POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

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

PRACA DYPLOMOWA MAGISTERSKA

Optymalizacja toru ruchu robota klasy linefollower

Path optimization for linefollower robots

AUTOR:
Rafał Cymiński

PROWADZĄCY PRACĘ:
Dr inż. Robert Muszyński

OCENA PRACY:

WROCŁAW 2018

Spis treści

Rozdział 1

Wstęp

Robot klasy linefollower to konstrukcja autonomiczna, której zadaniem jest pokonanie trasy wyznaczonej przy pomocy czarnej linii na kontrastującym podłożu. Zasady rozgrywania konkurencji oraz specyfikacja trasy i robota ściśle określa regulamin zawodów [1, 2]. Obrys robota musi mieścić się na kartce papieru formatu A4. Waga i wysokość jest nieograniczona. Komunikacja z robotem w czasie rozgrywki jest zabroniona z wyjątkiem zdalnego uruchomienia i zatrzymania robota. Linia wyznaczająca trasę ma szerokość od 15 do 20 mm. Tor może posiadać zakręty o kącie mniejszym bądź równym kątowi prostemu oraz linie krzyżujące się pod kątem prostym. Na skrzyżowaniu robot powinien wybrać drogę na wprost. Dwie sąsiadujące ze sobą linie nie powinny przebiegać bliżej niż 150 mm od siebie. Zasadniczo zabronione jest opuszczanie toru oraz skracanie, omijanie jej fragmentów. Do opuszczenia trasy dochodzi, gdy robot swoim obrysem nie pokrywa linii wyznaczającej tor jazdy. Regulamin może zezwalać na opuszczenie trasy pod warunkiem powrotu na nią w miejscu opuszczenia lub punkcie poprzedzającym to miejsce.

Klasyczny robot startujący w konkurencji linefollower to platforma mobilna klasy (2,0). Posiada dwa koła z napędem różnicowym stanowiące dwa punkty podparcia. Robot klasy (2,0) charakteryzuje się dużą zwrotnością – potrafi wykonać obrót w miejscu, dlatego sterowanie taką platformą jest znacznie prostsze niż robotami innych klas. Zazwyczaj jest to konstrukcja modułowa składająca się z modułu głównego oraz modułu czujników linii. Takie rozwiązanie pozwala na regulację wysunięcia czujników i dostosowanie robota do poziomu trudności trasy. Dla zapewnienia stabilności potrzebny jest trzeci punkt podparcia. Najczęściej stanowi go moduł czujników linii.

Moduł główny zawiera kluczowe elementy robota potrzebne do jego działania, czyli napęd (silniki i koła), układ zasilający (akumulator, stabilizatory napięcia i włącznik zasilania) i sterujący (mikrokontroler, sterowniki silników, enkodery).

Istotnym aspektem, który ma wpływ na osiągane rezultaty, jest przyczepność kół. Od niej zależy maksymalna prędkość przejazdu, która jest zwykle dostosowywana do najtrudniejszego fragmentu trasy. Przy założeniu stałej prędkości postępowej robota, siła odśrodkowa działająca na konstrukcję rośnie wraz ze wzrostem krzywizny zakrętu. Jeśli ta siła jest zbyt duża dochodzi do utraty przyczepności kół a w konsekwencji nawet do wypadnięcia z trasy. Jednakże, ryzyko przejazdu z coraz większą prędkością, które podejmują zawodnicy może przynieść wymierny efekt w postaci poprawy rezultatu.

Częstokroć konstruktorzy montują dodatkowo w module głównym napęd tunelowy popularnie nazywaną turbiną[1], czyli wysokoobrotowy silnik elektryczny z przymocowanymi do wirnika łopatkami. Zwiększa on docisk robota do podłoża siłą odrzutu powietrza i skut-

[1]Nazwa jest błędna, gdyż turbina jest urządzeniem generującym energię. Z kolei napęd tunelowy zużywa energię elektryczną do wprawienia wirnika w ruch.

kuje poprawą przyczepności. W efekcie robot – w porównaniu do konstrukcji pozbawionej napędu tunelowego – potrafi przyspieszyć do prędkości maksymalnej oraz wyhamować na znacznie krótszym odcinku, a także może szybciej pokonywać zakręty. Zysk wynikający z zastosowania go jest na tyle duży, że organizatorzy zawodów zazwyczaj decydują się na podział konkurencji linefollower na roboty wyposażone w napęd tunelowy (konkurencja Linefollower Turbo) oraz nie posiadające go (Linefollower Standard lub Linefollower Light).

Kolejnym ulepszeniem jest wyposażenie robota w enkodery. Najczęściej są wykorzystywane enkodery magnetyczne, informujące o obrocie magnesu zamocowanego na kole lub wale silnika. Przy zastosowaniu dużej częstotliwości próbkowania wartość kąta obrotu może posłużyć do wyznaczenia chwilowej prędkości koła, niezbędnej w nawigacji zliczeniowej. Obecnie enkodery wykorzystuje się w pętli regulacji do sterowania rzeczywistą prędkością kół robota, co przekłada się na minimalizację poślizgów oraz precyzyjną jazdę po linii. Takie sterowanie jest odporne na zmianę zasilania w trakcie przejazdu, która wynika z rozładowywania akumulatora i objawia się spadkiem napięcia.

Zmianę położenia robota można wyznaczyć stosując nawigację zliczeniową. Metoda ta polega na ustaleniu przemieszczenia na podstawie translacji i rotacji wyliczanej z aktualnej prędkości obu kół.

W module czujników do wykrywania linii zwykle stosuje się od 8 do 16 czujników. Większa liczba sensorów pozwala na szerszy przegląd otoczenia robota oraz bardziej precyzyjne oszacowanie położenia względem linii, jednakże wymaga większych zasobów sprzętowych do ich obsługi.

O sukcesie i jakości robota decyduje jego mechanika i zastosowane algorytmy sterowania. Obecnie wiele robotów klasy linefollower charakteryzuje się niemal identycznymi rozwiązaniami konstrukcyjnymi, które zostały wypracowane na przestrzeni lat, co świadczy o wysokim poziomie mechaniki. Niestety w większości z nich stosowane są proste algorytmy sterowania, korygujące położenie robota względem linii jedynie na bazie aktualnej informacji pochodzącej z czujników linii. Takie podejście wymaga potraktowania maksymalnej prędkości jazdy robota jako parametru zadawanego a priori o wartości dobieranej przed przejazdem tak, by robot był w stanie przejechać najciaśniejszy fragment trasy. To rozwiązanie jest nieefektywne, gdyż nie pozwala wykorzystać pełnych możliwości robota – potrzebny jest przynajmniej moduł, który pozwoliłby dostosować prędkość przed zbliżającym się zakrętem i zapewnił utrzymanie robota na trasie.

W pracy [7] zaproponowano algorytm, który w ramach przejazdu testowego sporządza mapę. Następnie wylicza on krzywiznę trasy i na jej podstawie dynamicznie oblicza maksymalną prędkość, z jaką może poruszać się robot na danym odcinku, która zapewnia utrzymanie się na torze. Algorytm efektywnie wykorzystuje długie proste zwiększając na nich prędkość maksymalną oraz zmniejsza prędkość przed ostrymi zakrętami, przez co sumarycznie uzyskuje lepsze czasy przejazdu.

Dalszą poprawę rezultatów można uzyskać optymalizując kształt rzeczywistego toru przejazdu tak, by nie zjechać z linii a równocześnie skrócić jego długość czy zmniejszyć krzywiznę. Optymalna ścieżka definiuje trajektorię, po której należy poruszać się, aby uzyskać najlepszy możliwy czas okrążenia dla danej trasy i pojazdu. Zależy on od kilku czynników, takich jak kształt toru, moc silników czy przyczepność kół. Algorytmy poszukiwania optymalnej ścieżki są stosowane w sportach motorowych [3], jak również w grach i symulatorach wyścigowych. Popularnym środowiskiem symulacyjnym jest TORCS – The Open Racing Car Simulator [4], oferujący trójwymiarowe symulacje wyścigów samochodowych, wykorzystywany przez naukowców do badań i testowania algorytmów sterowania i optymalizacji toru ruchu. Przykład badań przy użyciu symulatora TORCS zaprezento-

wano w pracy [5].

Reasumując, by efektywnie sterować robotem klasy linefollower w celu przejechania trasy należy:

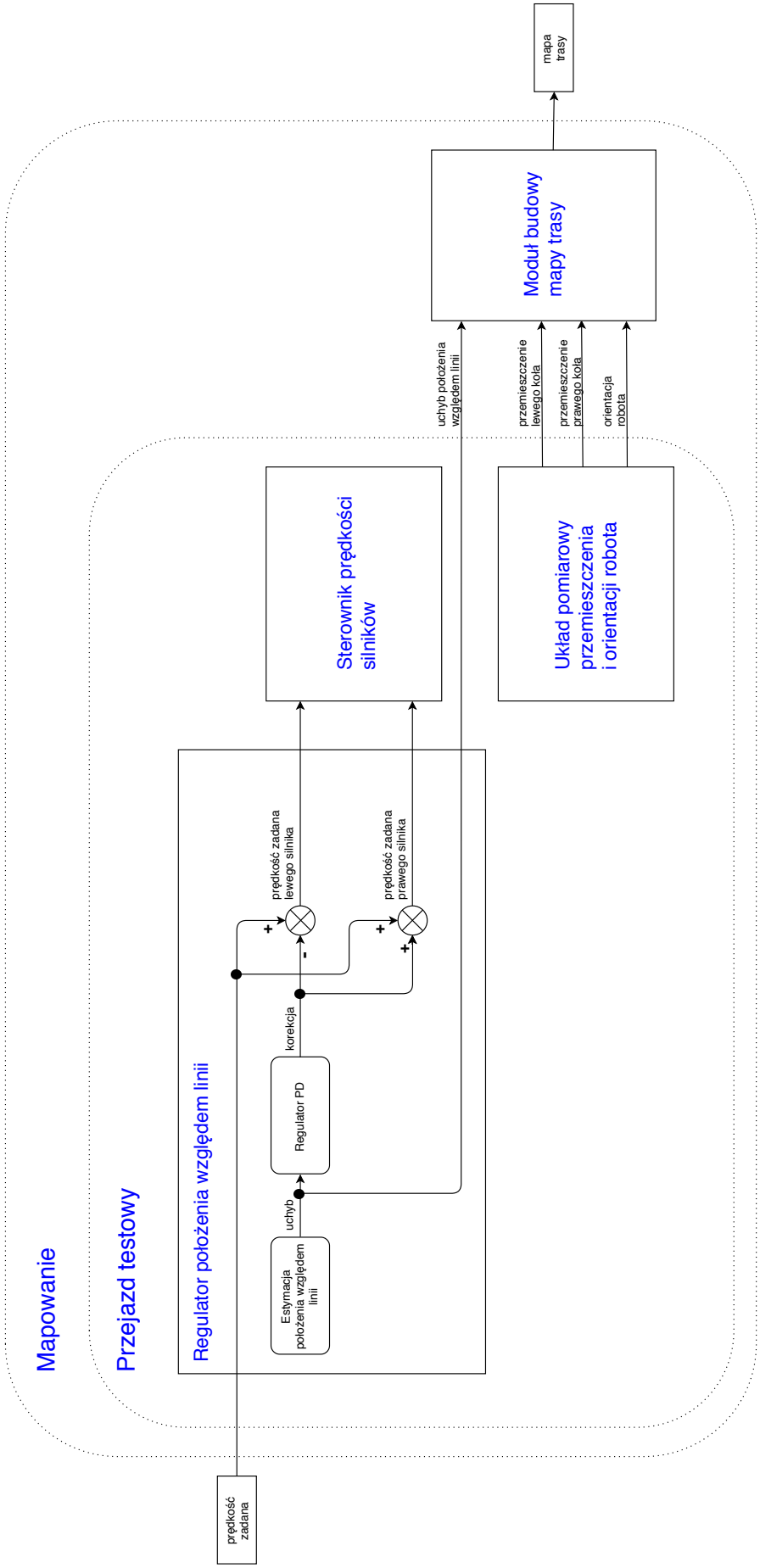
- w ramach przejazdu testowego sporządzić mapę trasy,
- zoptymalizować kształt trasy z uwzględnieniem własności dynamicznych robota,
- zaprojektować profil prędkości przejazdu trasy uzyskując pożądaną trajektorię ruchu robota,
- zrealizować właściwy przejazd według zaprojektowanej trajektorii pożądanej ze sprzężeniem zwrotnym od czujników linii.

W przypadku robota klasy (2,0) wyposażonego w enkodery oraz moduł do nawigacji inercyjnej IMU opisane powyżej etapy mogą zostać zrealizowane w następujący sposób. W trakcie przejazdu testowego z zadaną prędkością postępową, robot podąża za linią definiującą trajektorię. Dane uzyskane w trakcie przejazdu są przetwarzane na mapę trasy. Diagram funkcyjny procesu mapowania pokazano na rysunku 1.1. Optymalizacja kształtu trasy dokonywana jest zgodnie z wybranym algorytmem. Rezultatem jest optymalna ścieżka, która przekształcana jest w trajektorię na podstawie zaprojektowanego profilu prędkości przejazdu właściwego. Na rysunku 1.2 zaprezentowano diagram funkcyjny optymalizacji. Przejazd właściwy polega na śledzeniu otrzymanej w opisany sposób trajektorii. Do korekty położenia i prędkości robota zastosowano sterownik kinematyczny. Diagram funkcyjny dla tego etapu pokazano na rysunku 1.3. Zarówno w trakcie przejazdu testowego jak i właściwego używany jest układ pomiarowy do wyznaczania przemieszczenia i orientacji robota, oraz sterownik prędkości kół opisane w podrozdziałach 2.1 i 2.3.

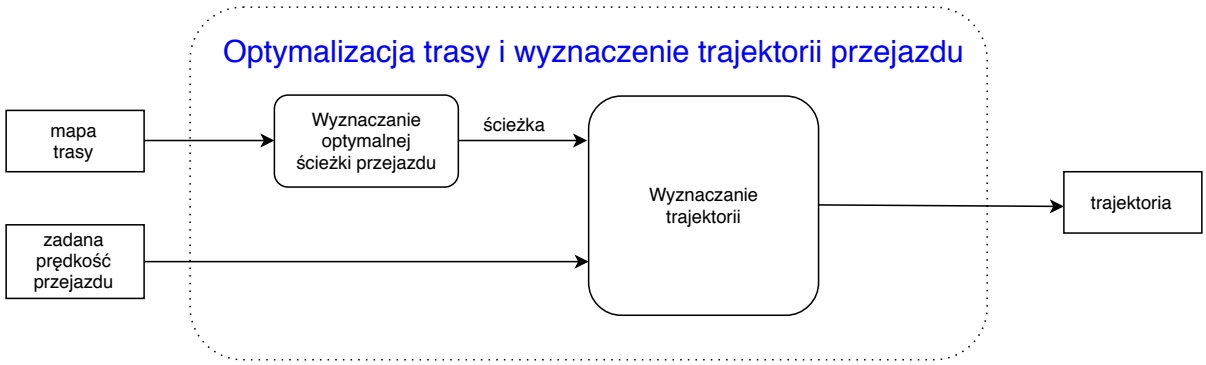
Celem pracy jest przegląd literatury i wybór algorytmów optymalizacji toru ruchu, które można zastosować w robotach klasy linefollower. Zadania do wykonania to:

- przygotowanie środowiska badania własności algorytmów,
- implementacja wskazanych algorytmów,
- porównanie własności zaimplementowanych algorytmów.

