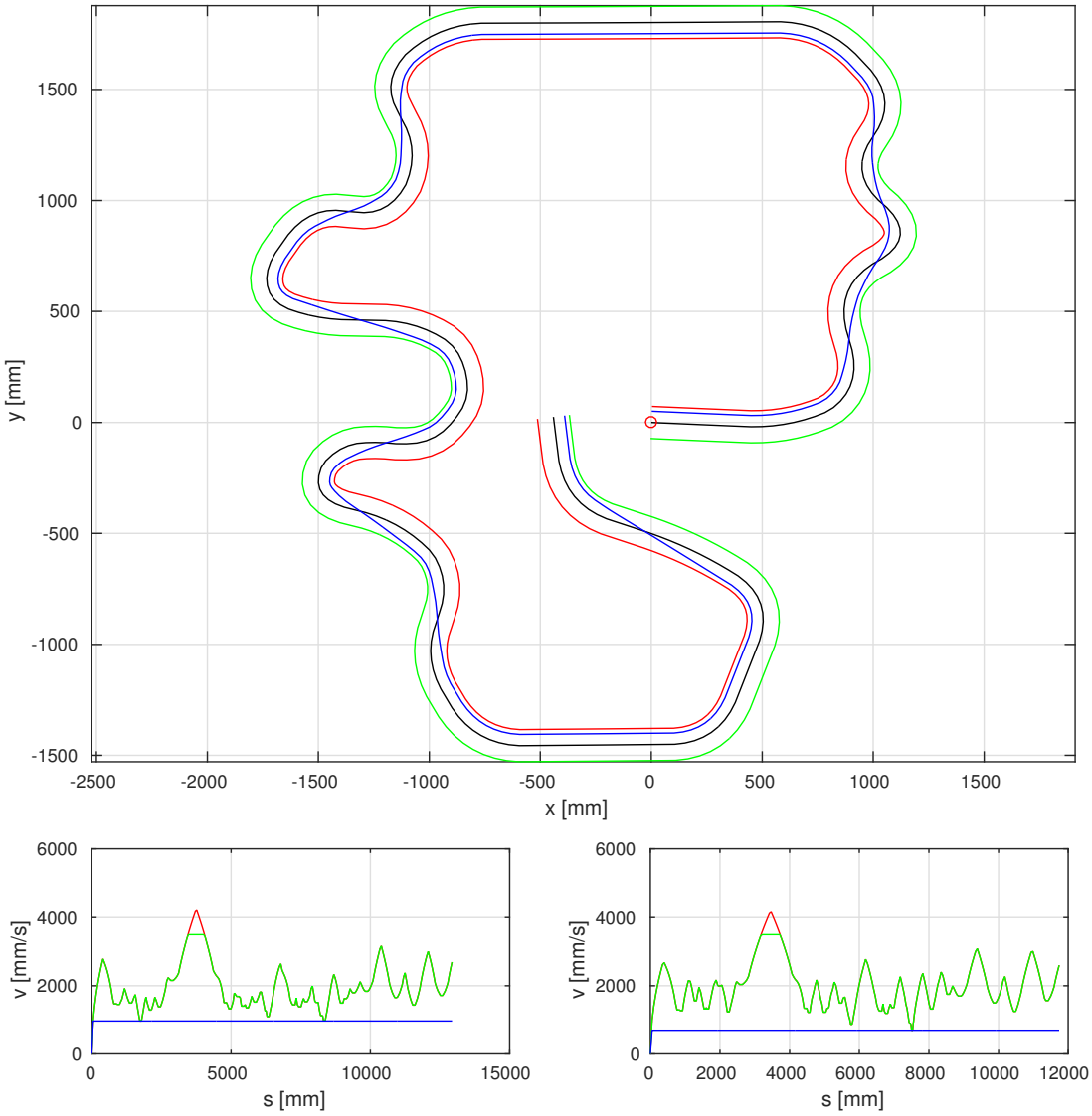
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	13.25	12.50	-5.6%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	1.43	1.44	0.4%
Jazda zachowawcza – czas $[s]$	9.31	8.76	-5.9%
Jazda optymalna – czas $[s]$	5.51	5.30	-3.8%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-40.8%	-39.5%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	5.52	5.32	-3.7%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	0.3%	0.4%	–

Rysunek 4.13 Wyniki symulacji dla przykładowej trasy



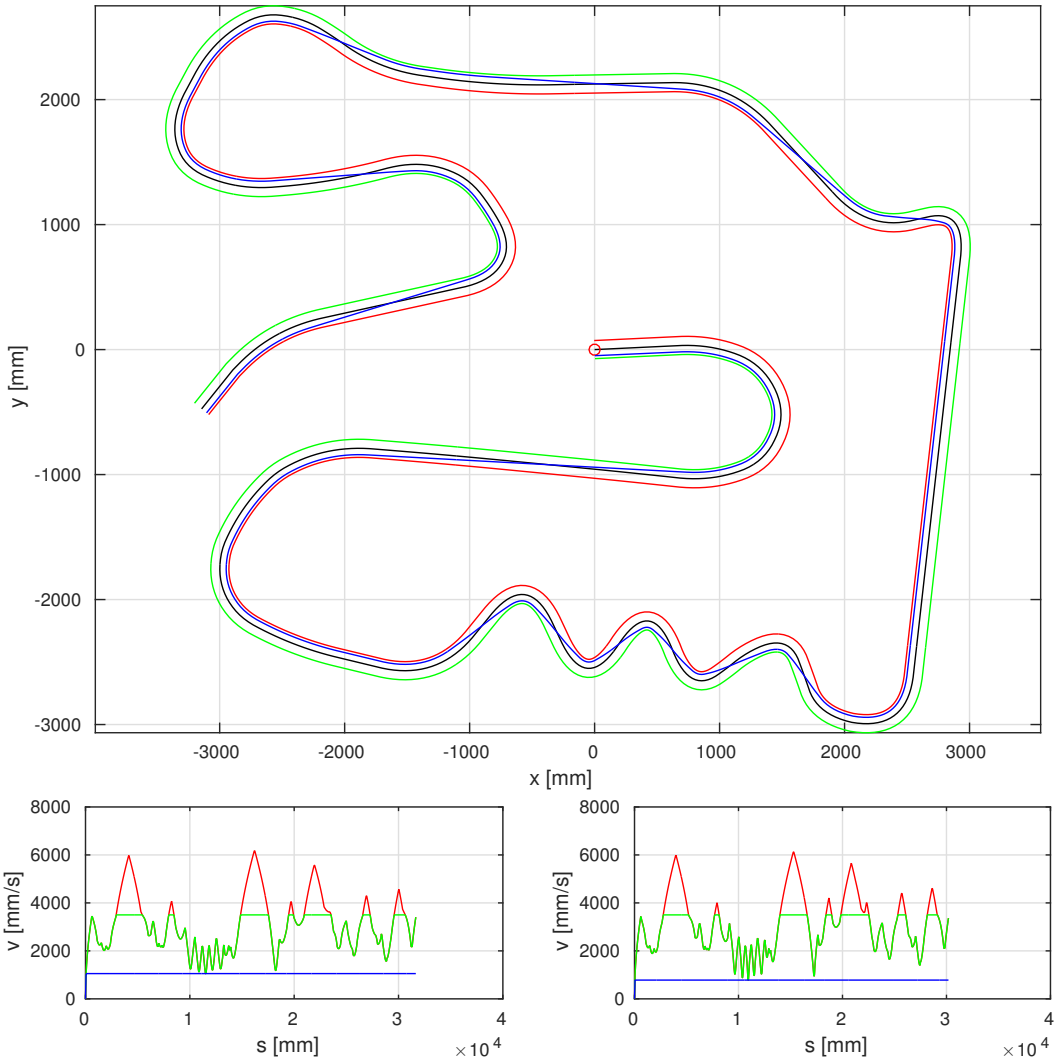
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	22.07	21.47	-2.7%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	1.56	1.39	-10.8%
Jazda zachowawcza – czas $[s]$	14.21	15.48	8.9%
Jazda optymalna – czas $[s]$	6.58	6.51	-1.2%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-53.7%	-58.0%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	7.49	7.39	-1.3%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	13.8%	13.6%	–

Rysunek 4.14 Wyniki symulacji dla przykładowej trasy



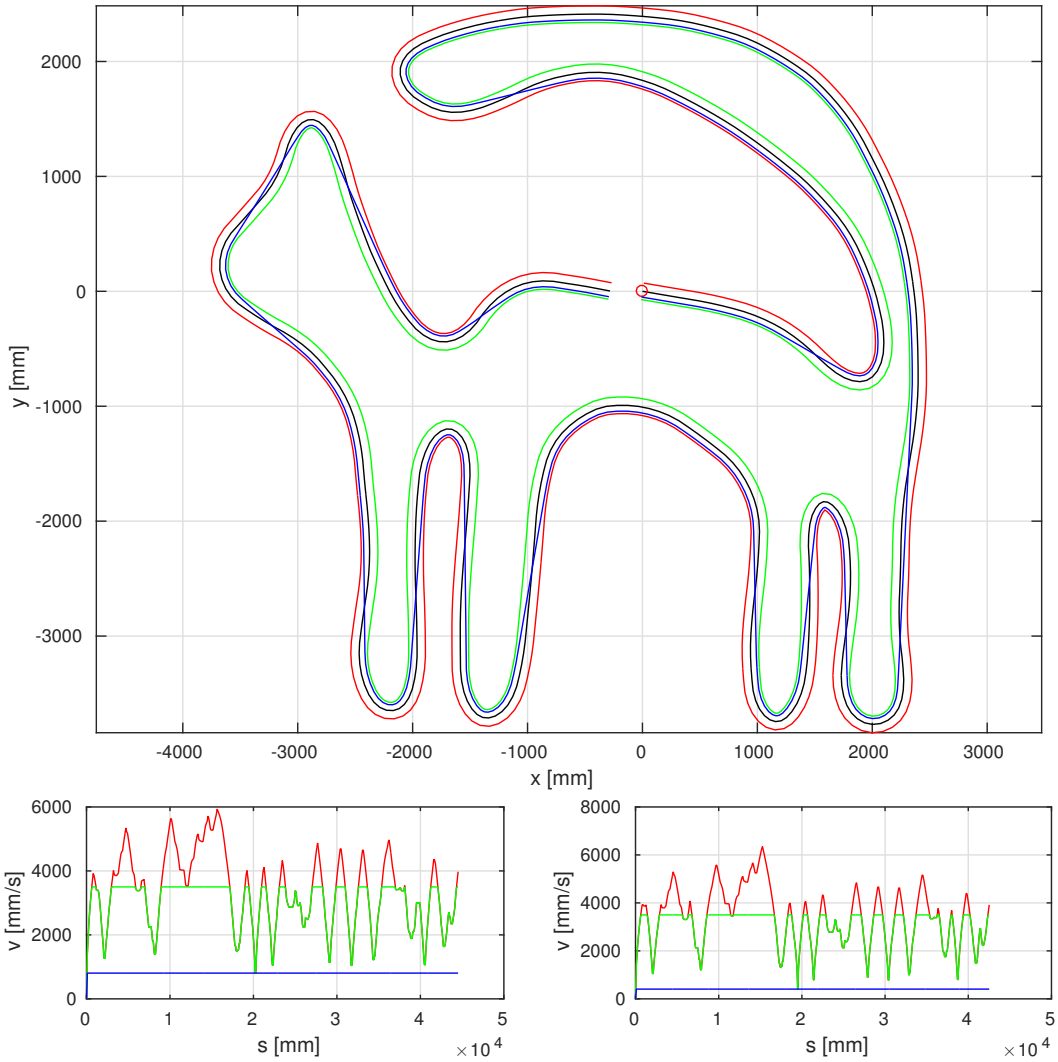
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	12.93	11.74	-9.2%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	0.96	0.66	-31.1%
Jazda zachowawcza – czas $[s]$	13.46	17.75	31.9%
Jazda optymalna – czas $[s]$	7.20	6.56	-8.9%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-46.5%	-63.0%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	7.22	6.57	-8.9%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	0.2%	0.2%	–