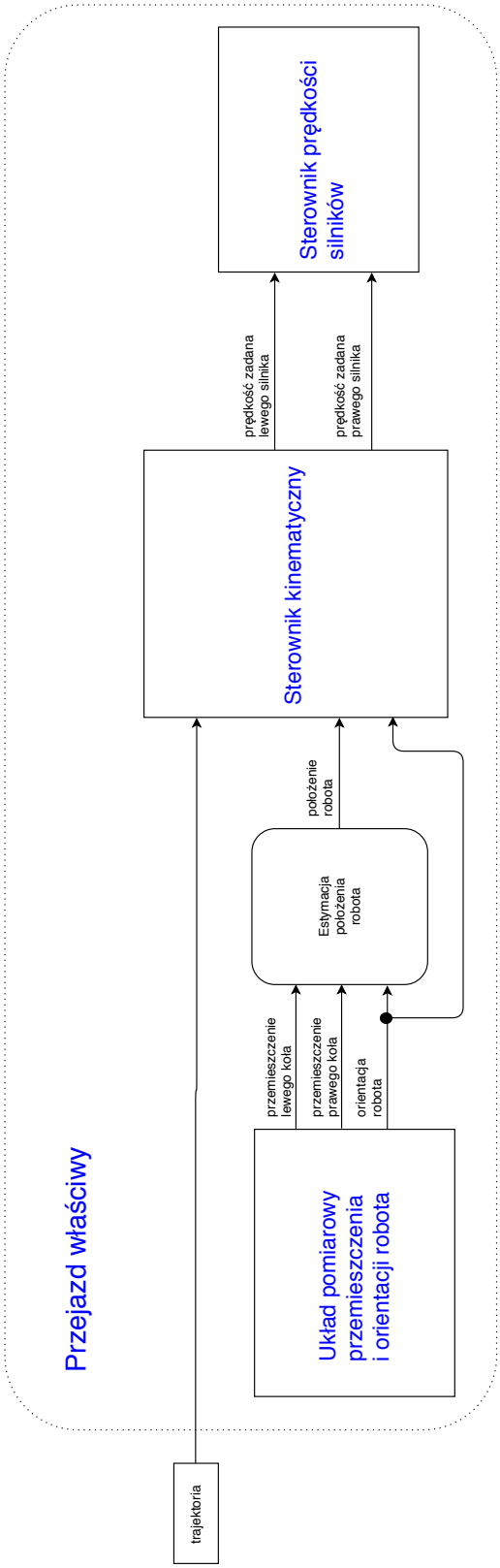
Układ pracy jest następujący. W rozdziale 2 zaprezentowano robota klasy linefollower, którego wykorzystano do przeprowadzenia eksperymentów. Rozdział 3 zawiera opis metod reprezentacji trasy. Rozdział 4 poświęcono mapowaniu i przygotowaniu mapy trasy do procesu optymalizacji. W rozdziale 5 omówiono algorytmy wyznaczania optymalnej ścieżki. Rozdział 6 traktuje o algorytmach sterowania robotem dla zadania przejazdu testowego, w którym wykonywane jest mapowanie, oraz przejazdu właściwego po optymalnej trajektorii. W rozdziale 7 przedstawiono wyniki przeprowadzonych symulacji i eksperymentów. Rozdział 8 podsumowuje wykonaną pracę i uzyskane rezultaty oraz zawiera propozycje dalszego postępowania.



Rysunek 1.1: Diagram funkcyjny procesu mapowania



Rysunek 1.2: Diagram funkcyjny procesu optymalizacji



Rysunek 1.3: Diagram funkcyjny właściwego przejazdu

Rozdział 2

Robot klasy linefollower

Jak wspomniano we wstępie, najpopularniejszym rozwiązaniem konstrukcyjnym stosowanym przez zawodników startujących w konkurencji linefollower jest wyposażenie robota w dwa silniki o napędzie różnicowym. Taką konstrukcję zalicza się do klasy (2,0) robotów mobilnych. Identyczne rozwiązanie zastosowano w robocie do badań w ramach tej pracy (zobacz rysunek 2.1). Poniżej opisano kolejno konstrukcję robota, jego kinematykę oraz sposób wyznaczania położenia i orientacji robota.

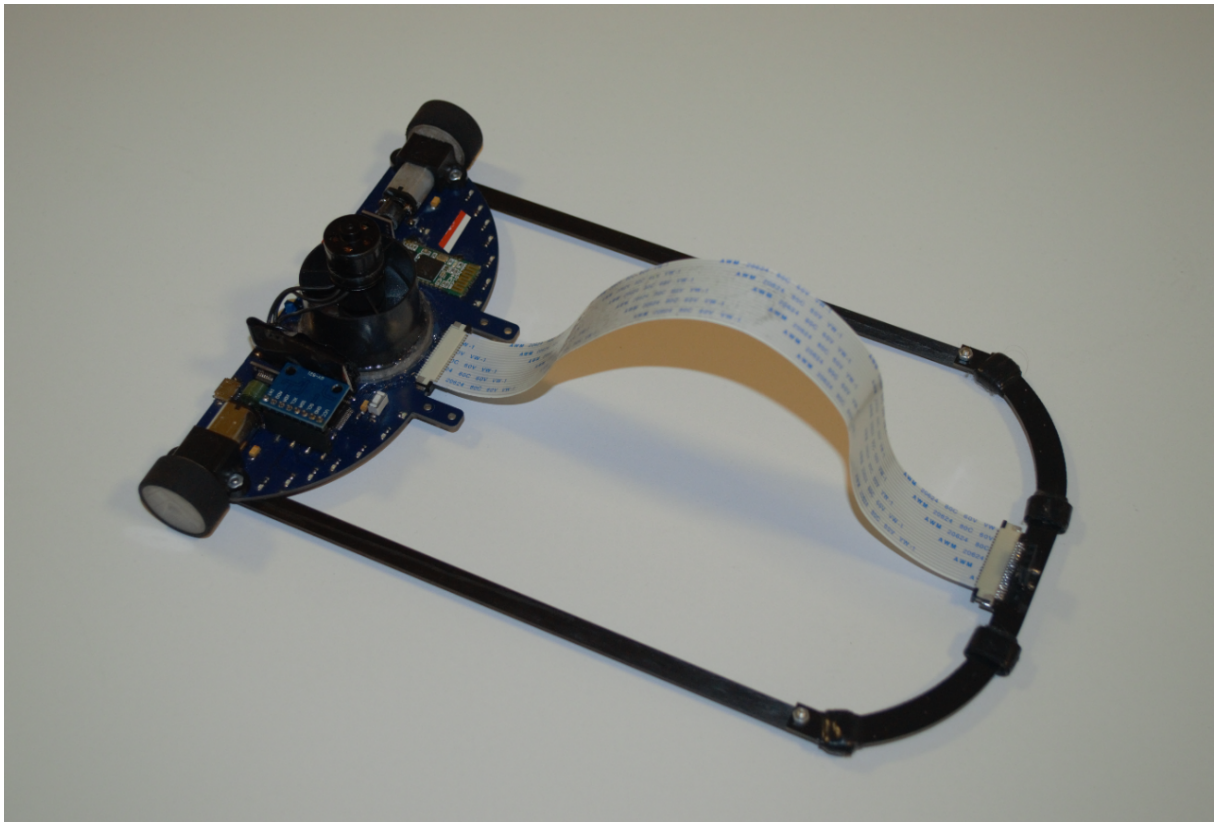
2.1 Konstrukcja robota

Konstrukcja składa się z dwóch modułów – modułu głównego i modułu czujników połączonych listwami węglowymi zapewniającymi sztywne łączenie mechaniczne oraz taśmą FFC, stanowiącą połączenie elektryczne. Zewnętrzne wymiary robota to szerokość $L = 16\ cm$ i 24 cm długości mieszczące się w dopuszczalnych wymiarach kartki formatu A4, czyli 21 cm na 29.7 cm. Moduł czujników jest wysunięty do przodu na odległość $l = 215mm$ względem środka osi robota. Podstawowe wymiary zaprezentowano na rysunku 2.2. Zasilanie stanowią 3-celowe pakiety akumulatorów litowo-polimerowych o napięciu nominalnym 11.1V i pojemności 450mAh, co pozwala na około minutę ciągłej jazdy. Sumaryczna waga robota to 140 gramów.

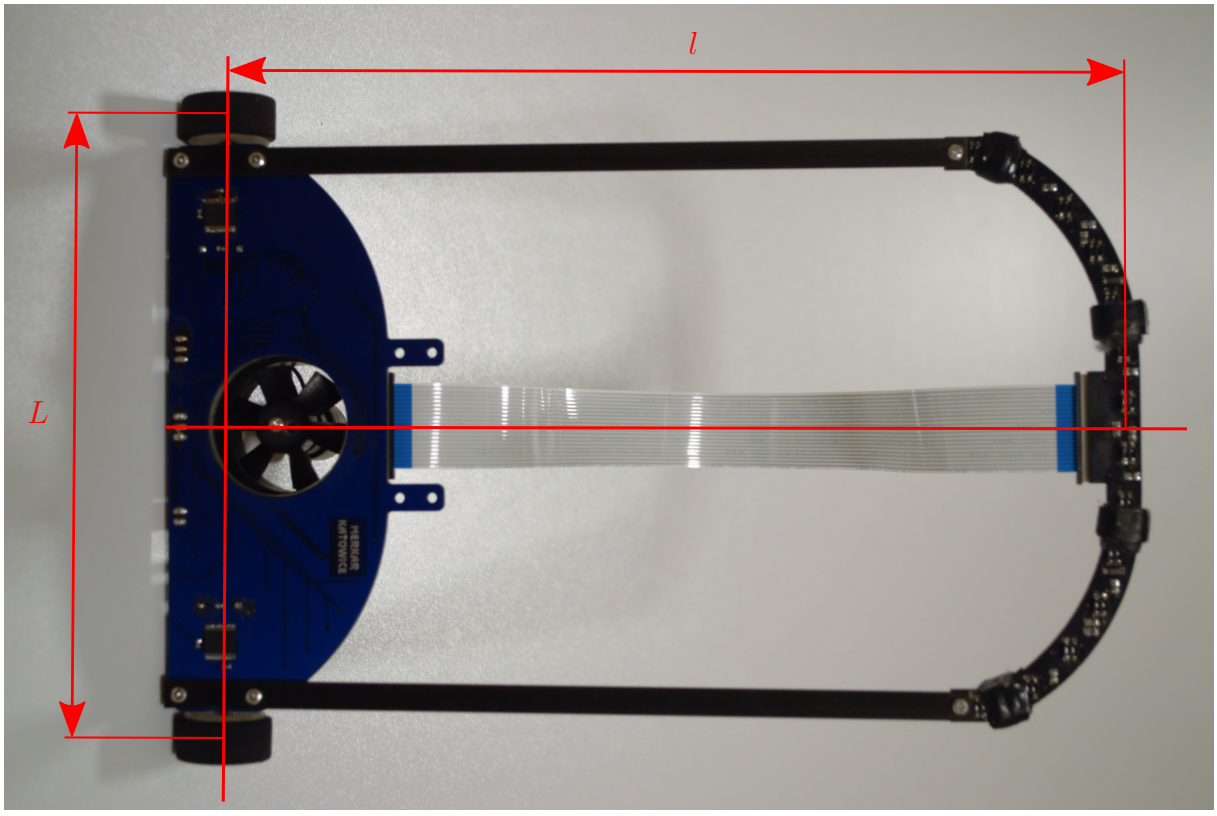
Moduł główny zawiera 32-bitowy mikrokontroler STM32F103RCT6 [8] z rdzeniem Cortex-M3, 256 kB pamięci Flash oraz 48 kB pamięci RAM i maksymalnym taktowaniem 72 MHz. Komunikacja z komputerem odbywa się przez interfejs dwukierunkowej komunikacji szeregowej UART oraz moduł Bluetooth HC-06 [9] z następującymi ustawieniami:

- prędkość transmisji – 115,2 kbps,
- 8 bitów danych,
- 1 bit stopu,
- brak bitu parzystości.

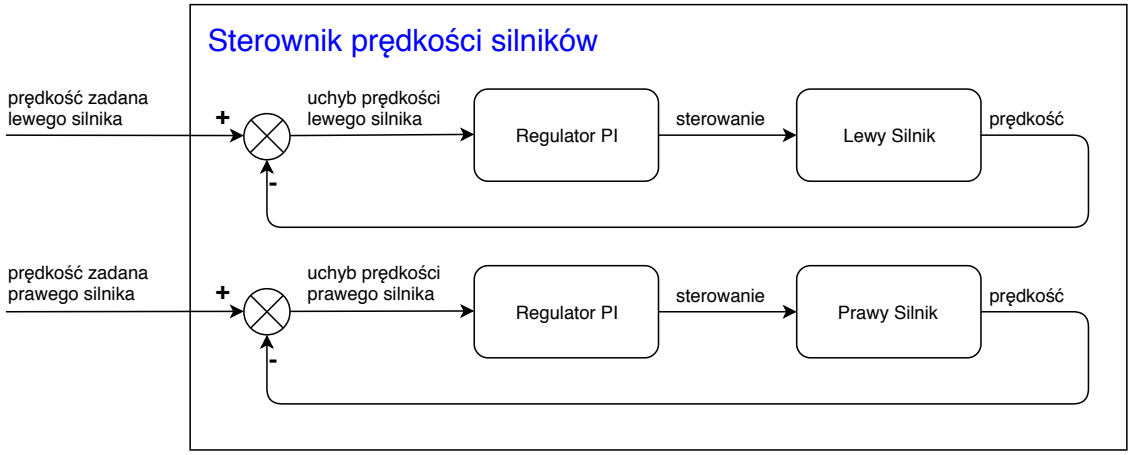
Przetwornik analogowo-cyfrowy (ADC) służy do pomiaru napięcia na kolektorach fototranzystorów czujników linii oraz poziomu naładowania akumulatora. Moduł DMA zapewnia bezpośredni zapis do pamięci mikrokontrolera, dzięki czemu procesor nie oczekuje na zakończenie pomiarów przetwornika. Każdemu czujnikowi linii odpowiadają diody LED sygnalizujące widoczność linii. Pozwalają one także na wykrycie awarii czujników.



Rysunek 2.1: Robot linefollower wykorzystywany do badań



Rysunek 2.2: Podstawowe wymiary robota



Rysunek 2.3: Sterownik prędkości silników

Napęd stanowią dwa silniki Pololu High Power z przekładnią 10:1 oraz przedłużonym wałem [10]. Maksymalna prędkość silnika według noty katalogowej to 3000 obrotów na minutę za przekładnią, co w połączeniu z kołami o średnicy $\phi = 23\ mm$ oznacza teoretyczną maksymalną prędkość robota na poziomie $3,6\frac{m}{s}$. Robot wyposażony jest w opony Mini-Z o szerokości 9 mm, które są stosowane w wyścigach pojazdów zdalnie sterowanych, gdyż charakteryzują się dużą przyczepnością i trwałością.