Rysunek 4.15 Wyniki symulacji dla przykładowej trasy



	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	31.66	30.19	-4.6%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	1.05	0.78	-25.7%
Jazda zachowawcza – czas $[s]$	30.13	38.68	28.4%
Jazda optymalna – czas $[s]$	11.75	11.38	-3.1%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-61.0%	-70.6%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	12.36	12.01	-2.9%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	5.3%	5.5%	–

Rysunek 4.16 Wyniki symulacji dla przykładowej trasy



	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	44.51	42.52	-4.5%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	0.81	0.41	-49.6%
Jazda zachowawcza – czas $[s]$	55.42	104.95	89.4%
Jazda optymalna – czas $[s]$	15.53	15.06	-3.0%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-72.0%	-85.6%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	16.62	16.15	-2.8%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	7.0%	7.2%	–

Rysunek 4.17 Wyniki symulacji dla przykładowej trasy

Rozdział 5

Podsumowanie

Celem pracy było zaproponowanie sposobu postępowania pozwalającego na przejazd robota klasy linefollower po trasie w minimalnym możliwym czasie. Przedstawiono użyteczne metody reprezentacji trasy, pokazując jednocześnie ich równoważność. Pierwszoplanową intencją autora pracy było pokazanie możliwości sprowadzenia bardzo konkretnych problemów postawionych na jej początku, do problemów pochodnych i już rozwiązanych, gromadząc jednocześnie tę wiedzę w jednym miejscu. Zastosowane algorytmy często są używane w zupełnie innych dziedzinach nauki niż robotyka, przez co już samo nazwanie rozwiązania pochodnego, do którego można byłoby przekształcić analizowany problem, nie zawsze jest łatwe i intuicyjne.

W części opisującej szacowanie położenia zaproponowano metodę korekcji niedoskonałości czujników linii. Opisano sposób zwiększenia dokładności wyznaczenia położenia linii, jak również podejście służące do samego wykrywania obecności linii. Używając dostępnych metod przetwarzania obrazów zaproponowano taki sposób określania położenia robota na trasie, który jest odporny na błędy powodowane użyciem tylko czujników przyrostowych. Zaproponowano również wykorzystanie wiedzy o aktualnym położeniu robota w celu zawężenia obszaru poszukiwań. W części dotyczącej optymalizacji opisano algorytm wyznaczający profil prędkości wzdłuż trasy, zapewniający możliwie maksymalne wykorzystanie warunków fizycznych. Zaprezentowano również sposób optymalizacji kształtu ścieżki. Przeprowadzono symulacje przejazdów robota dla zbioru tras testowych i zobrazowano potencjalne zyski wynikające z wykorzystania opisanych optymalizacji.

Dalsze prace powinny obejmować zaimplementowanie algorytmu obliczającego ścieżkę o najmniejszej krzywiźnie, uwzględnienie mocy silników przy wyznaczaniu optymalnego profilu prędkości, zaimplementowanie algorytmów SAD i SSD szacujących położenie robota na trasie, porównanie tych metod w rzeczywistych warunkach i wykonanie eksperymentów pozwalających na dobranie szerokości sygnału wzorcowego oraz współczynnika σ określającego obszar poszukiwań.

Warto zauważyć, że ze względu na dużą ilość analizowanych danych, problematycznym może być przesyłanie ich z robota do komputera zewnętrznego w czasie rzeczywistym. Sensownym wydaje się więc przeprowadzenie symulacji działania całego robota w celu sprawdzenia opisanych algorytmów na możliwie dużej liczbie tras.

Dodatek A

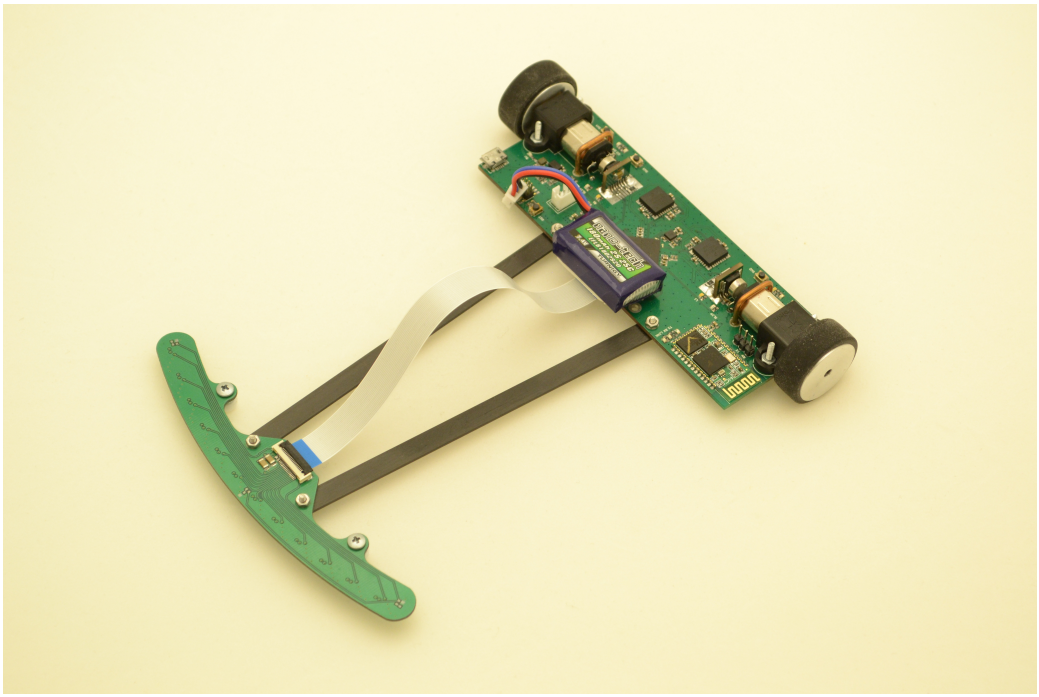
Konstrukcja robota

Konstrukcję robota użytego w pracy stanowią dwie płytki drukowane połączone płaskownikami z włókna węglowego oraz taśmą FFC. Przedstawiają to rysunki A.1 oraz A.2.

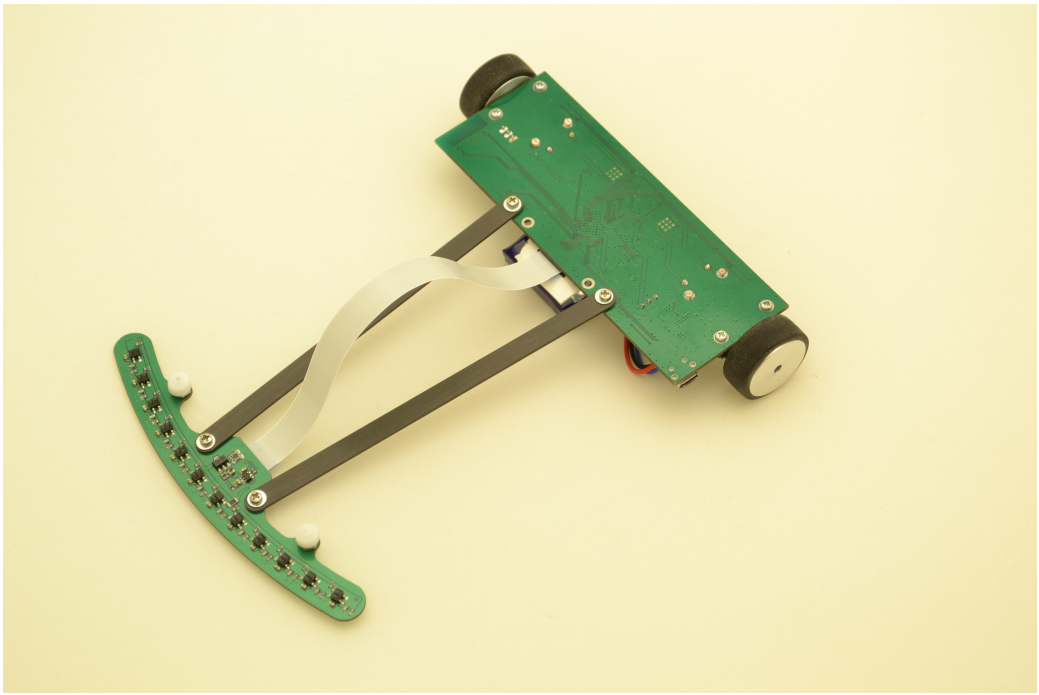
Płytka główna zawiera mikrokontroler MK66FX1M0VMD18 z rdzeniem ARM Cortex-M4F. Cechuje się on maksymalną częstotliwością taktowania 180MHz, pamięcią RAM o rozmiarze 256kB i 1MB pamięci flash. Napęd stanowią dwa silniki Pololu 10:1 HPCB, pozwalające na osiągnięcie na prostej trasie prędkości rzędu $5\frac{m}{s}$. Sterowanie nimi odbywa się przez dwa mostki MC33926, a umieszczone za silnikami enkodery magnetyczne AS5040 pozwalają na precyzyjną kontrolę prędkości obrotowych kół. W osi obrotu znajduje się zintegrowany czujnik MPU9250, zawierający 3-osiowy akcelerometr, żyroskop oraz magnetometr. Robot zasilany jest ogniwem litowo-polimerowym Turnigy Nano-Tech o pojemności 180mAh i napięciu znamionowym 7.4V. do dostarczenia napięcia 3.3V układom logiki użyto przetwornicy TPS62130, a układ scalony LT1618 pozwala na ładowanie baterii przy użyciu standardowych ładowarek USB. Robot wyposażony jest w moduł Bluetooth HC-05 realizujący bezprzewodową komunikację z komputerem, 5 przycisków oraz 15 sygnalizacyjnych diod LED.

Płytka czujników zawiera 12 transoptorów odbiciowych KTIR0711S oraz układ kontroli prądu diod IR. Czujniki odbiciowe rozmieszczono co $9.5mm$ na łuku o promieniu $160mm$, co daje kąt między nimi około $3.4°$. Płytka wyposażona jest w dwa teflonowe ślizgacze, pozwalające na poruszanie się tej części robota z minimalnymi oporami.

Robot cechuje się szerokością $145mm$, długością $175mm$ i masą $100g$. Rozstaw kół wynosi $135mm$. Średnica kół z oponami to $26mm$. Zastosowane opony pochodzą z popularnych modeli samochodów Mini-Z i dobrze oczyszczone pozwalają osiągnąć na typowym podłożu, jakim jest laminowana płyta HDF, współczynnika tarcia rzędu $\mu = 2$. Odległość między osią obrotu robota a łukiem wyznaczanym przez czujniki linii, wynosi $160mm$.



Rysunek A.1 Widok z góry

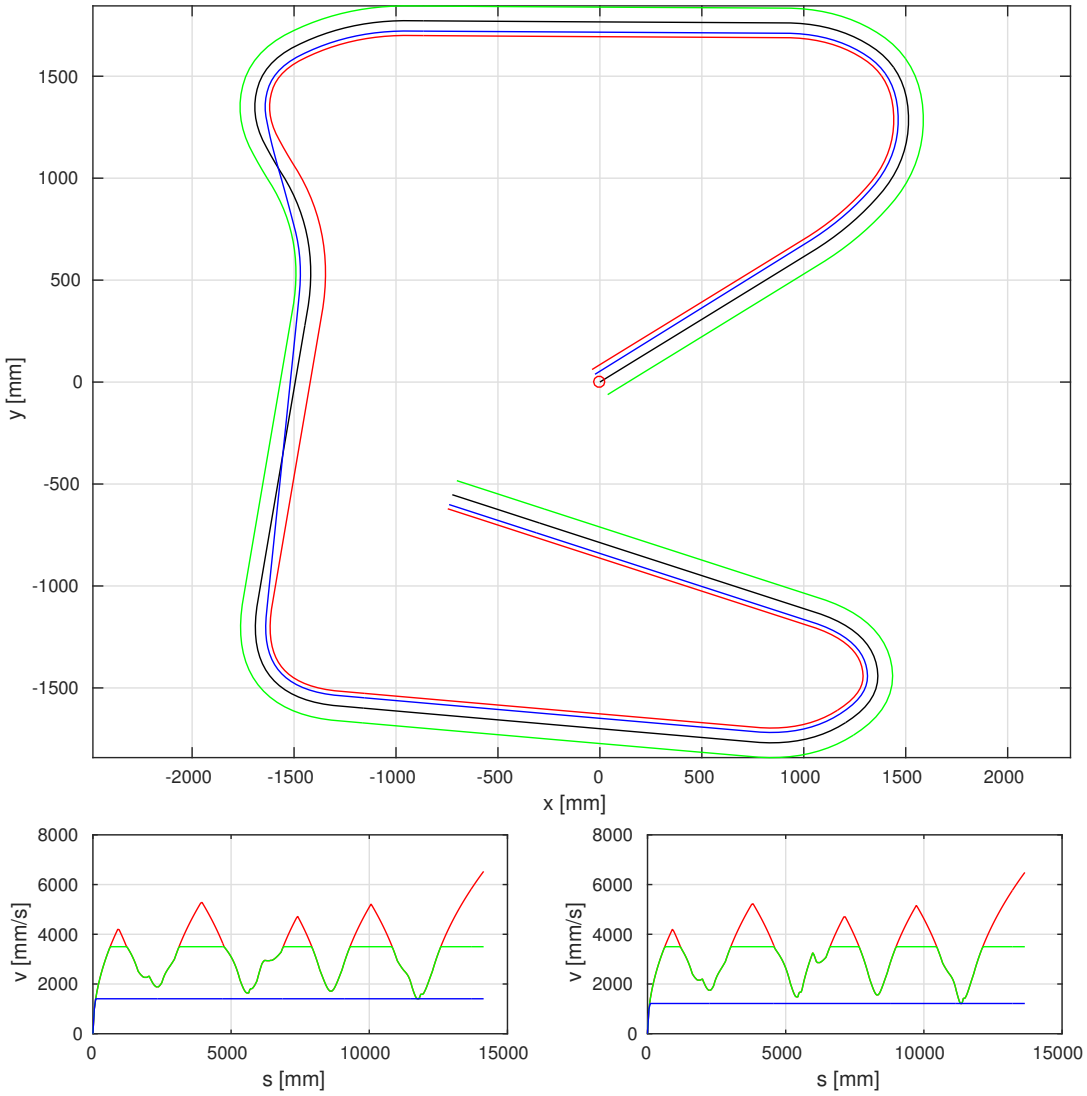


Rysunek A.2 Widok z dołu

Dodatek B

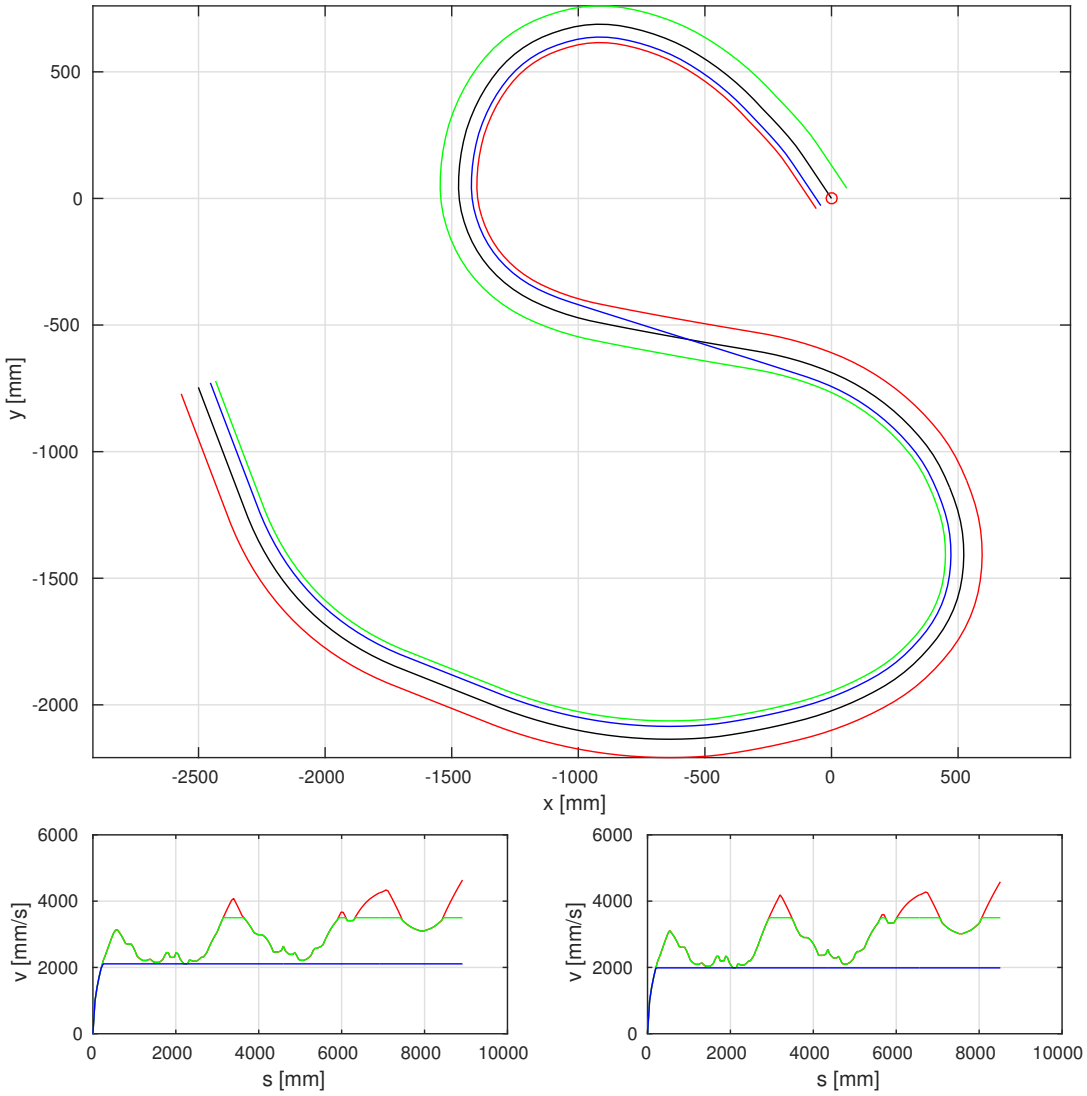
Uzupełnienie wyników optymalizacji

Rysunki B.1 – B.14 przedstawiają wyniki optymalizacji uzyskane w ramach prowadzonych analiz przykładowych tras i stanowią uzupełnienie wyników opisanych w podrozdziale 4.3.