Przedłużone wały silników użyto do zamocowania magnesów wymaganych do pracy enkoderów, które służą do pomiaru prędkości obrotowej kół. Zastosowany w konstrukcji enkoder magnetyczny AS5040 [11] firmy AMS charakteryzuje się rozdzielczością 512 impulsów na pełen obrót wału silnika. Przy zastosowanej przekładni silnika uzyskuje się 5120 impulsów na obrót koła, co daje rozdzielczość około 14 μm na impuls. Odczyt z enkoderów odbywa się poprzez wyjścia kwadraturowe. Konwersję impulsów d_{imp} na dystans pokonany przez koło w milimetrach d_{mm} dokonujemy według zależności

$$d_{mm}[mm] = \frac{d_{imp}}{n} \cdot \pi \cdot \phi[mm], \tag{2.1}$$

gdzie n to liczba impulsów na pełen obrót koła a $\phi[mm]$ średnica koła podana w milimetrach. Po podstawieniu wartości dla prezentowanej konstrukcji uzyskujemy:

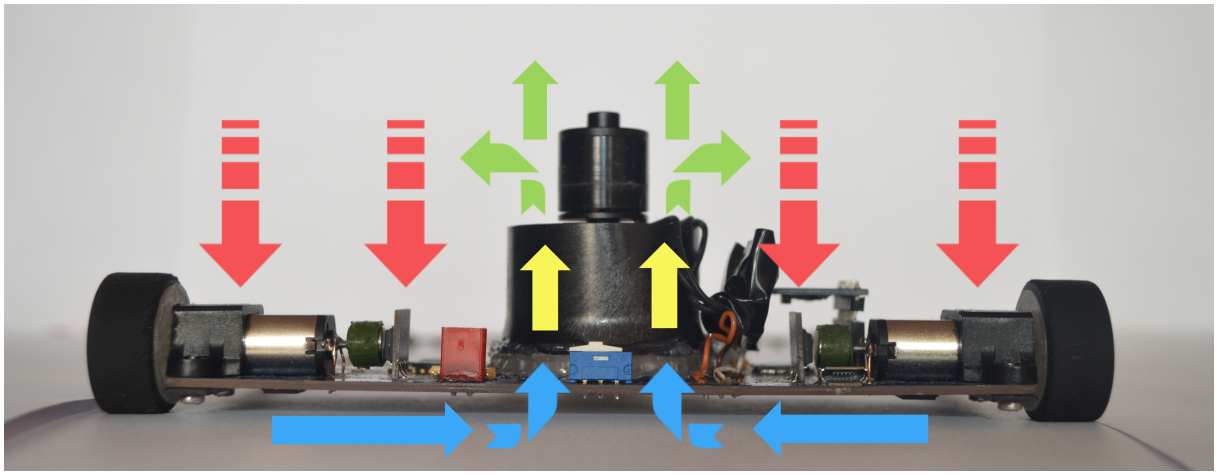
$$d_{mm} = \frac{d_{imp}}{5120} \cdot \pi \cdot 23[mm] \approx 0,0141 \cdot d_{imp}[mm] \tag{2.2}$$

Enkodery są wykorzystywane przez niskopoziomowy sterownik prędkości silników przedstawiony na rysunku 2.3. Jego zadaniem jest utrzymanie zadanych prędkości przy pomocy regulatora proporcjonalno-całkującego.

Kluczowym elementem konstrukcji zwiększającym docisk do podłoża i minimalizującym poślizgi jest napęd tunelowy. Składa się on z:

- wysokoobrotowego silnika bezszczotkowego (BLDC) o parametrze 7000KV[1]. Przykładowo, dla napięcia 12 V oznacza to ponad 80 000 rpm,
- tunelu, w kształcie walca, umożliwiającego równomierny ruch powietrza,
- wirnika wyposażonego w 6 łopatek generujących ruch powietrza.

[1]Jednostka KV oznacza liczbę obrotów na minutę (rpm) na każdy wolt napięcia zasilającego



Rysunek 2.4: Zasada działania napędu tunelowego

Ideę działania napędu tunelowego przedstawia rysunek 2.4. Ruch obrotowy wirnika powoduje zasysanie powietrza spod robota (kolor niebieski), przepływ tunelem (kolor żółty) i wyrzut powietrza z dużą prędkością nad robotem (kolor zielony). Wytworzona w ten sposób różnica ciśnień skutkuje siłą dociskową (kolor czerwony) w kierunku podłoża działającą na robota od góry.

Robot jest wyposażony w moduł IMU[2] MPU-6050 [12] firmy InvenSense, który jest powszechnie używany w stabilizacji dronów i samolotów RC. Posiada on 3-osiowy akcelerometr do pomiaru przyspieszeń, 3-osiowy żyroskop do pomiaru prędkości kątowych oraz cyfrowy termometr do kompensacji pomiarów. W pracy korzystano z maksymalnych zakresów pomiarowych, które wynoszą $\pm 16g$ dla akcelerometru i $\pm 2000°/s$ dla żyroskopu. Układ posiada wyprowadzenia umożliwiające podłączenie zewnętrznego magnetometru. Komunikacja z mikrokontrolerem odbywa się po magistrali I^2C w trybie Fast Mode z prędkością 400 kbps z podłączonym dodatkowym wyprowadzeniem generującym przerwanie, gdy nowy pomiar jest gotowy do odczytu.

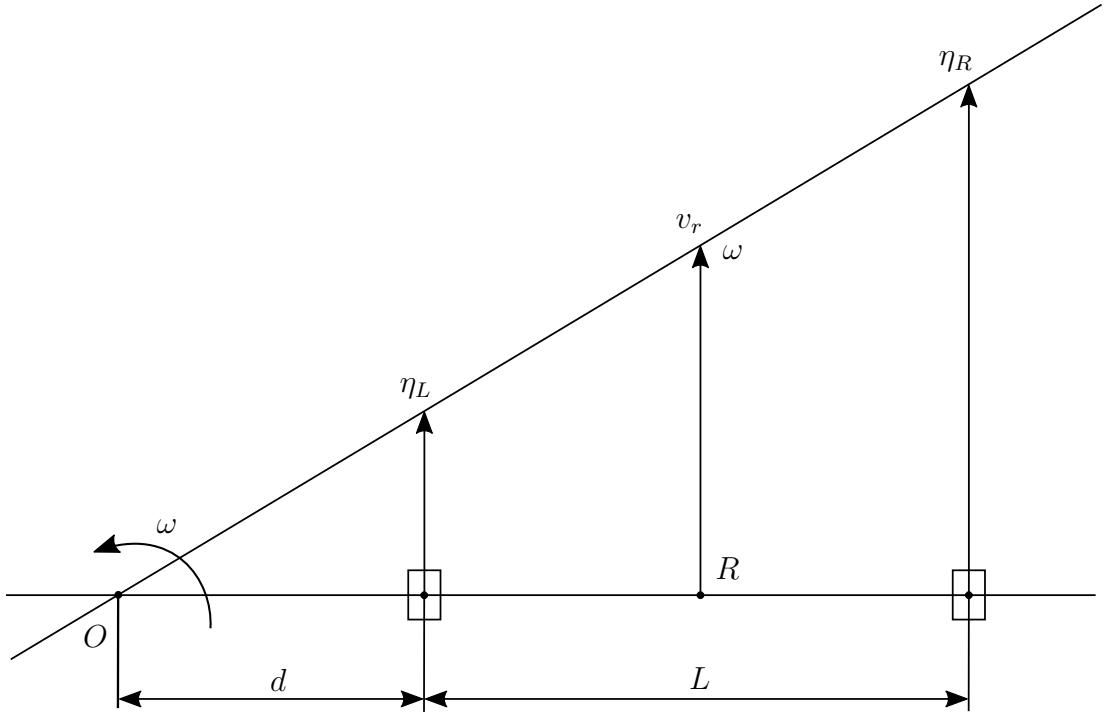
Moduł czujników składa się z 14 transoptorów odbiciowych KTIR0711s [13] do detekcji linii na kontrastującym podłożu. Każdy z transoptorów jest zbudowany z diody emitującej światło podczerwone, które jest wychwytywane przez umieszczony w nim odbiornik w postaci fototranzystora. Napięcie mierzone na kolektorze tranzystora jest odwrotnie proporcjonalne do ilości światła odbitego od podłoża, a zatem, z faktu dużej absorpcji światła dla ciemniejszych powierzchni wartość napięcia na kolektorze dla czarnej linii jest wysoka a dla białego podłoża niska. W procesie kalibracji czujnika dobierana jest wartość progowa, powyżej której przyjmuje się, że pod czujnikiem znajduje się czarna linia.

2.2 Model kinematyki robota

Model kinematyki robota mobilnego klasy (2,0) w postaci ogólnej opisany jest wzorem [14]

$$\dot{q}_r = \begin{pmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{pmatrix} = \begin{bmatrix} \frac{\cos\theta}{2} & \frac{\cos\theta}{2} \\ \frac{\sin\theta}{2} & \frac{\sin\theta}{2} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{pmatrix} \eta_L \\ \eta_R \end{pmatrix}, \tag{2.3}$$

[2] ang. Inertial Measurement Unit



Rysunek 2.5: Geometryczna prezentacja ruchu robota

gdzie q_r to wektor zmiennych stanu robota, η_L i η_R oznaczają odpowiednio prędkość postępową lewego i prawego koła a L szerokość robota (rozstaw kół).

Analizując zależności geometryczne pomiędzy prędkościami poszczególnych elementów robota (zobacz rysunek 2.5) uzyskujemy wyrażenie na prędkość postępową środka osi robota

$$v_r = \frac{\eta_L + \eta_R}{2} \tag{2.4}$$

oraz prędkość kątową robota

$$\omega = \frac{\eta_L}{d}. \tag{2.5}$$

Wartość d możemy wyliczyć na podstawie twierdzenia o podobieństwie trójkątów, które określa relację

$$\frac{d}{d+L} = \frac{\eta_L}{\eta_R} \tag{2.6}$$

co daje

$$d = \frac{\eta_L L}{\eta_R - \eta_L} \tag{2.7}$$

i ostatecznie po podstawieniu do (2.5)

$$\omega = \frac{\eta_R - \eta_L}{L}. \tag{2.8}$$

Całkując wartości uzyskanych w ten sposób prędkości dostajemy wyrażenie na translację T oraz rotację R w chwili t w postaci

$$T(t) = \int_0^t v_r dt = \int_0^t \frac{\eta_L + \eta_R}{2} dt, \tag{2.9}$$

$$R(t) = \int_0^t \omega dt = \int_0^t \frac{\eta_R - \eta_L}{L} dt. \tag{2.10}$$

Przekształcając niniejsze zależności do postaci dyskretnej, co jest wymagane do cyfrowej implementacji algorytmu, otrzymujemy wyrażenia

$$T_i = \frac{d_{L,i} + d_{R,i}}{2}, \tag{2.11}$$

$$R_i = \frac{d_{R,i} - d_{L,i}}{L}, \tag{2.12}$$

gdzie T_i, R_i to wartości translacji i rotacji a $d_{L,i}$, $d_{R,i}$ to dystans pokonany przez odpowiednio lewe i prawe koło w i-tej iteracji. Kinematyka robota w postaci dyskretnej opisana jest równaniem

$$q_{r,i} = \begin{pmatrix} x_{r,i} \\ y_{r,i} \\ \theta_{r,i} \end{pmatrix} = \begin{pmatrix} x_{r,i-1} + T_i \cdot \cos\theta_{r,i} \\ y_{r,i-1} + T_i \cdot \sin\theta_{r,i} \\ \theta_{r,i-1} + R_i \end{pmatrix}, \tag{2.13}$$

co po uwzględnieniu zależności (2.11), (2.12) daje

$$q_{r,i} = \begin{pmatrix} x_{r,i} \\ y_{r,i} \\ \theta_{r,i} \end{pmatrix} = \begin{pmatrix} x_{r,i-1} \\ y_{r,i-1} \\ \theta_{r,i-1} \end{pmatrix} + \begin{bmatrix} \frac{\cos\theta_{r,i}}{2} & \frac{\cos\theta_{r,i}}{2} \\ \frac{\sin\theta_{r,i}}{2} & \frac{\sin\theta_{r,i}}{2} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{pmatrix} d_{L,i} \\ d_{R,i} \end{pmatrix}. \tag{2.14}$$

Jako stan początkowy układu (2.14) przyjmujemy

$$q_{r,0} = \begin{pmatrix} x_{r,0} \\ y_{r,0} \\ \theta_{r,0} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{2.15}$$

Wyznaczanie orientacji robota wyłącznie na podstawie prędkości chwilowej kół obarczone jest błędem kumulującym się w czasie, co utrudnia prawidłowe wyznaczenie pozycji robota. W podrozdziale 2.3 zostanie omówiony sposób wyznaczania orientacji układu IMU, którą wykorzystano wprost do ustalenia orientacji robota $\theta_{r,i}$, natomiast dystans pokonany przez lewe i prawe koło w i-tej iteracji posłuży do wyznaczenia pozycji. Tak sformułowane zasady dają opis kinematyki robota w postaci dyskretnej

$$q_{r,i} = \begin{pmatrix} x_{r,i} \\ y_{r,i} \\ \theta_{r,i} \end{pmatrix} = \begin{pmatrix} x_{r,i-1} \\ y_{r,i-1} \\ 0 \end{pmatrix} + \begin{bmatrix} \frac{\cos\theta_{r,i}}{2} & \frac{\cos\theta_{r,i}}{2} & 0 \\ \frac{\sin\theta_{r,i}}{2} & \frac{\sin\theta_{r,i}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} d_{L,i} \\ d_{R,i} \\ \overline{\theta}_{r,i} \end{pmatrix}, \tag{2.16}$$

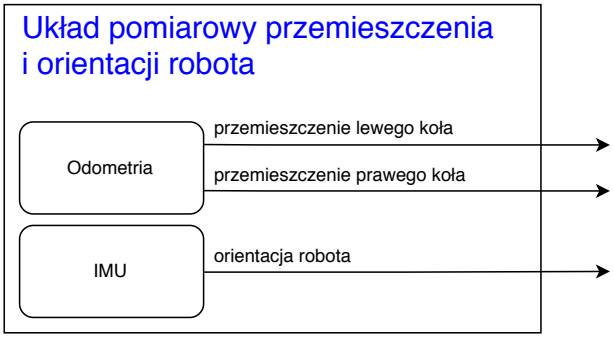
gdzie $\overline{\theta}_{r,i}$ oznacza orientację uzyskaną z modułu IMU.