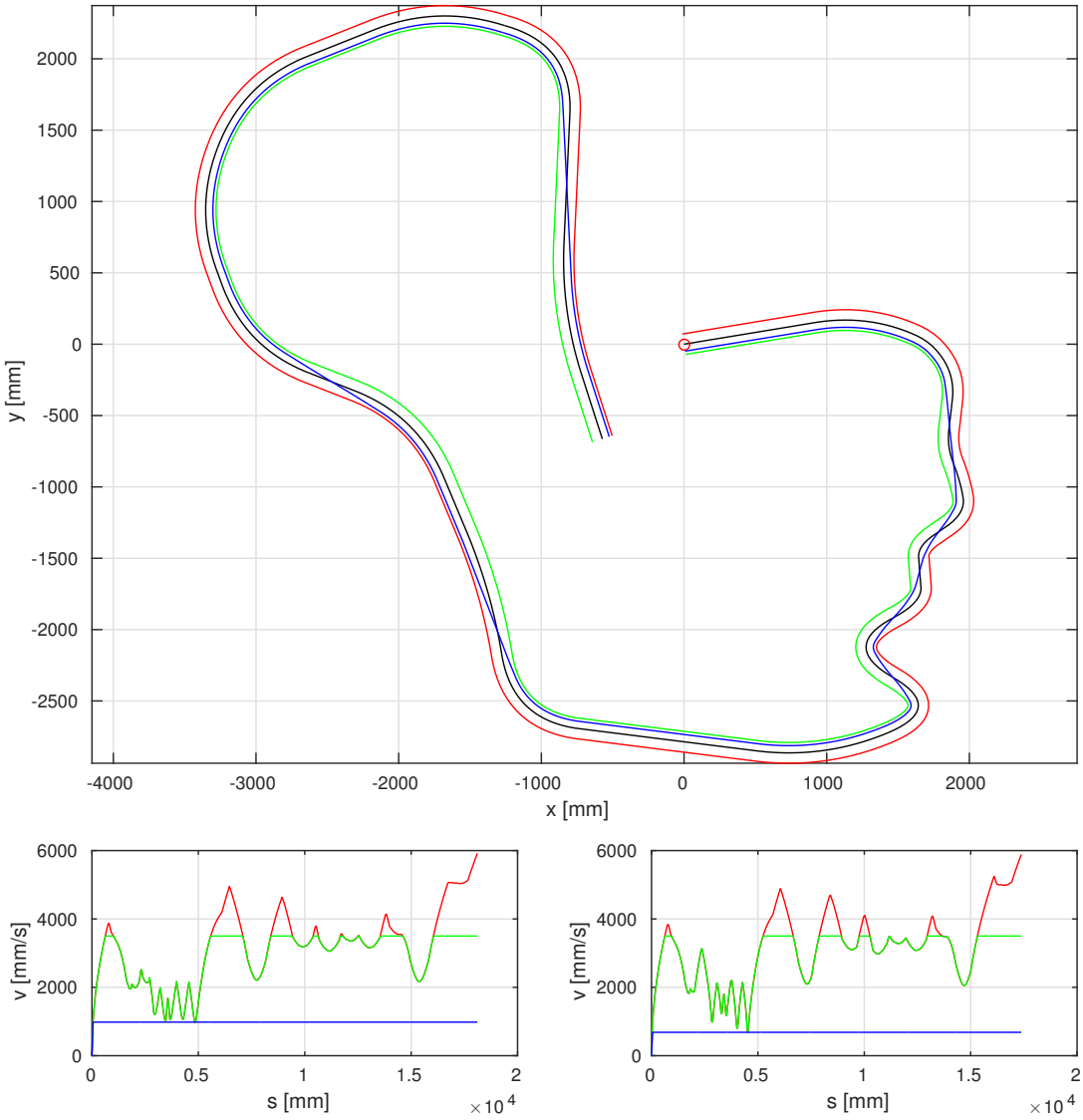
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	14.14	13.65	-3.4%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	1.41	1.22	-13.5%
Jazda zachowawcza – czas $[s]$	10.13	11.29	11.5%
Jazda optymalna – czas $[s]$	4.81	4.72	-1.8%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-52.5%	-58.1%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	5.19	5.09	-1.9%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	7.8%	7.7%	–

Rysunek B.1 Wyniki symulacji dla przykładowej trasy



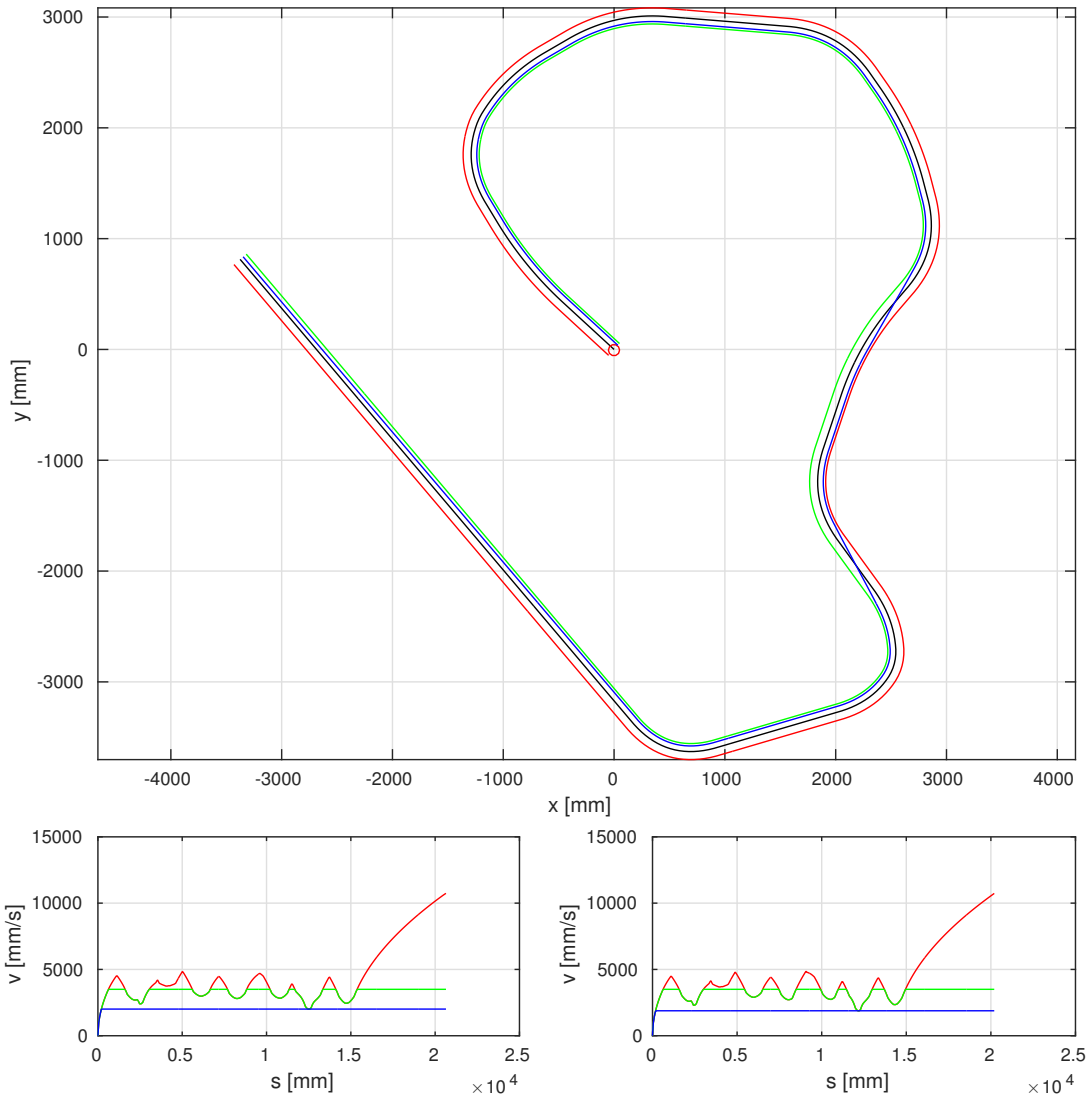
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	8.92	8.51	-4.5%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	2.11	1.99	-5.7%
Jazda zachowawcza – czas $[s]$	4.34	4.38	1.1%
Jazda optymalna – czas $[s]$	3.21	3.14	-2.2%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-26.0%	-28.4%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	3.28	3.21	-2.3%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	2.2%	2.1%	–

Rysunek B.2 Wyniki symulacji dla przykładowej trasy



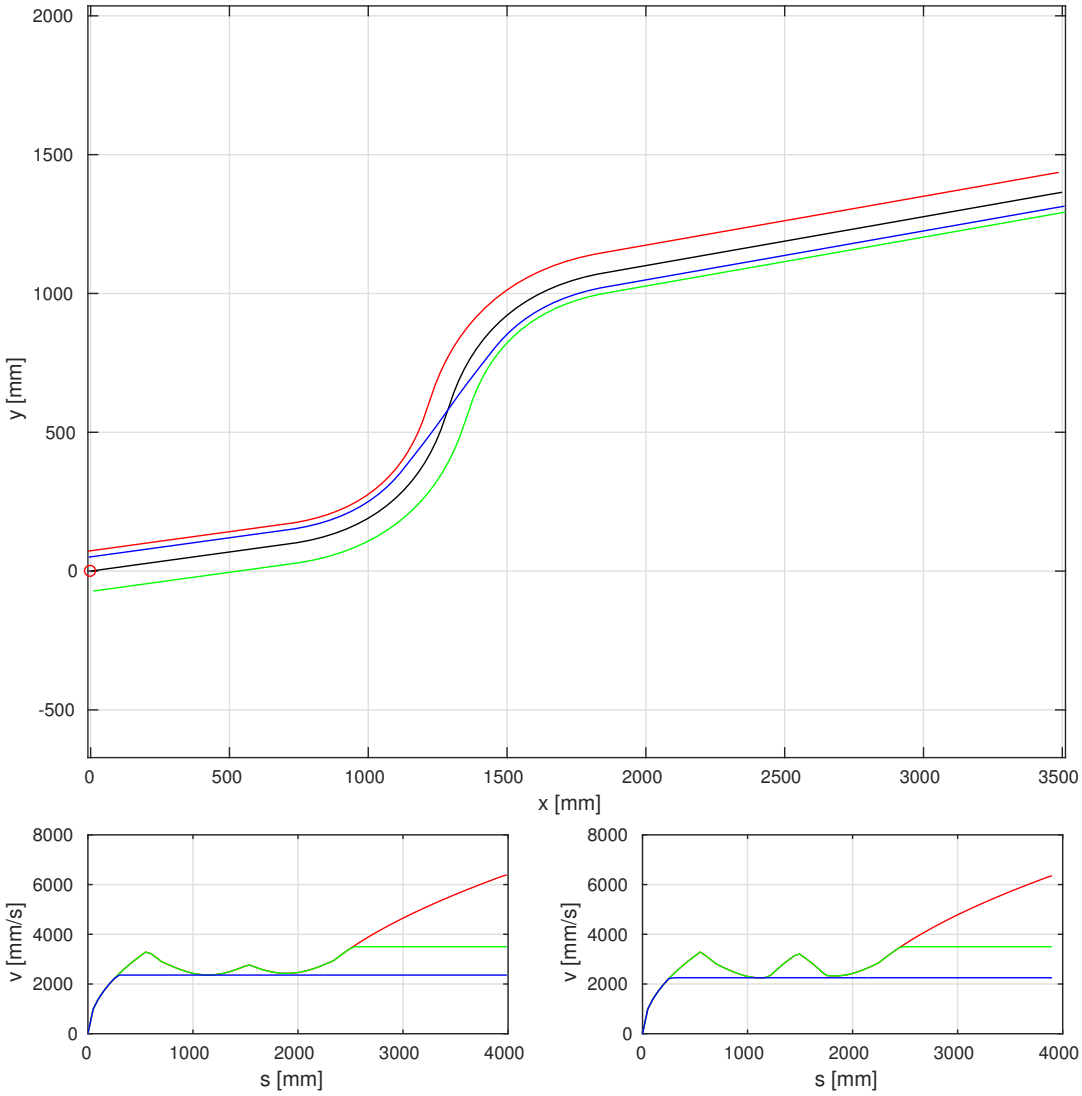
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	18.11	17.37	-4.1%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	0.98	0.68	-30.3%
Jazda zachowawcza – czas $[s]$	18.49	25.47	37.8%
Jazda optymalna – czas $[s]$	6.61	6.30	-4.7%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-64.2%	-75.3%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	6.92	6.60	-4.5%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	4.6%	4.8%	–

Rysunek B.3 Wyniki symulacji dla przykładowej trasy



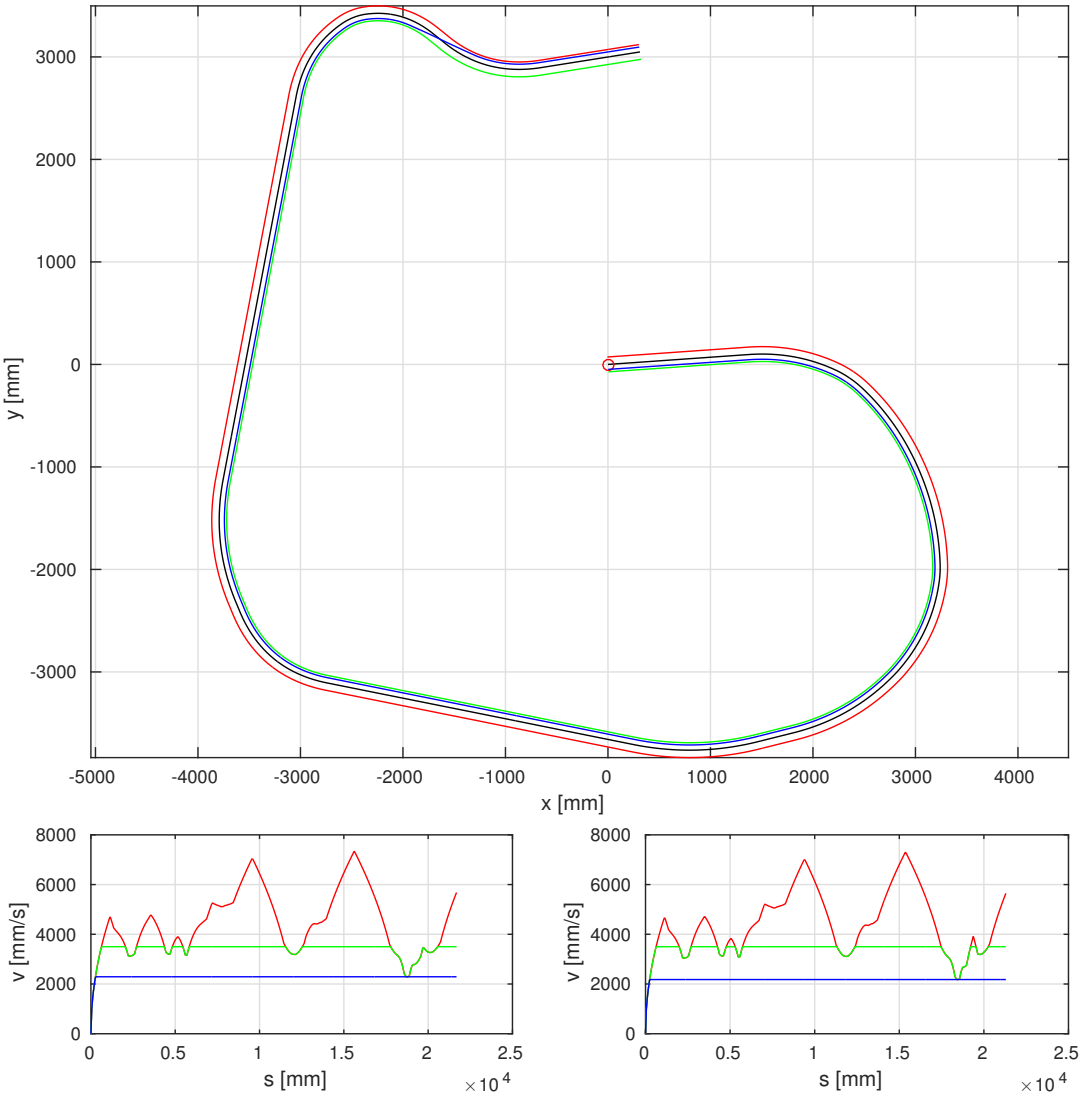
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	20.64	20.20	-2.1%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	2.01	1.88	-6.4%
Jazda zachowawcza – czas $[s]$	10.37	10.83	4.4%
Jazda optymalna – czas $[s]$	5.53	5.46	-1.3%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-46.7%	-49.6%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	6.57	6.49	-1.3%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	18.7%	18.7%	–

Rysunek B.4 Wyniki symulacji dla przykładowej trasy



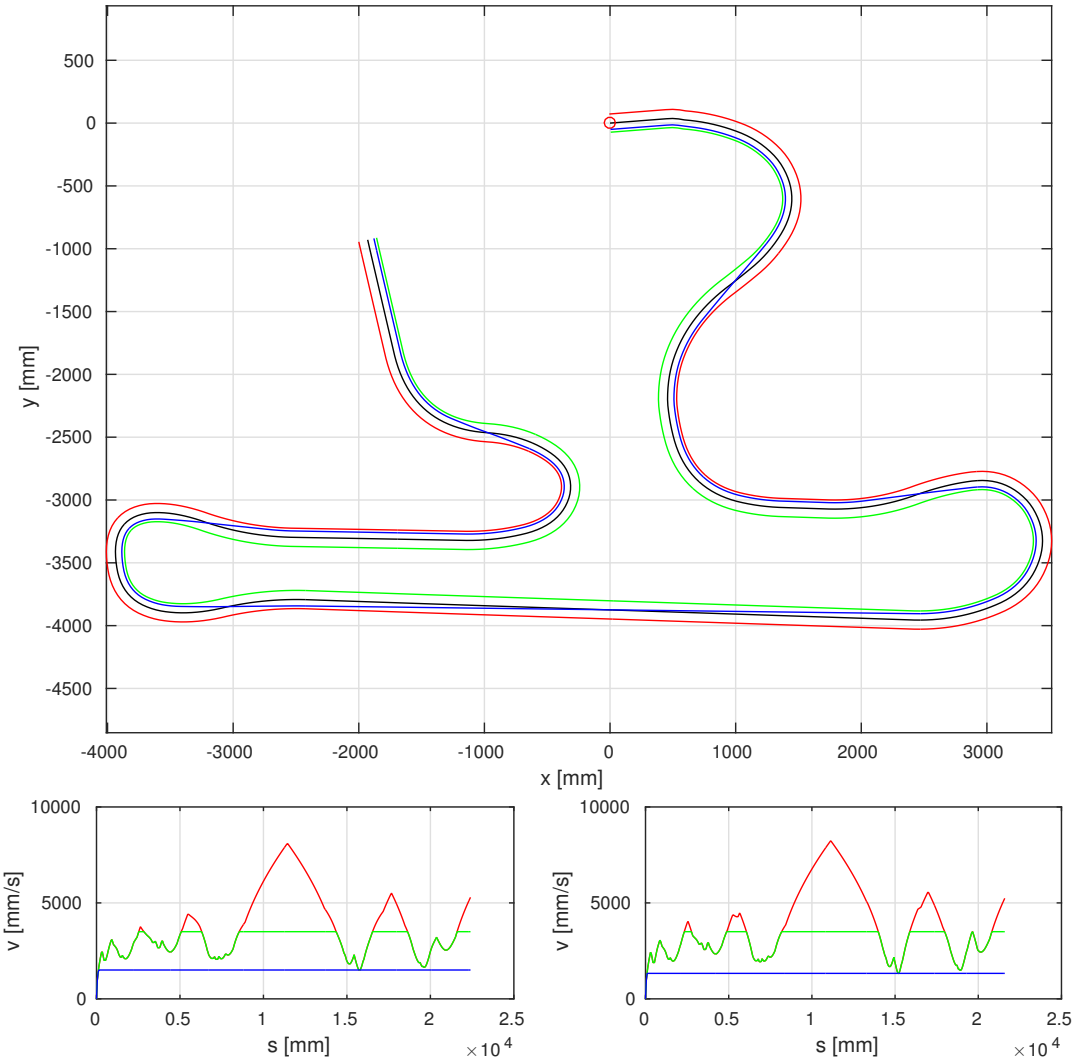
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	3.99	3.90	-2.3%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	2.36	2.25	-4.6%
Jazda zachowawcza – czas $[s]$	1.81	1.85	1.9%
Jazda optymalna – czas $[s]$	1.38	1.35	-2.1%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-23.9%	-26.9%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	1.50	1.47	-2.2%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	8.9%	8.8%	–

Rysunek B.5 Wyniki symulacji dla przykładowej trasy



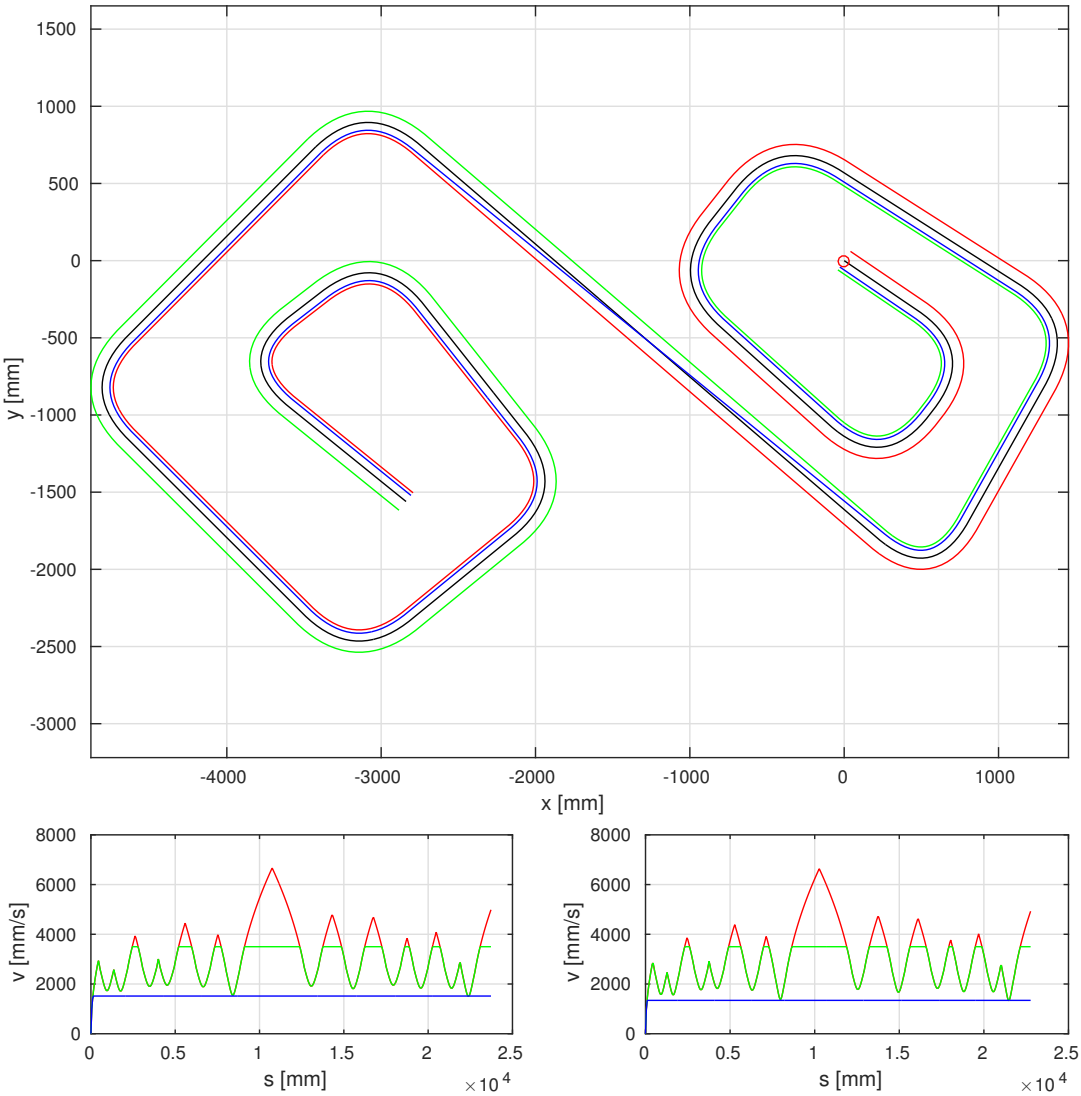
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	21.68	21.30	-1.8%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	2.29	2.18	-4.9%
Jazda zachowawcza – czas $[s]$	9.58	9.88	3.1%
Jazda optymalna – czas $[s]$	5.34	5.30	-0.9%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-44.2%	-46.4%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	6.56	6.48	-1.3%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	22.8%	22.3%	–