Położenie środka modułu czujników $P_{c,i}$ wyznaczamy na podstawie aktualnej pozycji robota, jego orientacji oraz informacji o wysunięciu modułu czujników względem środka osi robota oznaczonej jako odległość l (zobacz rysunek 2.2)

$$P_{c,i} = \begin{pmatrix} x_{c,i} \\ y_{c,i} \end{pmatrix} = \begin{pmatrix} x_{r,i} + l \cdot \cos\theta_{r,i} \\ y_{r,i} + l \cdot \sin\theta_{r,i} \end{pmatrix}. \tag{2.17}$$

2.3 Wyznaczanie położenia i orientacji robota

Zgodnie z równaniem (2.16) do wyznaczenia położenia w i-tym kroku potrzebne są informacje o przemieszczeniu obu kół oraz estymowana orientacja robota. Dane te zapewnia



Rysunek 2.6: Układ pomiarowy przemieszczenia i orientacji robota

układ przedstawiony na rysunku 2.6. System odometrii wykorzystuje enkodery do obliczenia zmiany położenia kątowego a następnie na podstawie równania (2.2) przekształca je na dystans, jaki pokonało każde koło. Z kolei orientacja wyznaczana jest bezpośrednio przez moduł IMU. Wybór wymienionego układu pomiaru własności inercyjnych przedstawiony w podrozdziale 2.1 jest podyktowany wyposażeniem w jednostkę DMP[3], która pozwala na przeliczanie mierzonych danych na orientację względem orientacji początkowej[4]. DMP pobiera dane z akcelerometru, żyroskopu i opcjonalnego zewnętrznego magnetometru a następnie przetwarza je przy pomocy algorytmów, których producent nie ujawnia, dostarczając jako wynik kwaternion. Rezultat można odczytać bezpośrednio z rejestru DMP lub z bufora FIFO układu MPU-6050. Jednostka DMP posiada dostęp do zewnętrznego wyprowadzenia układu, służącego do generowania przerwań, gdy nowy pomiar jest gotowy do odczytu.

Na etapie testów uzyskana częstotliwość generowania pomiarów przy użyciu jednostki DMP oscylowała wokół wartości 120 Hz. Biblioteka do obsługi modułu jest wyposażona w funkcje przeliczające dane z kwaternionów na radiany, co pozwala wprost uzyskać informację o orientacji robota. Po zamontowaniu modułu dokonano kalibracji czujników na płaskim podłożu w nieruchomej pozycji.

[3] ang. Digital Motion Processor
[4] Orientacja początkowa jest określana automatyczznie przez DMP po jego włączeniu

Rozdział 3

Reprezentacja trasy

Sposób reprezentacji trasy powinien być intuicyjny i umożliwiać analizę tak, by potencjalne błędy można było łatwo zdiagnozować i skorygować. Zagadnienie dotyczy zarówno prezentacji mapy wykonanej w trakcie przejazdu próbnego, w którym robot porusza się z małą prędkością, w celu dokładnego odtworzenia trasy, a także dalszych modyfikacji, których celem jest optymalizacja toru właściwego przejazdu. Tak sformułowane zadanie realizują dwie podstawowe formy reprezentacji trasy:

- uszeregowany zbiór punktów trasy $(x_i, y_i)^T$ opisanych w kartezjańskim układzie współrzędnych,
- krzywizna trasy w funkcji odległości $K(s)$.

Obie metody są równoważne, co oznacza, że przy zastosowaniu transformacji z uwzględnieniem warunków początkowych można uzyskać reprezentację trasy zarówno w postaci zbioru punktów $(x_i, y_i)^T$ oraz krzywizny $K(s)$.

3.1 Uszeregowany zbiór punktów

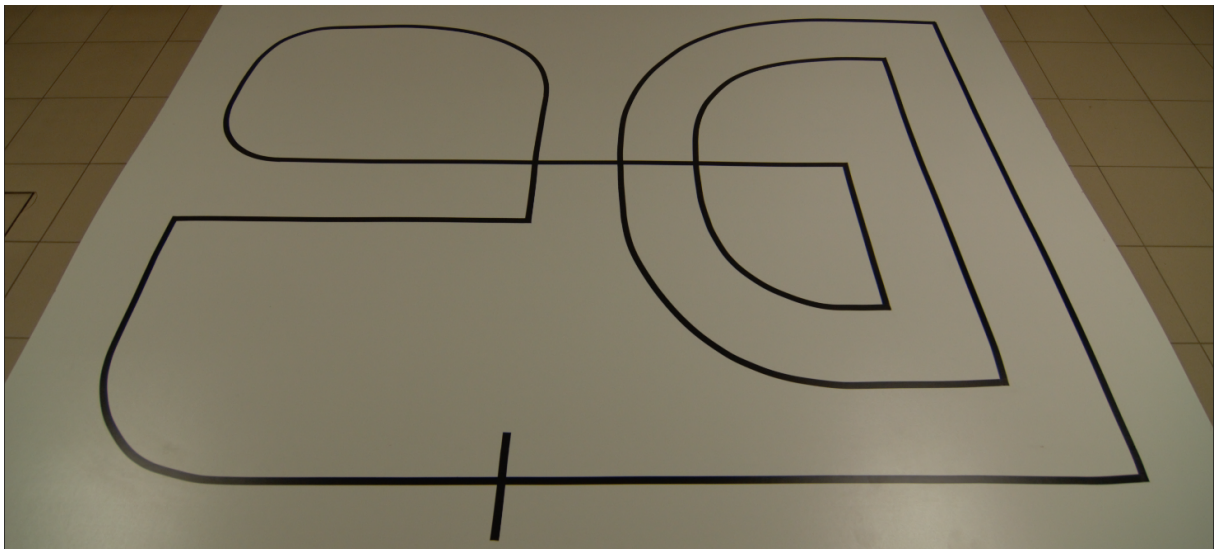
Zbiór punktów $P_i = (x_i, y_i)^T$ to najprostsza forma reprezentacji przebiegu zdyskretyzowanej trasy. Umożliwia on łatwe porównanie z rzeczywistą trasą, wychwycenie błędów i korektę algorytmów mapowania oraz prezentację wygenerowanych rozwiązań w procesie optymalizacji trajektorii. Jednocześnie, zawiera pełną informację o trasie. Mimo to, analiza krzywizny trasy w przedstawionej reprezentacji może być trudna, zwłaszcza dla długich tras i mapowania z małymi odstępami między kolejnymi punktami. Przykładową trasę oraz jej postać dyskretną uzyskaną w procesie mapowania przedstawiono na rysunkach 3.1 i 3.2.

3.2 Krzywizna trasy w funkcji odległości

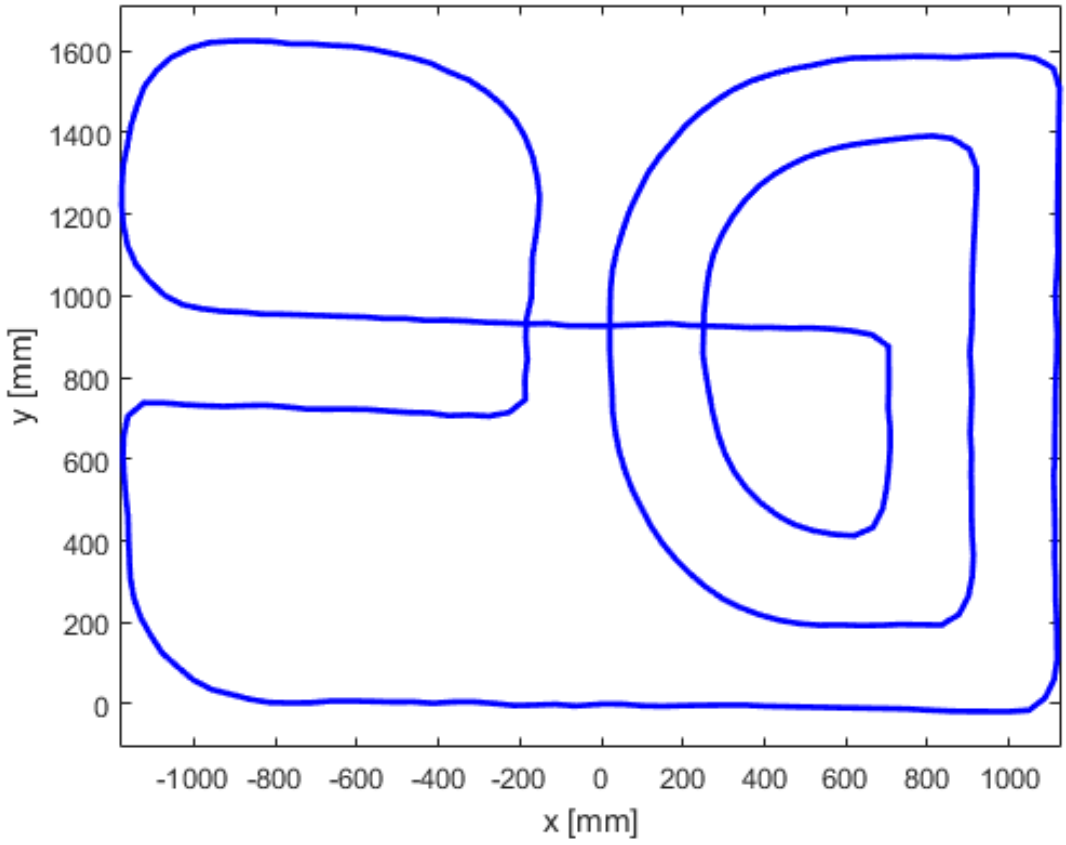
Krzywizna trasy jest kluczowa na etapie optymalizacji trajektorii. Zależy od niej maksymalna prędkość z jaką może poruszać się robot w danym fragmencie. Niestety, sama krzywizna trasy jako jedyna forma jej reprezentacji jest niewystarczająca, gdyż stosując ją, na podstawie wykresów trudno jest ocenić poprawność generowanych rozwiązań.

By wyliczyć krzywiznę trasy $K(s)$ w funkcji odległości należy wyznaczyć przyrost odległości s w postaci

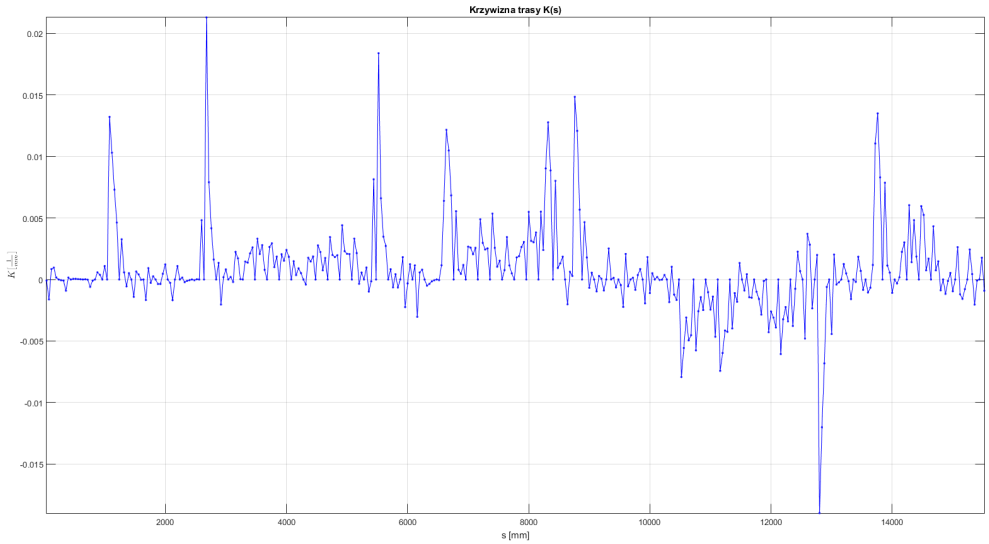
$$s_i = s_{i-1} + T_i, \tag{3.1}$$



Rysunek 3.1: Trasa testowa



Rysunek 3.2: Mapa trasy testowej



Rysunek 3.3: Krzywizna trasy testowej

gdzie $s_0 = 0$ i

$$T_i = \sqrt{dx_i^2 + dy_i^2} \tag{3.2}$$

z $dx_i = x_i - x_{i-1}$ i $dy_i = y_i - y_{i-1}$, (x_i, y_i) będącymi składowymi punktu P_i, oraz aktualną orientację trasy w postaci

$$\alpha_i = atan2(dy_i, dx_i). \tag{3.3}$$

Funkcja $atan2$ zwraca wartość kąta z przedziału $[-\pi, \pi)$ między osią X a prostą przechodzącą przez punkt (dx_i, dy_i) i początek układu współrzędnych. Znaki obu argumentów są wykorzystywane do wyznaczenia ćwiartki, do której należy wynik. Krzywiznę określa się jako stosunek zmiany orientacji trasy do przyrostu odległości

$$K_i = \frac{d\alpha_i}{T_i}, \tag{3.4}$$

gdzie

$$d\alpha_i = \alpha_i - \alpha_{i-1}{}^{1}. \tag{3.5}$$

Przyjmuje się, że dla stanu początkowego s_0 krzywizna $K_0 = 0$. Rysunek 3.3 przedstawia wykres krzywizny w funkcji odległości dla trasy testowej z rysunku 3.1.

[1] Zmianę orientacji określa się jako minimalną wartość obrotu jaki należy wykonać, aby z orientacji α_{i-1} przejść do α_i. Wartość ta należy do przedziału $[-\pi, \pi)$.

Rozdział 4

Budowa mapy trasy

Proces budowy mapy trasy przebiega w dwóch etapach: mapowania, kiedy to uzyskujemy zbiór punktów reprezentujących trasę oraz wstępnej obróbki, w którym mapa dostosowywana jest do potrzeb procesu optymalizacji ścieżki – zastosowane algorytmy zakładają stały odstęp między kolejnymi punktami reprezentującymi trasę oraz wymagają wyznaczenia referencyjnych punktów krańcowych ograniczających tor ruchu robota.

4.1 Mapowanie

Mapowanie, czyli wstępny przejazd robota celem budowy mapy trasy to pierwszy etap zadania optymalizacji ścieżki. Na tym etapie kluczowe jest precyzyjne odtworzenie trasy z odpowiednio małymi odstępami między kolejnymi próbkami pozycji. W tym celu stosuje się małą prędkość przejazdu robota oraz dużą częstotliwość próbkowania. Rysunek 1.1, który zaprezentowano w rozdziale 1 przedstawia zaimplementowany algorytm mapowania – regulacja PD położenia względem linii z rzeczywistą kontrolą prędkości kół.

W pętli głównej programu robota obliczany jest błąd położenia modułu czujników względem linii. Informacja ta służy do wyznaczenia korekty prędkości na koła przy użyciu regulatora PD, która minimalizuje uchyb. Regulator działa z częstotliwością 125 Hz, co pozwala na szybką reakcję w przypadku zmiany położenia względem linii.