Rysunek B.6 Wyniki symulacji dla przykładowej trasy



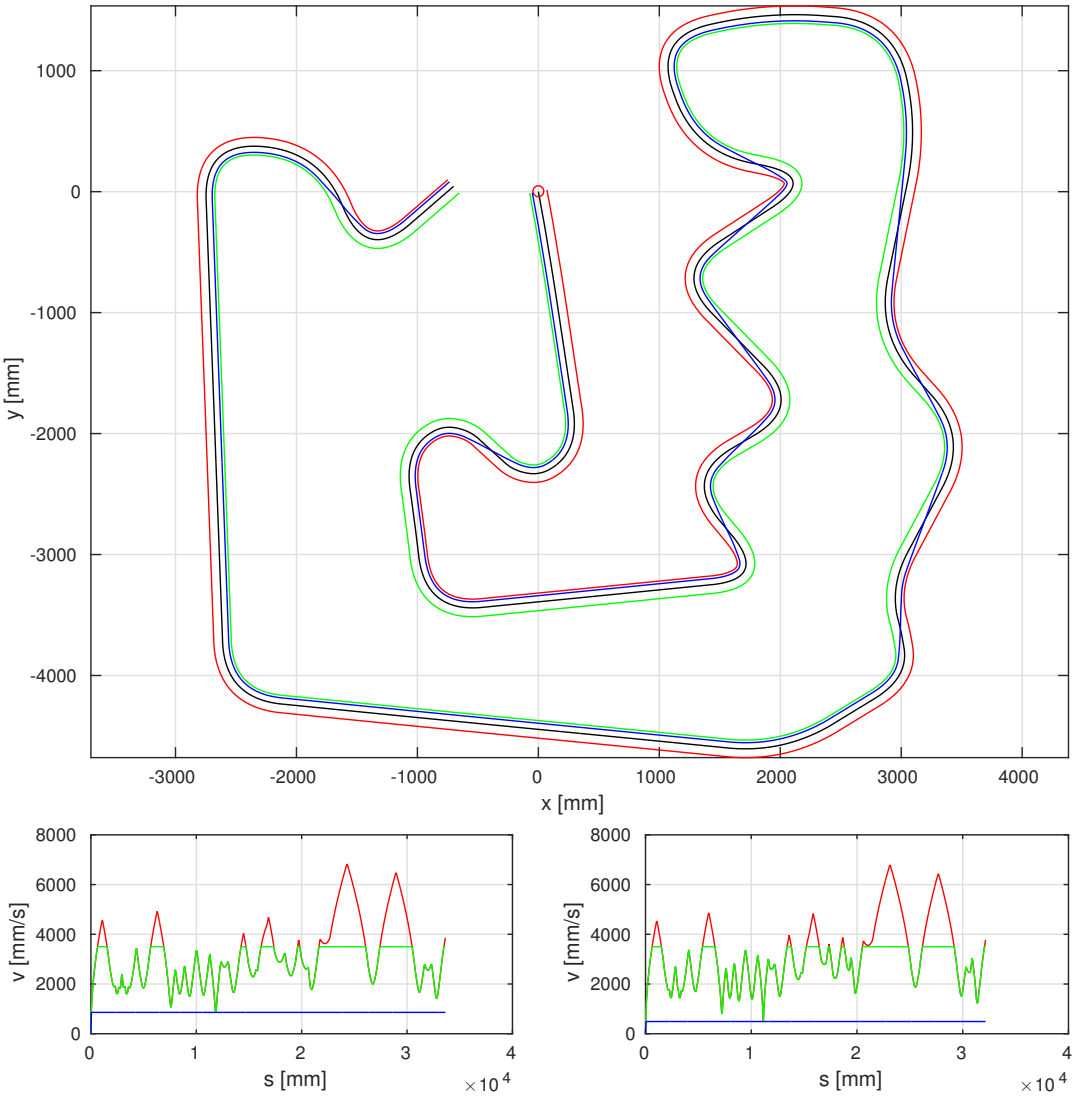
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	22.40	21.59	-3.6%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	1.50	1.33	-11.7%
Jazda zachowawcza – czas $[s]$	14.98	16.34	9.0%
Jazda optymalna – czas $[s]$	7.12	6.90	-3.0%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-52.5%	-57.7%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	7.97	7.80	-2.2%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	12.0%	13.0%	–

Rysunek B.7 Wyniki symulacji dla przykładowej trasy



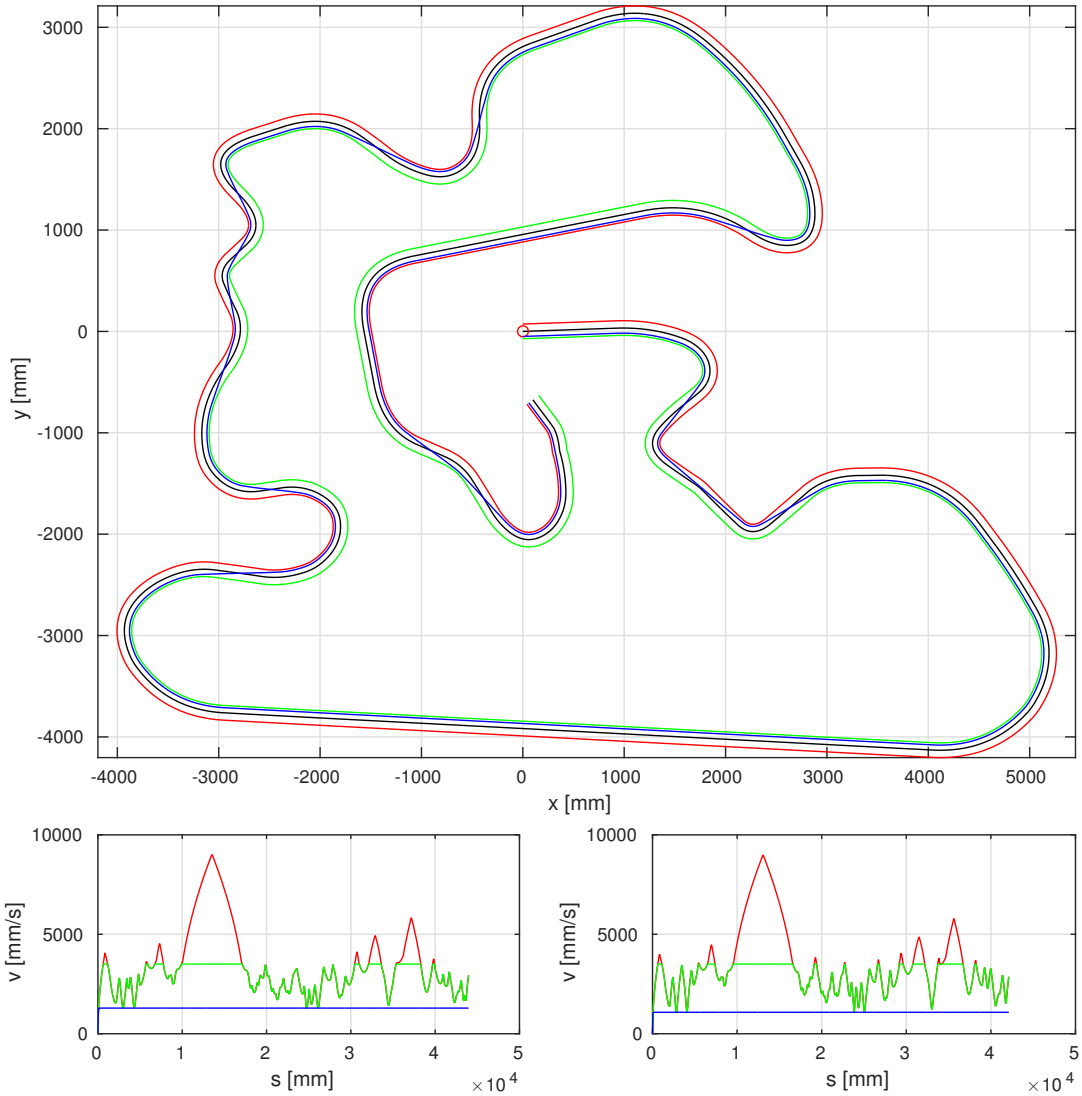
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	23.72	22.76	-4.1%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	1.52	1.34	-11.5%
Jazda zachowawcza – czas $[s]$	15.73	17.04	8.3%
Jazda optymalna – czas $[s]$	8.61	8.50	-1.3%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-45.3%	-50.1%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	9.08	8.95	-1.5%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	5.5%	5.2%	–

Rysunek B.8 Wyniki symulacji dla przykładowej trasy



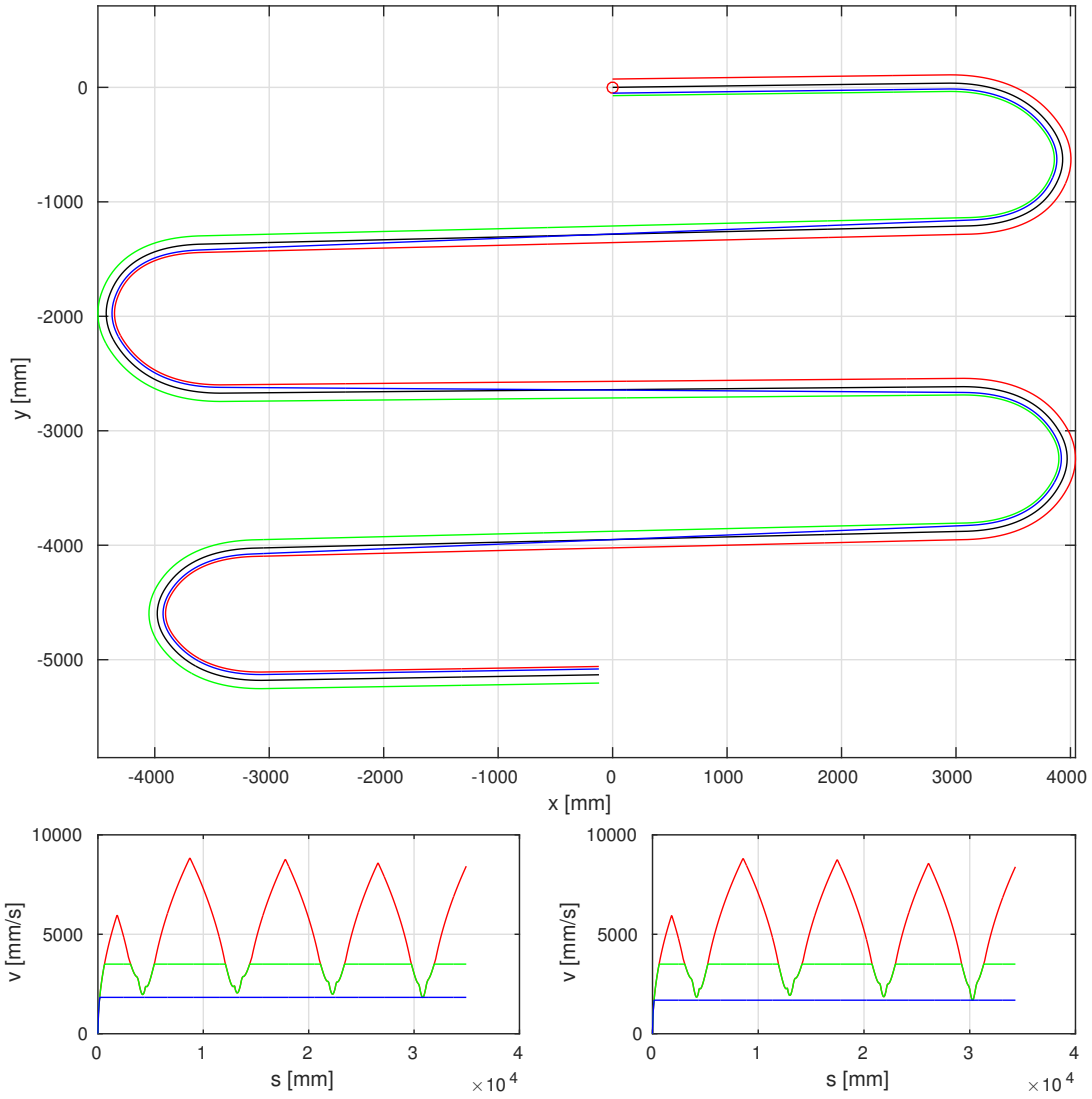
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	33.63	32.16	-4.4%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	0.86	0.49	-42.8%
Jazda zachowawcza – czas $[s]$	39.23	65.61	67.2%
Jazda optymalna – czas $[s]$	12.32	11.98	-2.8%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-68.6%	-81.7%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	13.09	12.72	-2.8%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	6.2%	6.2%	–

Rysunek B.9 Wyniki symulacji dla przykładowej trasy



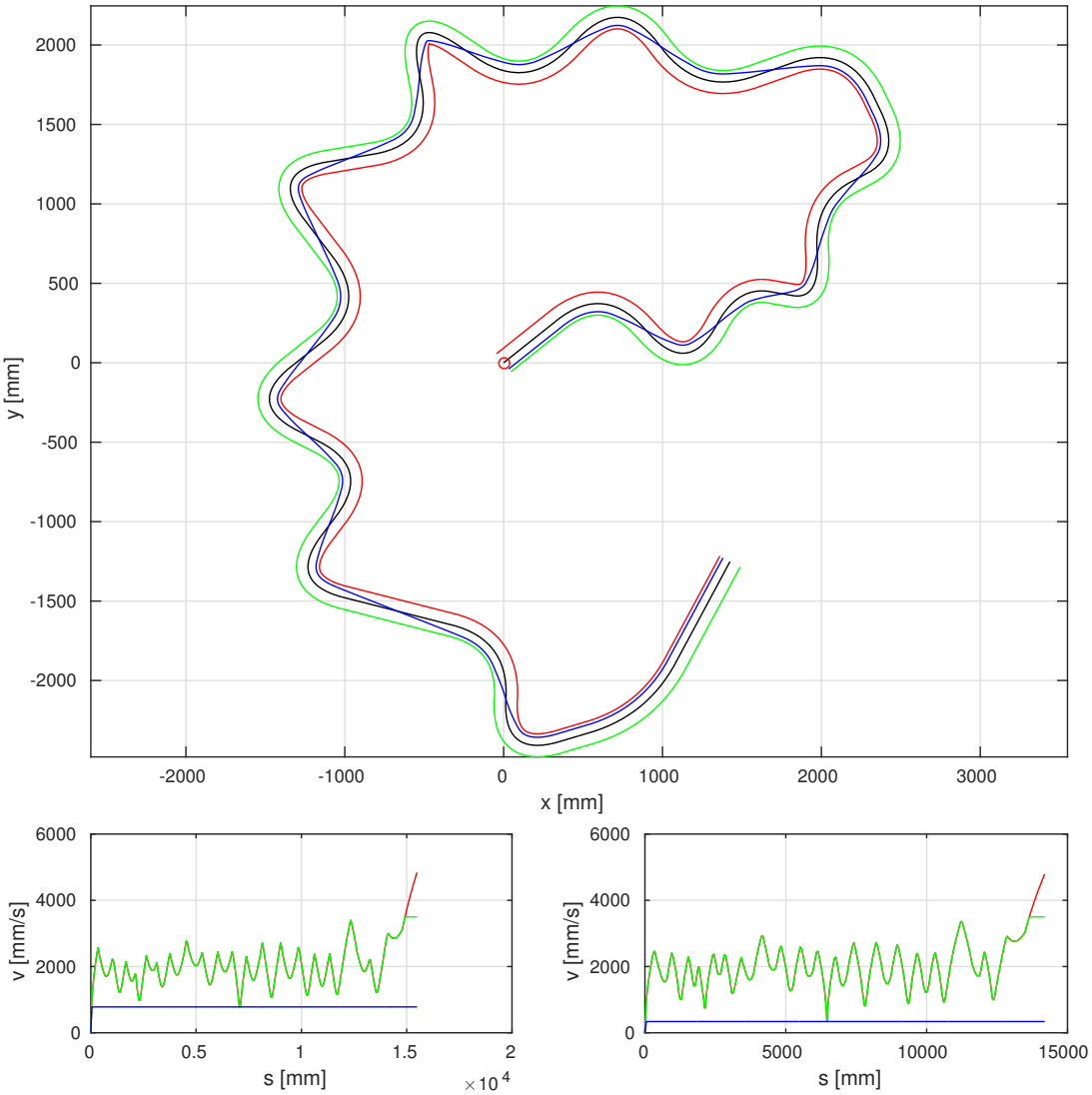
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	43.96	42.12	-4.2%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	1.29	1.08	-16.3%
Jazda zachowawcza – czas $[s]$	34.18	39.09	14.4%
Jazda optymalna – czas $[s]$	15.37	14.80	-3.7%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-55.0%	-62.1%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	16.57	15.97	-3.6%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	7.8%	7.9%	–

Rysunek B.10 Wyniki symulacji dla przykładowej trasy



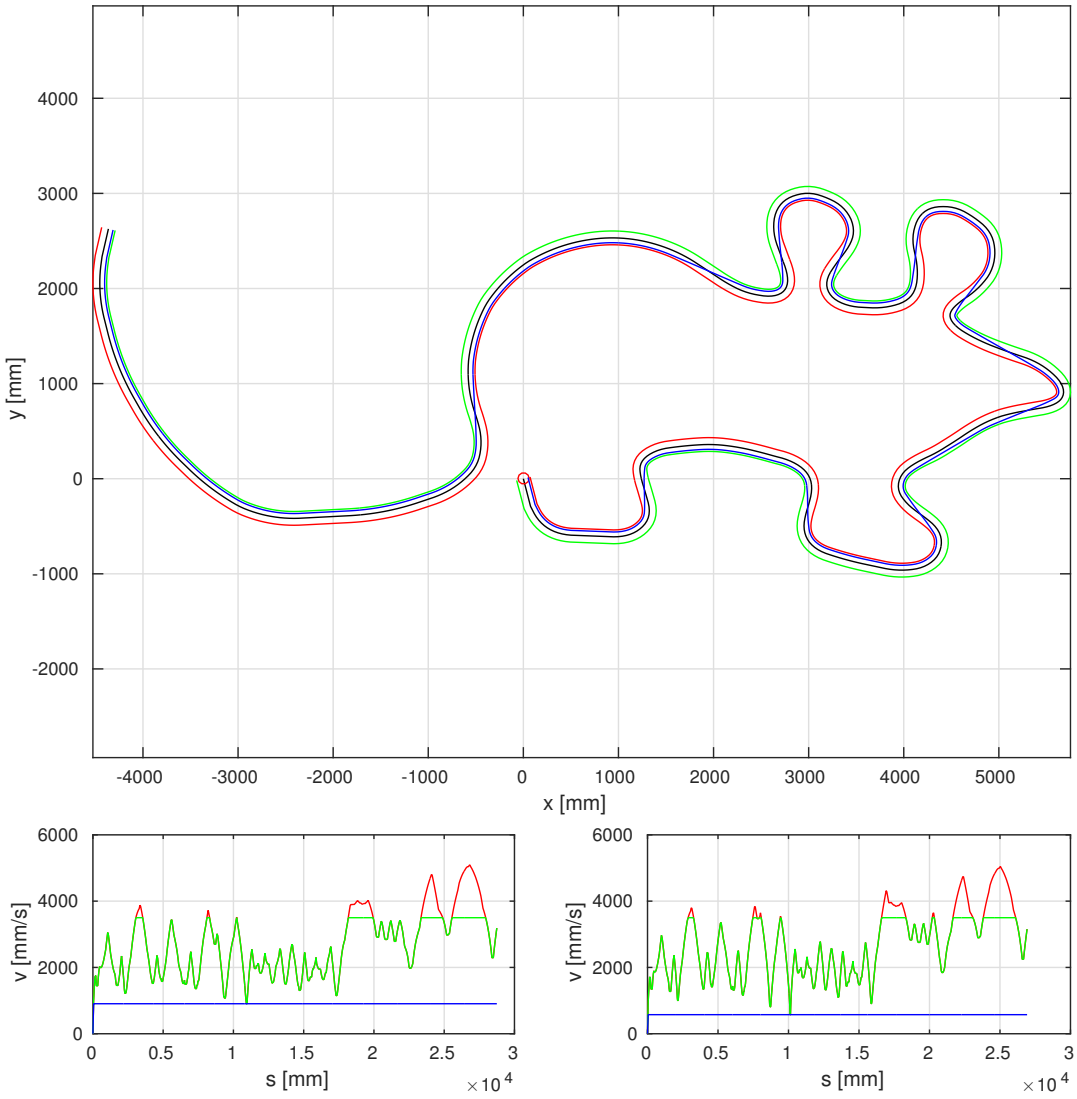
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	34.95	34.32	-1.8%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	1.83	1.69	-7.7%
Jazda zachowawcza – czas $[s]$	19.22	20.44	6.4%
Jazda optymalna – czas $[s]$	8.10	8.02	-1.0%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-57.8%	-60.8%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	11.04	10.93	-1.0%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	36.3%	36.3%	–