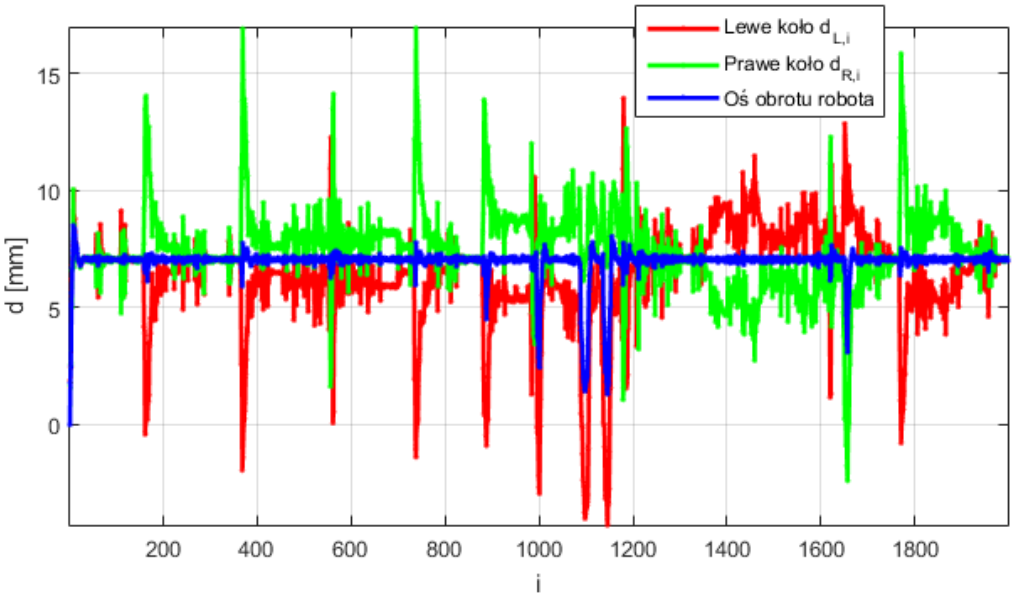
W przerwaniu zewnętrznym od modułu IMU wykonywany jest odczyt orientacji robota. Jak wspomniano w podrozdziale 2.3, jednostka DMP generuje nowe pomiary orientacji ze zmienną częstotliwością, która na etapie testów wyniosła około 120 Hz. Wartość ta ogranicza maksymalną częstotliwość, z jaką robot dokonuje mapowania.

Przerwanie wewnętrzne od timera z częstotliwością 1000 Hz synchronizuje cały proces mapowania. Obługa przerwania składa się z:

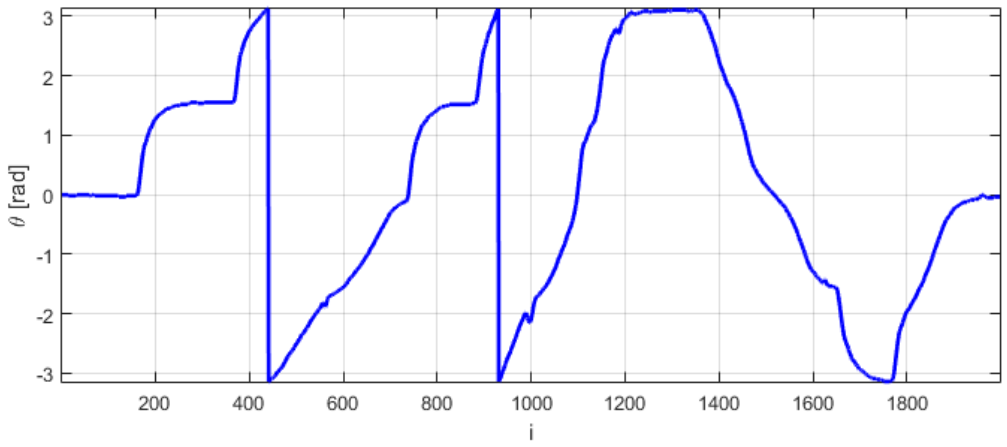
- sterowania rzeczywistą prędkością kół – wykorzystuje informacje o zmianie położeń kątowych kół względem poprzedniej iteracji, które przyjmuje się jako prędkości chwilowe silników. Regulator PI koryguje je na podstawie różnicy prędkości zadanej i chwilowej.

- wysyłania pomiarów – dane przesyłane przez robota w trakcie mapowania to przemieszczenie lewego i prawego koła ($d_{L,i}$,$d_{R,i}$), ostatni odczyt orientacji robota θ oraz uchybu położenia względem linii u. Przyjęto, że dane pomiarowe przesyłane są w co dziesiątym przerwaniu, co stanowi częstotliwość transmisji równą 100 Hz. Takie rozwiązanie zapewnia aktualizację stanu robota w regularnych odstępach czasu.

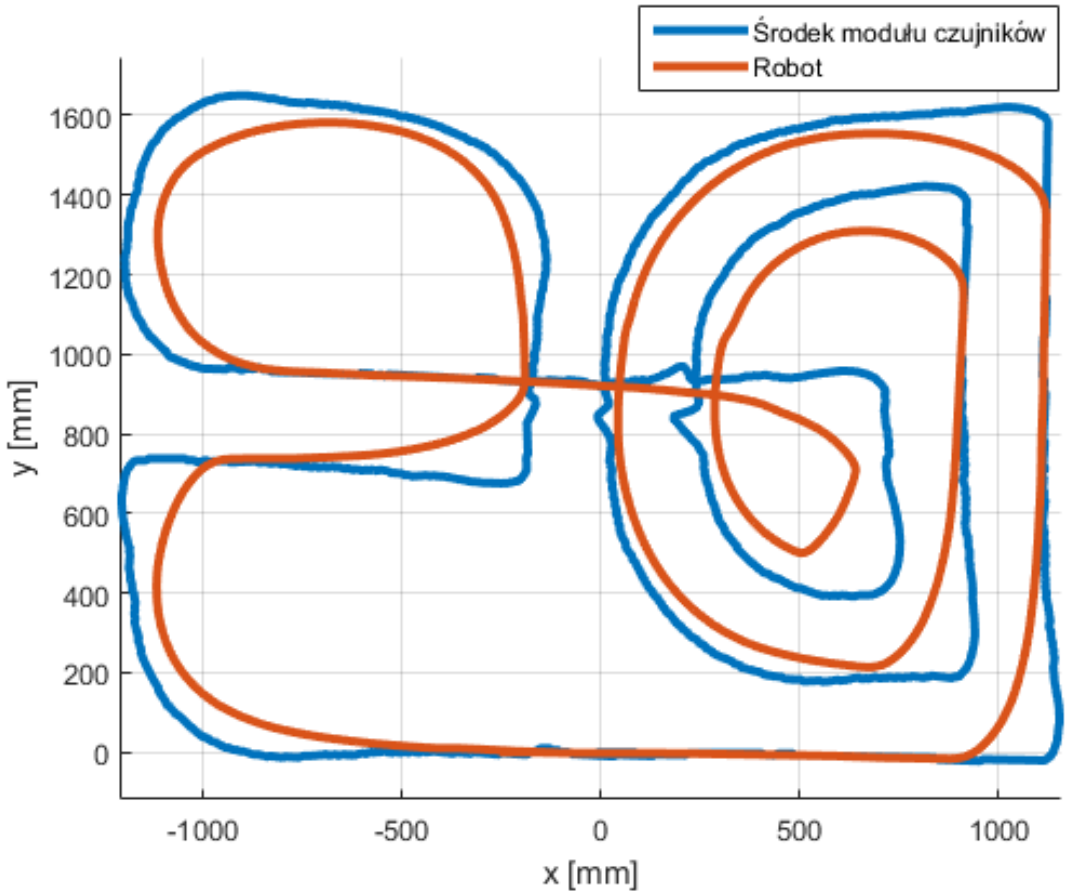
Przykładowe dane zebrane w procesie mapowania zaprezentowano na rysunkach 4.1 i 4.2.



Rysunek 4.1: Dystans pokonany w i-tej iteracji



Rysunek 4.2: Orientacja robota w i-tej iteracji



Rysunek 4.3: Ścieżka robota i modułu czujników

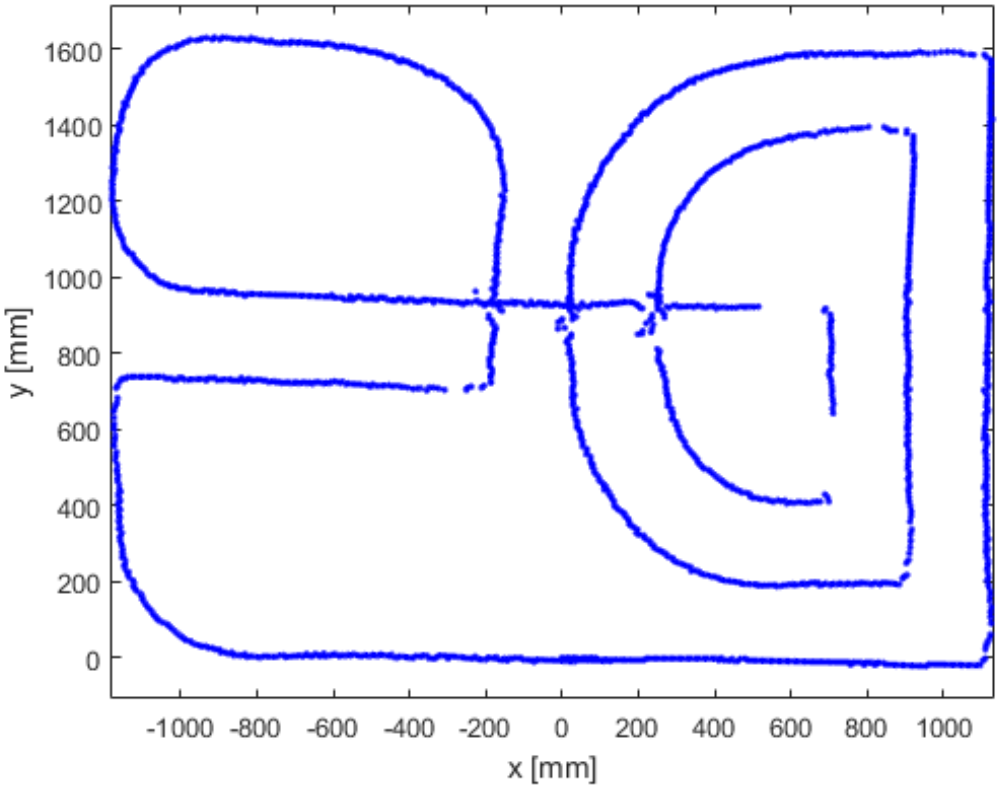
Stosując równania (2.16), (2.17) wyznaczono trajektorię, po której poruszał się robot, oraz trajektorię modułu czujników. Odpowiadające im ścieżki zaprezentowano na rysunku 4.3. Aby wygenerować mapę trasy należy obliczyć przesunięcie względem środka osi robota w każdej iteracji, do czego wykorzystujemy informację o uchybie położenia względem linii u_i oraz wysunięciu modułu czujników l. Przyjmijmy, że funkcja f wyznacza przesunięcie podłużne Δx_i oraz przesunięcie poprzeczne Δy_i od środka osi robota

$$\begin{pmatrix} \Delta x_i \\ \Delta y_i \end{pmatrix} = f(u_i, l). \tag{4.1}$$

Przesunięcia dla poszczególnych wartości uchybu ściśle zależą od rozmieszczenia czujników linii. Następnie z równania

$$P_{l,i} = \begin{pmatrix} x_{l,i} \\ y_{l,i} \end{pmatrix} = \begin{pmatrix} x_{r,i} + \Delta x_i \cdot \cos\theta_{r,i} - \Delta y_i \cdot \sin\theta_{r,i} \\ y_{r,i} + \Delta x_i \cdot \sin\theta_{r,i} + \Delta y_i \cdot \cos\theta_{r,i} \end{pmatrix} \tag{4.2}$$

wyznacza się położenie linii w układzie współrzędnych kartezjańskich. Otrzymane punkty tworzą uszeregowany zbiór reprezentujący mapę trasy. Należy zaznaczyć, że wyznaczenie punktu odpowiadającego linii trasy nie zawsze jest możliwe. Może dojść do sytuacji, że robot swoim obrysem lub modułem czujników opuszcza trasę i żaden z czujników nie wykrywa linii. Obserwujemy to zazwyczaj na ostrych zakrętach. Mapę przykładowej trasy uzyskanej w procesie mapowania przedstawia rysunek 4.4. Widoczne są na nim dwa fragmenty przerwy w trasie wynikające ze wspomnianego opuszczenia trasy za ciasnymi zakrętami. Sposób poprawy mapy opisano w podrozdziale 4.2.



Rysunek 4.4: Uzyskana mapa trasy

4.2 Interpolacja liniowa

Metodę interpolacji liniowej zastosowano do wygenerowania mapy ze stałym krokiem Δs. Przyjęta długość kroku $\Delta s = 50mm$ redukuje błędy wynikające z ograniczonej dokładności estymacji pozycji linii i objawiające się oscylacjami o dużej częstotliwości na wykresie krzywizny w funkcji przebytej drogi. Jednocześnie, zadany krok nie powoduje nadmiernego zniekształcenia trasy, zwłaszcza we fragmentach o dużej krzywiźnie. Porównanie mapy oraz krzywizny trasy w funkcji przebytej odległości przed i po interpolacji przedstawiono na rysunkach 4.5 i 4.6.

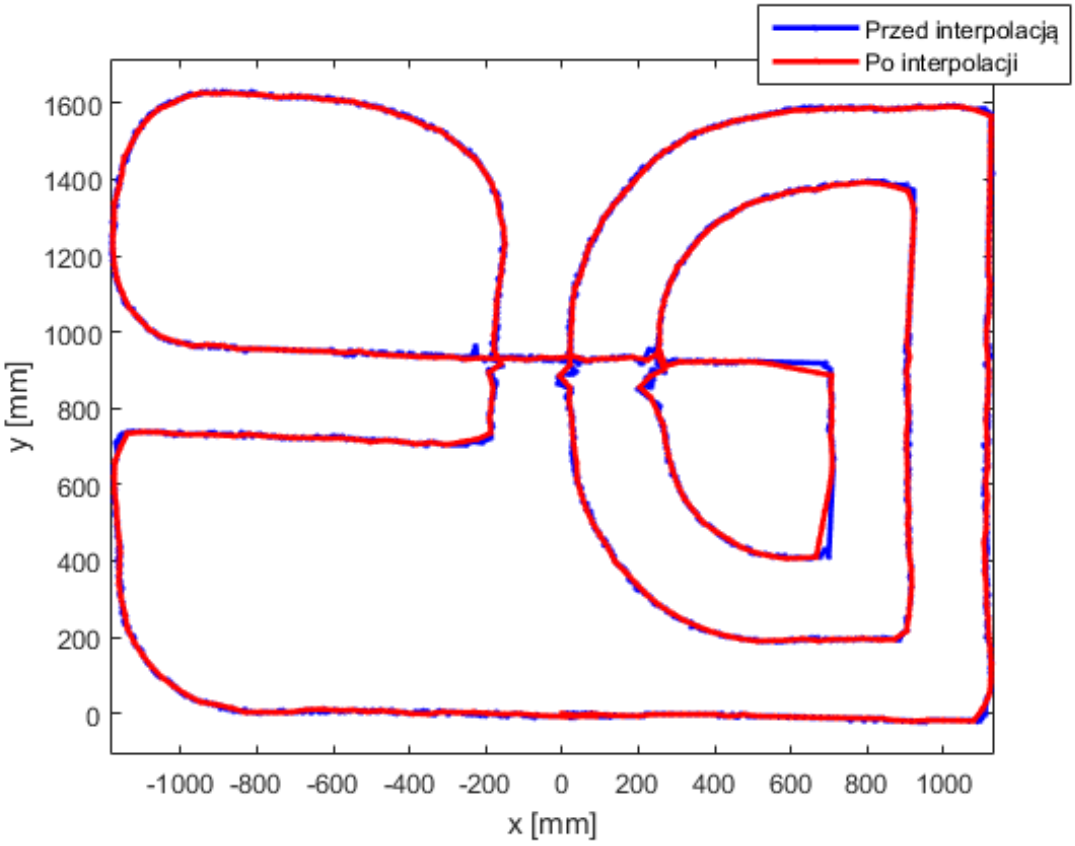
Niżej zaproponowany algorytm dokonuje interpolacji trasy w n krokach, gdzie n oznacza liczbę punktów w uszeregowanym zbiorze (x_i, y_i) reprezentującym trasę przed interpolacją.

0. Krok wstępny:
   Przyjmij punkt startowy mapy jako pierwszy punkt referencyjny $P_{ref} = (x_{ref}, y_{ref})$ i ustaw jako pierwszy element zbioru punktów po interpolacji oraz ustaw wartość licznika $i = 2$.

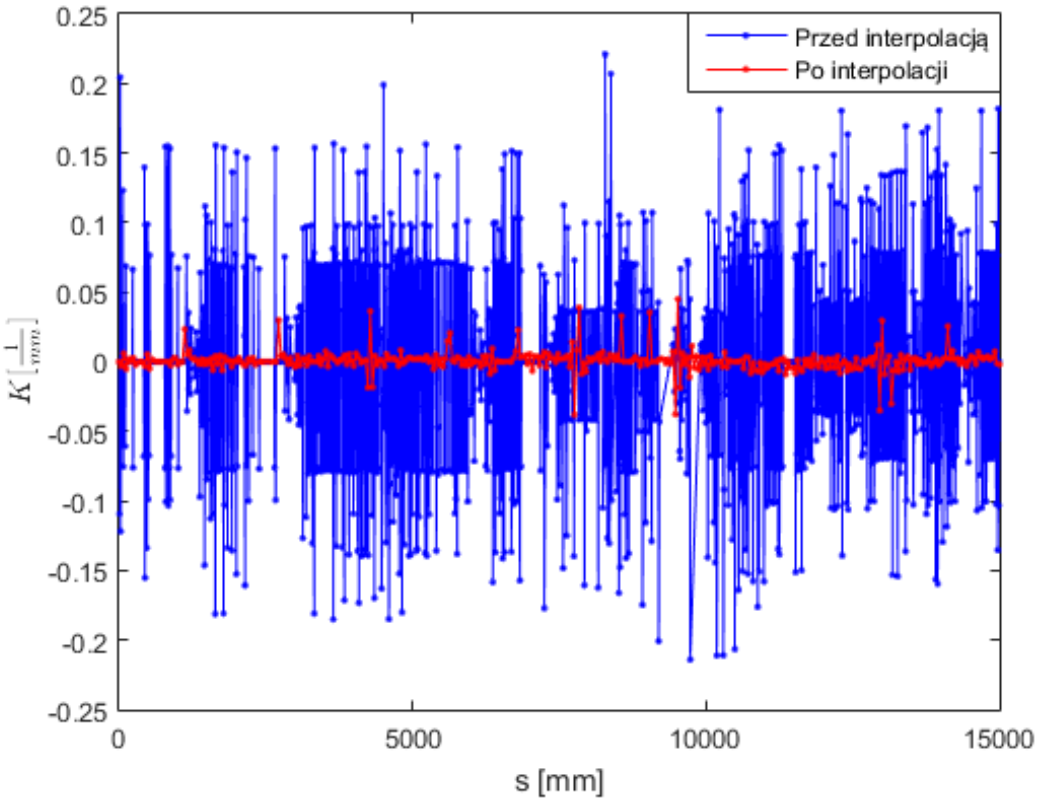
1. Jeżeli $i > n$ to STOP.

2. Jeżeli $||P_{l,i} - P_{ref}|| < \Delta s$ to idź do kroku 5.

3. Wyznacz punkt P w odległości Δs od punktu referencyjnego P_{ref} leżącego na odcinku łączącym punkty P_{ref} i $P_{l,i}$.



Rysunek 4.5: Mapa trasy przed i po interpolacji



Rysunek 4.6: Krzywizna trasy przed i po interpolacji

$$P = P_{ref} + \frac{P_{l,i} - P_{ref}}{||(P_{l,i} - P_{ref})||} \cdot \Delta s \tag{4.3}$$

4. Dodaj punkt P do zbioru punktów mapy po interpolacji i ustaw jako nowy punkt referencyjny $P_{ref} = P$.

5. Zwiększ i o 1, idź do kroku 1.

4.3 Dopuszczalny tor ruchu

Jak wspomniano na wstępie tego rozdziału, aby przejazd był zaliczony, robot nie może opuścić swoim obrysem żadnego fragmentu trasy. Stąd konieczne jest wyznaczenie lewostronnej x_L oraz prawostronnej x_R granicy toru ruchu robota, która spełnia niniejsze założenie. Dla rozpatrywanego zagadnienia szerokość toru jest stała i odpowiada szerokości robota L (zobacz rysunek 2.2).

Punkty graniczne są wyznaczane na podstawie aktualnego $P_{l,i}$ oraz sąsiadujących punktów referencyjnych $P_{l,i-1}$, $P_{l,i+1}$. Idea polega na wyznaczeniu kąta α między wektorem $\overrightarrow{P_{l,i-1}P_{l,i+1}}$ a osią odciętych OX układu współrzędnych

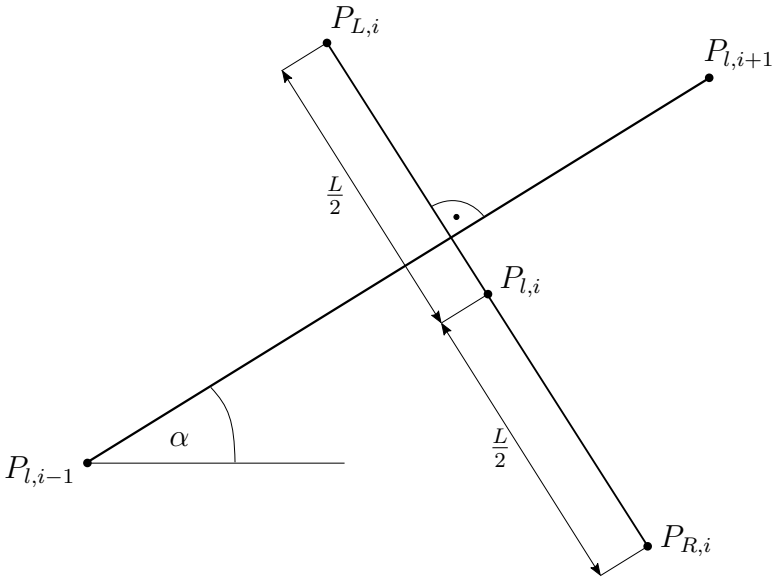
$$\alpha_i = atan2(y_{l,i+1} - y_{l,i-1}, x_{l,i+1} - x_{l,i-1}). \tag{4.4}$$

Następnie z równań

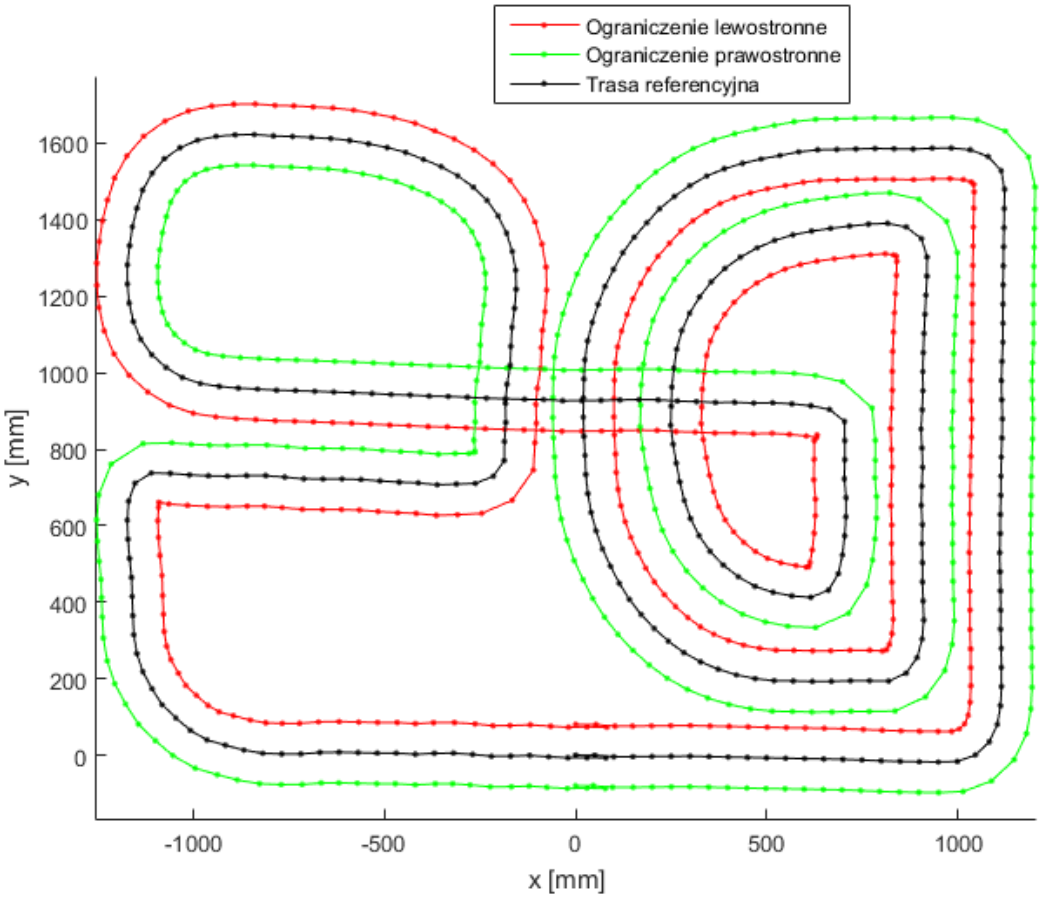
$$P_{L,i} = \begin{pmatrix} x_{L,i} \\ y_{L,i} \end{pmatrix} = \begin{pmatrix} x_{l,i} + \frac{L}{2} \cdot \cos(\alpha_i + \frac{\pi}{2}) \\ y_{l,i} + \frac{L}{2} \cdot \sin(\alpha_i + \frac{\pi}{2}) \end{pmatrix}, \tag{4.5}$$

$$P_{R,i} = \begin{pmatrix} x_{R,i} \\ y_{R,i} \end{pmatrix} = \begin{pmatrix} x_{l,i} + \frac{L}{2} \cdot \cos(\alpha_i - \frac{\pi}{2}) \\ y_{l,i} + \frac{L}{2} \cdot \sin(\alpha_i - \frac{\pi}{2}) \end{pmatrix} \tag{4.6}$$

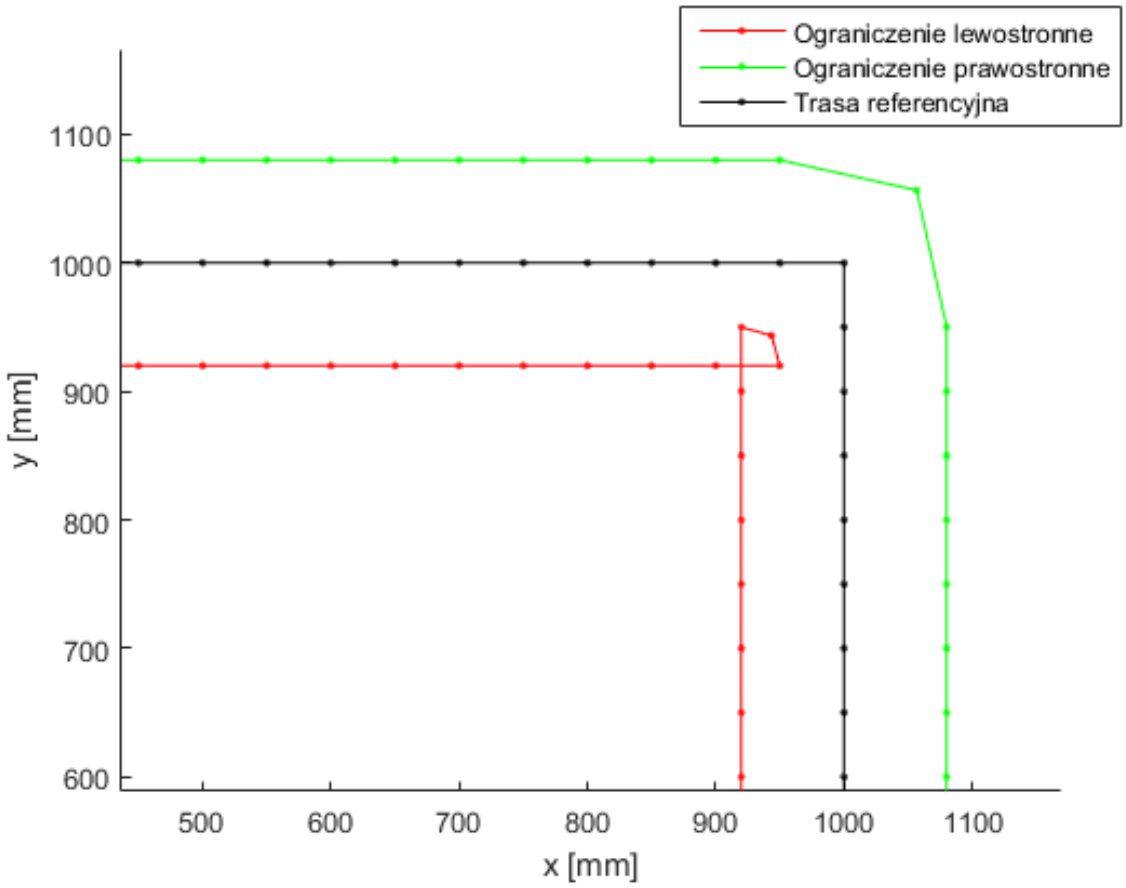
obliczany jest odpowiednio lewy $P_{L,i}$ i prawy $P_{R,i}$ punkt graniczny w odległości $\frac{L}{2}$ od punktu referencyjnego $P_{l,i}$ w kierunku prostopadłym do wspomnianego wektora. Na rysunku 4.7 zaprezentowano tę metodę w postaci geometrycznej. Przykład dopuszczalnego toru dla trasy testowej przedstawiono na rysunku 4.8. Uzyskany zbiór punktów granicznych posłuży w procesie optymalizacji toru ruchu, który opisano w kolejnym rozdziale. Na rysunku 4.9 pokazano dopuszczalny toru dla zakrętu o mierze kąta prostego. Punkty graniczne układają się w specyficzny sposób zawężając tor w otoczeniu zakrętu od jego wewnętrznej strony. Obserwujemy to, gdy szerokość toru L jest większa od odstępu między punktami referencyjnymi trasy Δs przyjętego na etapie interpolacji liniowej.