Rysunek B.11 Wyniki symulacji dla przykładowej trasy



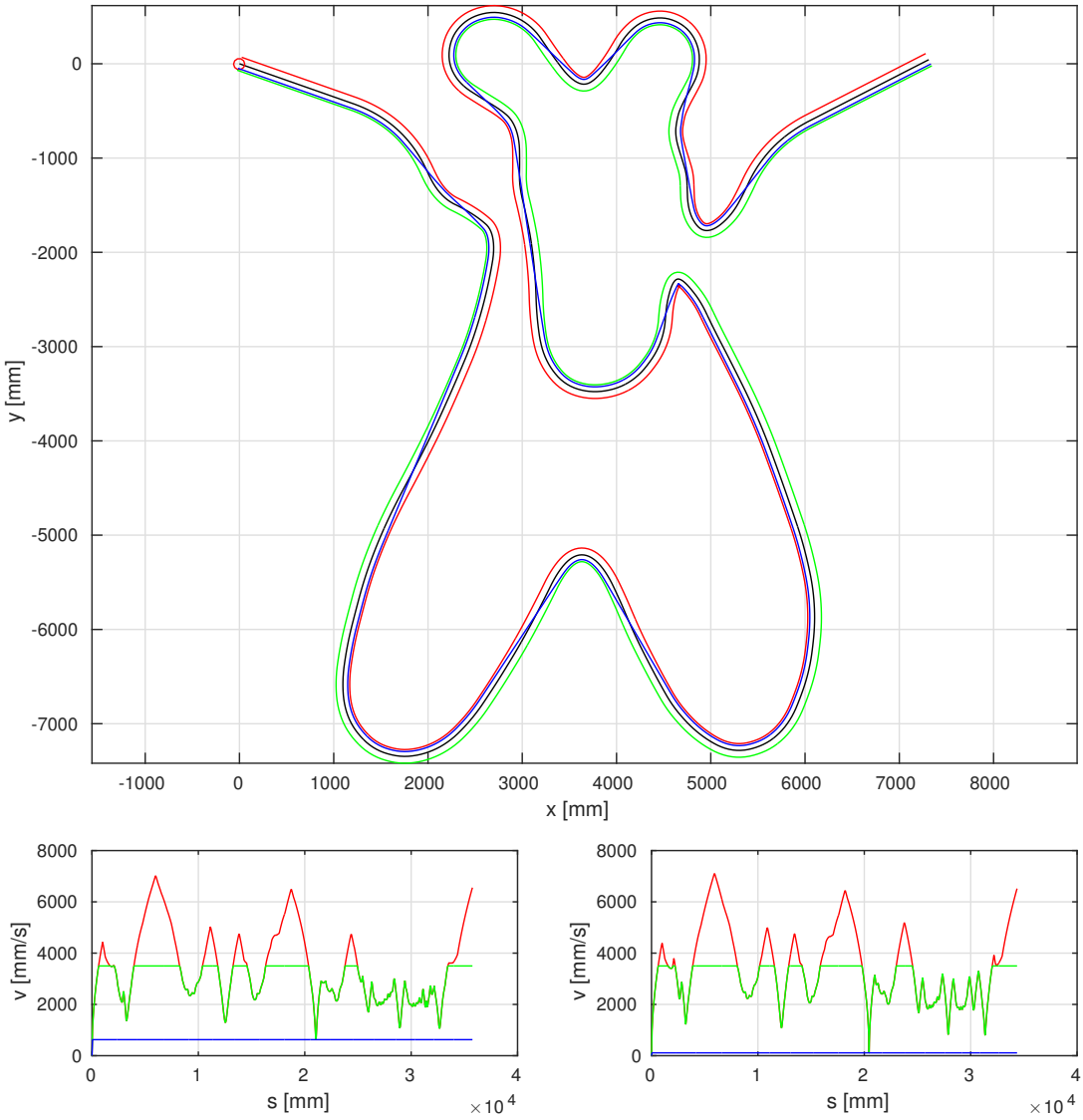
	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	15.49	14.19	-8.4%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	0.78	0.34	-56.5%
Jazda zachowawcza – czas $[s]$	19.90	41.93	110.7%
Jazda optymalna – czas $[s]$	8.21	7.75	-5.6%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-58.7%	-81.5%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	8.23	7.78	-5.6%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	0.3%	0.3%	–

Rysunek B.12 Wyniki symulacji dla przykładowej trasy



	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	28.74	26.92	-6.3%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	0.90	0.57	-36.5%
Jazda zachowawcza – czas $[s]$	31.82	46.92	47.4%
Jazda optymalna – czas $[s]$	12.14	11.69	-3.8%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-61.8%	-75.1%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	12.42	11.94	-3.8%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	2.2%	2.2%	–

Rysunek B.13 Wyniki symulacji dla przykładowej trasy



	Jazda dokładnie po trasie	Jazda ścieżką o minimalnej długości	Zmiana
Długość trasy $[m]$	35.76	34.36	-3.9%
Jazda zachowawcza – prędkość $[\frac{m}{s}]$	0.63	0.11	-82.2%
Jazda zachowawcza – czas $[s]$	57.22	308.40	438.9%
Jazda optymalna – czas $[s]$	12.21	11.86	-2.8%
Zmiana czasu przejazdu optymalnego w stosunku do zachowawczego	-78.7%	-96.2%	–
Jazda optymalna z ograniczeniem v_{max} – czas $[s]$	13.31	12.97	-2.6%
Zmiana czasu przejazdu optymalnego po dodaniu ograniczenia v_{max}	9.0%	9.3%	–

Rysunek B.14 Wyniki symulacji dla przykładowej trasy

Bibliografia

[1] Distance metrics. https://numerics.mathdotnet.com/distance.html. [online; dostęp 2 lutego 2017].

[2] Shortest path and minimum curvature path – implementation. http://math.stackexchange.com/questions/444289/shortest-path-and-minimum-curvature-path-implementation. [online; dostęp 24 stycznia 2017].

[3] Strona zawodów Robomaticon. http://www.robomaticon.pl/. [online; dostęp 24 stycznia 2017].

[4] Strona zawodów RobotChallenge. https://www.robotchallenge.org/. [online; dostęp 24 stycznia 2017].

[5] Strona zawodów Robotic Arena. http://www.roboticarena.pl/pl/. [online; dostęp 24 stycznia 2017].

[6] F. Alsaade, Y. Fouda. Template matching based on SAD and pyramid. *International Journal of Computer Science and Information Security*, 10(4):11–16, 2012.

[7] Baton. Algorytm linefollowera w C – dla początkujących i nie tylko. http://forbot.pl/blog/artykuly/podstawy/algorytm-linefollowera-c-poczatkujacych-id2722. [online; dostęp 24 listopada 2016].

[8] M. Botta, V. Gautieri, D. Loiacono, P. L. Lanzi. Evolving the optimal racing line in a high-end racing game. *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, strony 108–115. IEEE, 2012.

[9] F. Braghin, F. Cheli, S. Melzi, E. Sabbioni. Race driver model. *Computers & Structures*, 86(13):1503–1516, 2008.

[10] L. Cardamone, D. Loiacono, P. L. Lanzi, A. P. Bardelli. Searching for the optimal racing line using genetic algorithms. *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, strony 388–394. IEEE, 2010.

[11] Dondu. Robotyka: Linefollower – podążanie za linią. http://mikrokontrolery.blogspot.com/2011/03/Robotyka-Linefollower-Podazanie-za-linia.html. [online; dostęp 24 listopada 2016].

[12] Y. Fouda, A. R. Khan. Normalize cross correlation algorithm in pattern matching based on 1-D information vector. *Trends in Applied Sciences Research*, 10(4):195, 2015.

[13] D. Hale. An efficient method for computing local cross-correlations of multi-dimensional signals. *CWP Report*, 656, 2006.

[14] J. O. Smith III. Spectral audio signal processing. https://ccrma.stanford.edu/~jos/sasp/Quadratic_Interpolation_Spectral_Peaks.html. [online; dostęp 22 listopada 2016].

[15] Kingbright. KTIR0711S Datasheet. http://www.tme.eu/pl/Document/3755fa2108a850a17e4cb6b83e1f65fe/KTIR0711S.PDF. [online; dostęp 24 stycznia 2017].

[16] Koło Naukowe Robotyków "KoNaR". Regulamin zawodów robotów, katerogia LineFollower light. http://roboticarena.pl/static/Main/regulaminy/LF_Light.pdf. [online; dostęp 24 stycznia 2017].

[17] A. Kumar, A. Joshi, A. Kumar, A. Mittal, D. Gangodkar. Template matching application in geo-referencing of remote sensing temporal image. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 7(2):201–210, 2014.

[18] S. W. Smith. The scientist and engineer's guide to digital signal processing. http://www.dspguide.com/CH7.PDF. [online; dostęp 22 listopada 2016].

[19] J. Stolarz. Optymalizacja trajektorii przejazdu trasy robota klasy linefollower. Praca dyplomowa inżynierska, Politechnika Wrocławska, 2015.

[20] Treker. PID w robotyce amatorskiej - linefollower, cz. 1 człon P. http://www.forbot.pl/forum/topics20/algorytmy-pid-w-robotyce-amatorskiej-linefollower-cz1-czlon-p-vt5973.htm. [online; dostęp 24 listopada 2016].

[21] E. Velenis, P. Tsiotras. Optimal velocity profile generation for given acceleration limits: Receding horizon implementation. *American Control Conference, 2005. Proceedings of the 2005*, strony 2147–2152. IEEE, 2005.

[22] E. Velenis, P. Tsiotras. Optimal velocity profile generation for given acceleration limits: Theoretical analysis. *American Control Conference, 2005. Proceedings of the 2005*, strony 1478–1483. IEEE, 2005.