Rysunek 4.7: Metoda wyznaczania punktów granicznych



Rysunek 4.8: Dopuszczalny tor ruchu trasy testowej



Rysunek 4.9: Dopuszczalny tor dla kąta prostego

Rozdział 5

Algorytmy wyznaczania optymalnej ścieżki

Zadanie optymalizacji ścieżki wymaga na wstępie parametryzacji punktów trasy leżących pomiędzy punktami ograniczającymi tor ruchu. Następnie należy przyjąć kryterium jakości uwzględniające zależności między punktami, które pozwoli ocenić generowane rozwiązania i wyznaczyć to optymalne przez zmianę parametrów. W pracach [15, 16] zaproponowano dwa podstawowe algorytmy optymalizacji – najkrótszą ścieżkę oraz ścieżkę o minimalnej krzywiźnie, które przygotowano w ramach tej pracy. Oba algorytmy bazują wyłącznie na informacji o punktach granicznych.

5.1 Parametryzacja punktów

Położenie punktu P_i, który leży pomiędzy punktami granicznymi $P_{L,i}$, $P_{R,i}$ parametryzowanego współczynnikiem α_i wyznaczamy ze wzoru

$$P_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x_{R,i} + \alpha_i(x_{L,i} - x_{R,i}) \\ y_{R,i} + \alpha_i(y_{L,i} - y_{R,i}) \end{pmatrix} = \begin{pmatrix} x_{R,i} + \alpha_i \Delta x_i \\ y_{R,i} + \alpha_i \Delta y_i \end{pmatrix} = \begin{pmatrix} x_{R,i} \\ y_{R,i} \end{pmatrix} + \alpha_i \cdot \begin{pmatrix} \Delta x_i \\ \Delta y_i \end{pmatrix}, \quad (5.1)$$

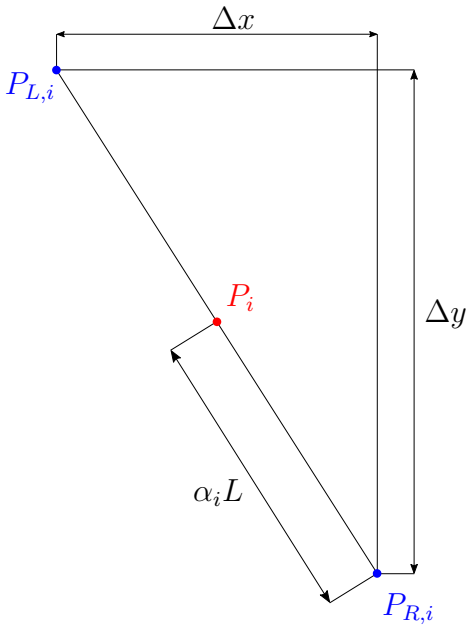
gdzie

$$\begin{cases} \Delta x_i = x_{L,i} - x_{R,i} \\ \Delta y_i = y_{L,i} - y_{R,i} \end{cases} \quad (5.2)$$

oraz współczynnik α_i spełnia warunek

$$0 \leqslant \alpha_i \leqslant 1. \quad (5.3)$$

Zadanie optymalizacji polega na wybraniu takich współczynników α_i, które minimalizują wskaźnik jakości. Geometryczną interpretację przedstawiono na rysunku 5.1. Przyjmując dla wszystkich punktów wartość współczynnika $\alpha = 0.5$, możemy odtworzyć trasę referencyjną na podstawie punktów granicznych.



Rysunek 5.1: Punkt P_i parametryzowany współczynnikiem α_i

5.2 Najkrótsza ścieżka

Algorytm wyznaczania najkrótszej ścieżki opiera się wprost na własnościach geometrycznych pomiędzy sąsiednimi punktami. Długość trasy to suma $n-1$ odcinków, gdzie n oznacza liczbę punktów reprezentujących trasę. Odległość między punktami możemy zapisać w postaci wektorowej

$$\Delta P_i = P_i - P_{i-1} = \begin{pmatrix} x_i - x_{i-1} \\ y_i - y_{i-1} \end{pmatrix} = \begin{pmatrix} \Delta P_{x,i} \\ \Delta P_{y,i} \end{pmatrix} = \begin{pmatrix} x_{r,i} - x_{r,i-1} + \alpha_i \Delta x_i - \alpha_{i-1} \Delta x_{i-1} \\ y_{r,i} - y_{r,i-1} + \alpha_i \Delta y_i - \alpha_{i-1} \Delta y_{i-1} \end{pmatrix}, \tag{5.4}$$

a następnie jako normę euklidesową

$$S_i = ||\Delta P_i|| = \sqrt{\Delta P_{x,i}^2 + \Delta P_{y,i}^2}. \tag{5.5}$$

Sumę kwadratów długości odcinków

$$S^2 = \sum_{i=2}^{n} S_i^2 = \sum_{i=2}^{n} \Delta P_{x,i}^2 + \Delta P_{y,i}^2 \tag{5.6}$$

możemy przedstawić jako funkcję wektora $\overline{\alpha}$

$$\overline{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} \tag{5.7}$$

niezależnych zmiennych

$$S^2 = \overline{\alpha}^T H_s \overline{\alpha} + B_s \overline{\alpha} + C, \tag{5.8}$$

gdzie C to wartość stała niezależna od wektora $\overline{\alpha}$. Współczynniki macierzy H_s obliczamy z następujących równań

$$H_s = \begin{bmatrix} h_{s_{1,1}} & h_{s_{1,2}} & 0 & & & & \cdots & 0 \\ h_{s_{2,1}} & h_{s_{2,2}} & h_{s_{2,3}} & 0 & & & \cdots & 0 \\ 0 & h_{s_{3,2}} & h_{s_{3,3}} & h_{s_{3,4}} & 0 & & \cdots & 0 \\ \vdots & & & & \ddots & & & \vdots \\ 0 & \cdots & 0 & h_{s_{i-1,i}} & h_{s_{i,i}} & h_{s_{i,i+1}} & \cdots & 0 \\ \vdots & & & & & \ddots & & \vdots \\ 0 & \cdots & & & & 0 & h_{s_{n,n-1}} & h_{s_{n,n}} \end{bmatrix}, \tag{5.9}$$

$$h_{s_{i-1,i}} = -\Delta x_{i-1}\Delta x_i - \Delta y_{i-1}\Delta y_i, \tag{5.10}$$

$$h_{s_{i,i+1}} = -\Delta x_i \Delta x_{i+1} - \Delta y_i \Delta y_{i+1}, \tag{5.11}$$

$$h_{s_{i,i}} = \begin{cases} 2(\Delta x_i^2 + \Delta y_i^2) & \text{dla } i \neq 1 \text{ oraz } i \neq n, \\ \Delta x_i^2 + \Delta y_i^2 & \text{dla } i = 1 \text{ lub } i = n. \end{cases} \tag{5.12}$$

Dla rozpatrywanego zagadnienia optymalizacji trasy szerokość toru jest stała i wynosi L. Wówczas w każdym punkcie P_i spełnione jest równanie

$$\Delta x_i^2 + \Delta y_i^2 = L^2.$$

Należy zauważyć, że H jest macierzą symetryczną, ponieważ zachodzi równość

$$h_{s_{i,j}} = h_{s_{j,i}}.$$

Z kolei wektor B_s konstruujemy według równania

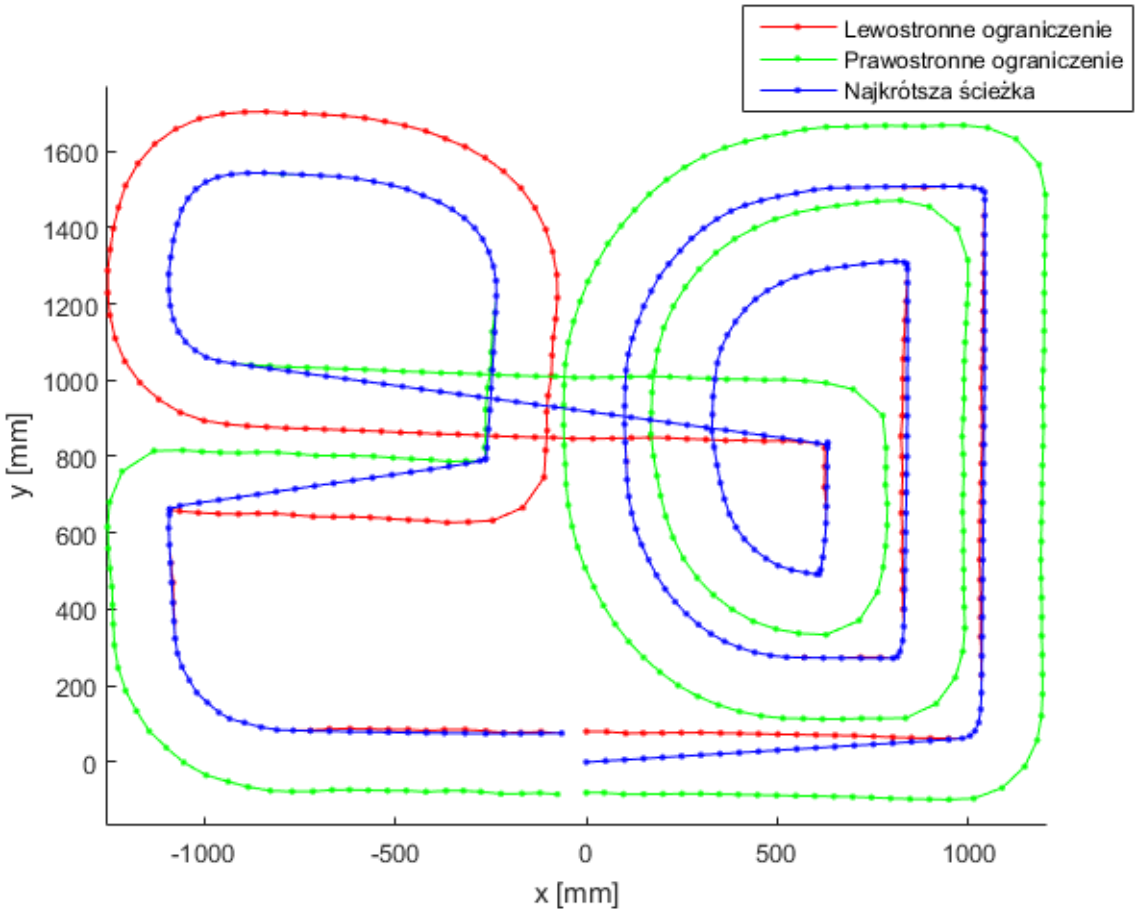
$$B_s = 2 \cdot \begin{pmatrix} (x_{r,1} - x_{r,2})\Delta x_1 + (y_{r,1} - y_{r,2})\Delta y_1 \\ (2x_{r,2} - x_{r,1} - x_{r,3})\Delta x_2 + (2y_{r,2} - y_{r,1} - y_{r,3})\Delta y_2 \\ \vdots \\ (2x_{r,i} - x_{r,i-1} - x_{r,i+1})\Delta x_i + (2y_{r,i} - y_{r,i-1} - y_{r,i+1})\Delta y_i \\ \vdots \\ (2x_{r,n-1} - x_{r,n-2} - x_{r,n})\Delta x_{n-1} + (2y_{r,n-1} - y_{r,n-2} - y_{r,n})\Delta y_{n-1} \\ (x_{r,n} - x_{r,n-1})\Delta x_n + (y_{r,n} - y_{r,n-1})\Delta y_n \end{pmatrix}^T. \tag{5.13}$$

Kryterium jakości potrzebne do wyznaczenia najkrótszej ścieżki opisane jest równaniem

$$J_s = \min_{\overline{\alpha}} \overline{\alpha}^T H_s \overline{\alpha} + B_s \overline{\alpha}. \tag{5.14}$$

Zadanie sprowadza się do wyznaczenia minimum n-wymiarowej formy kwadratowej J_s. Przykład optymalizacji przez wyznaczenie najkrótszej ścieżki przedstawiono na rysunku 5.2. Przyjmujemy, że punkt początkowy P_1 stanowi pierwszy punkt referencyjny $P_{l,1}$, co jest tożsame z założeniem

$$\alpha_1 = 0.5. \tag{5.15}$$



Rysunek 5.2: Wyznaczona najkrótsza ścieżka

5.3 Ścieżka o minimalnej krzywiźnie

W przypadku wyznaczania trasy o minimalnej krzywiźnie niemożliwe jest wyprowadzenie równania na kryterium jakości postaci

$$\min_{\overline{\alpha}} \overline{\alpha}^T H \overline{\alpha} + B\overline{\alpha} \tag{5.16}$$

na podstawie własności geometrycznych, gdyż przejście ze współrzędnych punktów parametryzowanych współczynnikami $\overline{\alpha}$ wymaga wykorzystania funkcji trygonometrycznych (zobacz podrozdział 3.2), w których nie można wydzielić współczynników α_i z postaci uwikłanej do postaci iloczynowej.

Autorzy prac [15, 16] zaproponowali rozwiązanie oparte na krzywych sklejanych[1] – wielomianów stopnia trzeciego postaci

$$\begin{cases} x_i(t) = a_{i,x} + b_{i,x}t + c_{i,x}t^2 + d_{i,x}t^3 \\ y_i(t) = a_{i,y} + b_{i,y}t + c_{i,y}t^2 + d_{i,y}t^3 \\ t(s) = \frac{s - s_{i0}}{ds_i} \end{cases}, \tag{5.17}$$

gdzie $t(s)$ reprezentuje odciętą funkcji krzywoliniowej normalizowaną długością i-tego odcinka trasy. Założenia, jakie autorzy poczynili przy opracowywaniu rozwiązania to:

[1] W języku angielskim krzywa sklejana określana jest jako spline.

- ciągłe pierwsze i drugie pochodne funkcji – oznacza to gładkie przejścia między odcinkami trasy,
- krzywa przechodząca przez wszystkie punkty P_i,
- ciągłość między ostatnim P_n a pierwszym P_1 punktem trasy – ogranicza to możliwość zastosowania algorytmu wyłącznie do zamkniętych tras, czyli takich, gdzie start i meta są w tym samym miejscu.

Krzywizna trasy $\hat{\Gamma}$ jest wyznaczana zgodnie z następującym równaniem.

$$\hat{\Gamma}^2 = \left(\frac{d^2x(s)}{ds^2}\right)^2 + \left(\frac{d^2y(s)}{ds^2}\right)^2 = \left(\frac{dt(s)}{ds}\right)^4 \left[\left(\frac{d^2x(t)}{dt^2}\right)^2 + \left(\frac{d^2y(t)}{dt^2}\right)^2\right]. \tag{5.18}$$

Długość odcinków jest stała i wynosi

$$\left(\frac{dt(s)}{ds}\right) = \frac{1}{\Delta s}, \tag{5.19}$$

więc możemy ją pominąć przy przekształceniu krzywizny $\hat{\Gamma}$ do funkcji kryterialnej J_Γ

$$J_\Gamma = \min_{\overline{\alpha}} \left(\frac{d^2x(t)}{dt^2}\right)^2 + \left(\frac{d^2y(t)}{dt^2}\right)^2. \tag{5.20}$$

Funkcja kryterialna wymaga obliczenia drugich pochodnych funkcji krzywych sklejanych w punkcie $t = 0$. Przykładowo, dla krzywej sklejanej $x(t)$ spełnione jest równanie drugiej pochodnej

$$\frac{d^2x(t)}{dt^2}|_{t=0} = D_x\overline{x}. \tag{5.21}$$

Wektor $\overline{x}$ liniowo zależy od wektora współczynników $\overline{\alpha}$

$$\overline{x} = \overline{x}_R + [dx]\overline{\alpha}, \tag{5.22}$$

gdzie $[dx]$ to macierz $n \times n$ tworzona według równania

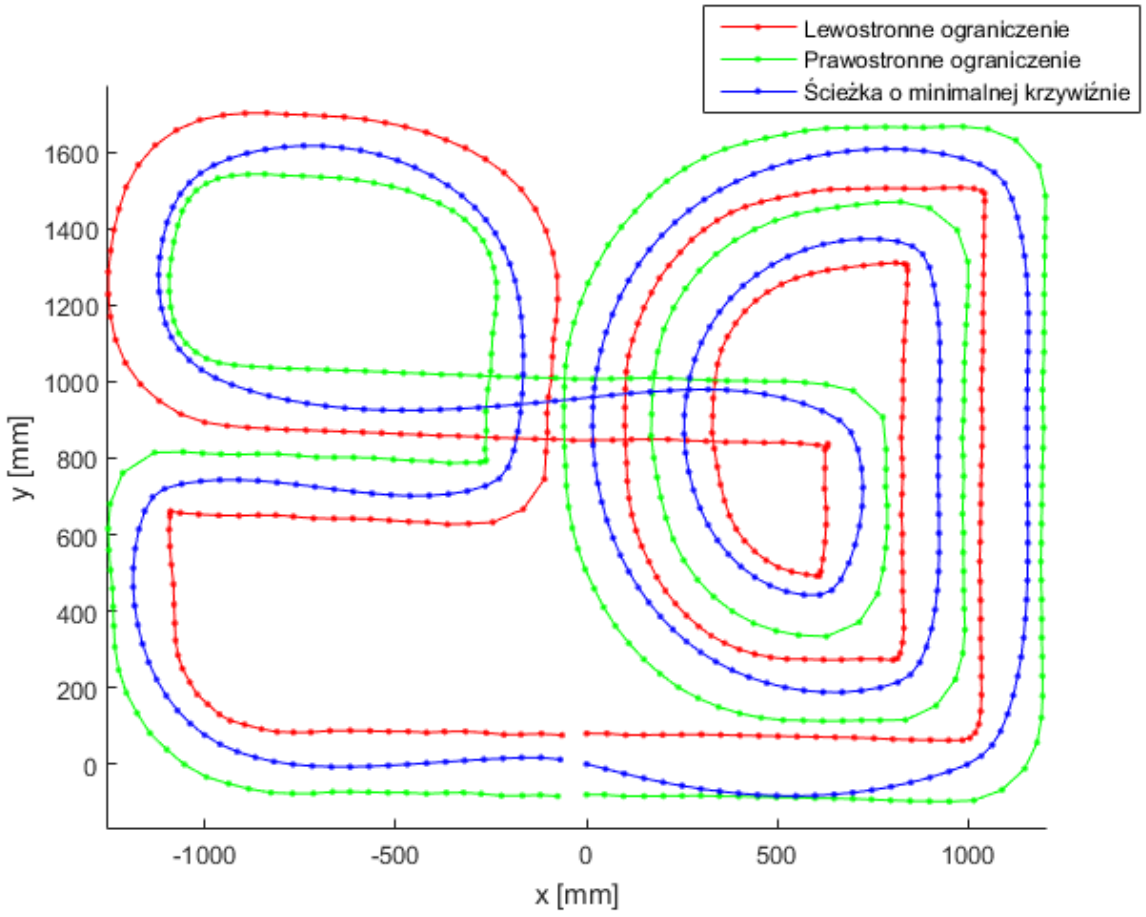
$$[dx]_{i,j} = \begin{cases} \Delta P_{x,i} & \text{dla } i = j \\ 0 & \text{dla } i \neq j \end{cases}. \tag{5.23}$$

Po podstawieniu równania (5.22) do (5.21) otrzymujemy

$$\left.\frac{d^2x(t)}{dt^2}\right|_{t=0} = D_x(\overline{x}_R + [dx]\overline{\alpha}) = D_x\overline{x}_R + D_x[dx]\overline{\alpha}. \tag{5.24}$$

Macierz D_x dla wymiaru $n = 5$ ma postać

$$D_x = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 \end{bmatrix} + \\ -6 \begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 1 & 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 4 & 1 & 0 & 0 & 1 \\ 1 & 4 & 1 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 1 & 4 & 1 \\ 1 & 0 & 0 & 1 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 \end{bmatrix}. \tag{5.25}$$



Rysunek 5.3: Ścieżka o minimalnej krzywiźnie

Wówczas kwadrat drugiej pochodnej możemy zapisać w następujący sposób

$$\begin{aligned}\left(\frac{d^2x(t)}{dt^2}|_{t=0}\right)^2 &= (D_x\overline{x}_R + D_x[dx]\overline{\alpha})^T (D_x\overline{x}_R + D_x[dx]\overline{\alpha}) \\ &= \overline{\alpha}^T \left([dx]^T D_x^T D_x[dx]\right)\overline{\alpha} + 2\left(\overline{x}_R^T D_x^T D_x[dx]\right)\overline{\alpha} + \overline{x}_R^T D_x^T D_x \overline{x}_R.\end{aligned} \tag{5.26}$$

Analogicznie postępujemy dla współrzędnych y. Po przekształceniach, funkcję kryterialną (5.20) zapisujemy w postaci macierzowej

$$J_\Gamma = \min_{\overline{\alpha}} \alpha^T H_\Gamma \alpha + B_\Gamma \alpha, \tag{5.27}$$

gdzie

$$H_\Gamma = [dx]^T D_x^T D_x[dx] + [dy]^T D_y^T D_y[dy], \tag{5.28}$$

$$B_\Gamma = 2\left(\overline{x}_R^T D_x^T D_x[dx] + \overline{y}_R^T D_y^T D_y[dy]\right). \tag{5.29}$$

Na rysunku 5.3 zaprezentowano uzyskaną ścieżkę o minimalnej krzywiźnie dla trasy testowej z założeniem, że punkt początkowy stanowi pierwszy punkt referencyjny $P_1 = P_{l,1}$.

Rozdział 6

Algorytmy sterowania robotem

Przejazd testowy i właściwy jest realizowany przy pomocy różnych algorytmów. Pierwszy bazuje na prostym algorytmie regulacji położenia względem linii zaprezentowanym na rysunku 1.1. Z kolei drugi jest bardziej złożony i wymaga wyznaczenia trajektorii przejazdu środka osi robota, co opisano poniżej.

6.1 Wyznaczanie trajektorii robota

Wyznaczenie trajektorii ruchu robota przebiega w dwóch etapach. Najpierw przekształcamy ścieżkę uzyskaną w procesie optymalizacji na trajektorię pozycji środka modułu czujników. Podstawą do jej wyznaczenia jest wybór prędkości zadanej v_d, z jaką chcemy pokonać trasę. Należy zaznaczyć, że prędkość ta nie oznacza prędkości postępowej środka osi robota, a prędkość dla środka modułu czujników rozumianej jako przemieszczenie tego punktu w określonym czasie. Sterowanie odbywa się z tą samą częstotliwością, z jaką robot przesyła pomiary i wynosi $f = 100$ Hz. Kolejne punkty należy wyznaczać przyrostowo w odległości $\Delta d = \frac{v_d}{f}$. Algorytm wyznaczania trajektorii środka modułu czujników jest analogiczny do algorytmu interpolacji liniowej, który opisano w podrozdziale 4.2.

W drugim etapie wyznaczamy trajektorię robota, jego orientację oraz prędkość postępową i kątową na podstawie trajektorii modułu czujników. Przyjmijmy, że $P_{c,i}$ oznacza pozycję środka modułu czujników w i-tej iteracji. Wówczas pozycję robota wyznaczamy według równania

$$P_{r,i} = \begin{pmatrix} \frac{d-l}{d}(x_{c,i} - x_{r,i-1}) \\ \frac{d-l}{d}(y_{c,i} - y_{r,i-1}) \end{pmatrix}, \tag{6.1}$$

gdzie

$$d = ||P_{c,i} - P_{r,i-1}||. \tag{6.2}$$

Orientacja robota wynosi

$$\theta_{r,i} = atan2(y_{c,i} - y_{r,i-1}, x_{c,i} - x_{r,i-1}). \tag{6.3}$$

Jako stan początkowy robota $P_{r,1}$ przyjmujemy

$$P_{r,1} = \begin{pmatrix} x_{r,1} \\ y_{r,1} \\ \theta_{r,1} \end{pmatrix} = \begin{pmatrix} -l \\ 0 \\ 0 \end{pmatrix}. \tag{6.4}$$

Następnie obliczamy chwilową prędkość postępową $v_{r,i}$ oraz kątową $\omega_{r,i}$ robota

$$v_{r,i} = ||P_{r,i+1} - P_{r,i}|| \cdot f, \tag{6.5}$$

$$\omega_{r,i} = (\theta_{r,i+1} - \theta_{r,i}) \cdot f. \tag{6.6}$$

6.2 Algorytm śledzenia trajektorii

Do śledzenia wyznaczonej trajektorii robota użyto algorytmu Samsona i Ait-Abderrahima [17], który w przeciwieństwie do klasycznego algorytmu Samsona obejmuje przypadek, gdy zadana prędkość postępowa i kątowa robota zależą od czasu. Algorytm ten bazuje na błędzie położenia między rzeczywistą a zadaną wartością trajektorii referencyjnej

$$\begin{cases} x_e = x_d - x \\ y_e = y_d - y \end{cases} \tag{6.7}$$

oraz błędzie orientacji

$$\theta_e = \theta_d - \theta. \tag{6.8}$$

Błąd położenia jest następnie przeliczany na referencyjny błąd śledzenia trajektorii względem układu robota

$$\begin{cases} e_x = \cos\theta \cdot x_e + \sin\theta \cdot y_e \\ e_y = -\sin\theta \cdot x_e + \cos\theta \cdot y_e. \end{cases} \tag{6.9}$$

Sterowanie kinematyczne zapewniające śledzenie trajektorii referencyjnej dane jest wzorami

$$v_{ref} = k_1 e_x + v_d \cos\theta_e, \tag{6.10}$$

$$\omega_{ref} = \omega_d + k_2\theta_e + v_d e_y \frac{\sin\theta_e}{\theta_e}. \tag{6.11}$$

Równania prędkości postępowej (6.10) i kątowej (6.11) wymagają przekształcenia do prędkości lewego i prawego koła, co robimy według zależności

$$\begin{cases} \eta_L = v_{ref} - \frac{L}{4}\omega_{ref} \\ \eta_R = v_{ref} + \frac{L}{4}\omega_{ref}. \end{cases} \tag{6.12}$$

Rozdział 7

Symulacje i eksperymenty

Na wstępie należy zdefiniować zasady, jakie muszą być spełnione, aby robot nie wypadł z trasy. Następnie zostaną one wykorzystane do przeprowadzenia symulacji oraz eksperymentów na rzeczywistym robocie.

7.1 Warunki poprawnego ukończenia przejazdu

Czynnikiem decydującym o realizowalności zadanej trajektorii jest wartość siły odśrodkowej F, działającej na robota w trakcie przejazdu, która opisana jest równaniem

$$F = \frac{mv_r^2}{r}, \tag{7.1}$$

gdzie m oznacza masę robota, v_r jego prędkość a r promień zakrętu. Przyjmiemy, że siła ta nie może przekroczyć wartości granicznej F_{gr}, która oznacza maksymalną wartość zapewniającą brak poślizgu poprzecznego kół

$$F \leqslant F_{gr}. \tag{7.2}$$

Zadanie przekształcamy do ograniczenia na prędkość maksymalną robota v_{max}

$$v_r \leqslant v_{max}(r), \tag{7.3}$$

która zależy od promienia zakrętu r

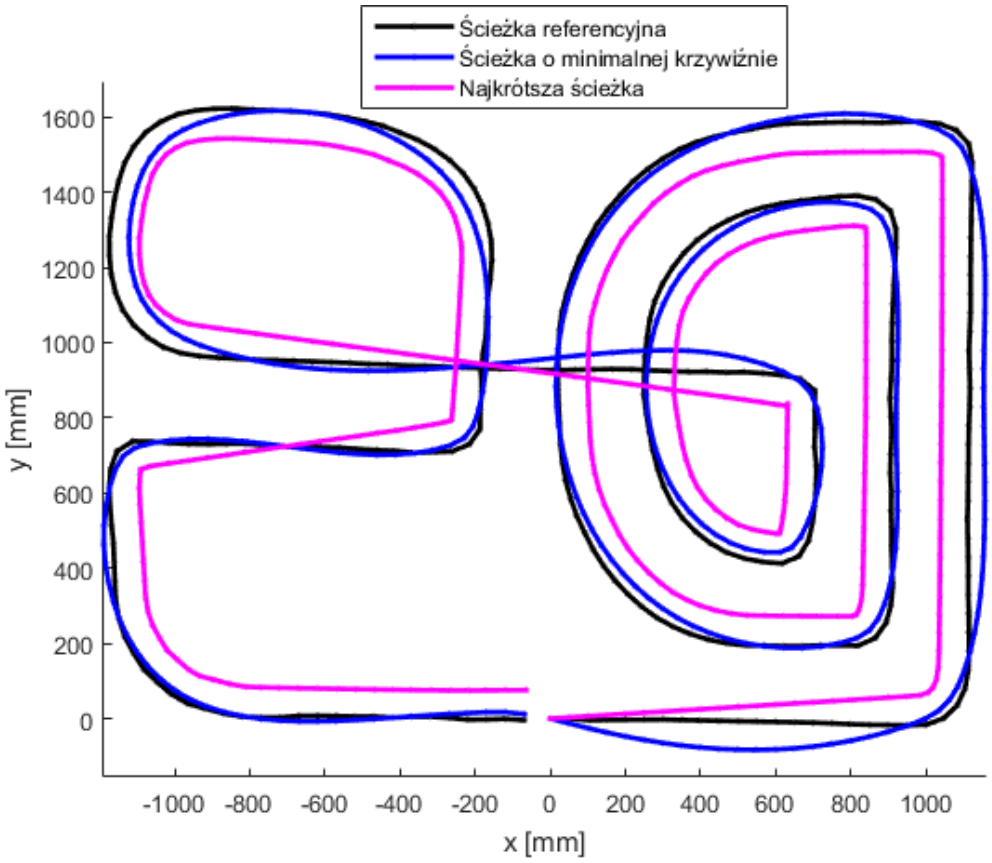
$$v_{max}(r) = \sqrt{r f_{tr} g}, \tag{7.4}$$

gdzie f_{tr} oznacza współczynnik tarcia poprzecznego kół a g przyspieszenie ziemskie. Wartość współczynnika wyznaczamy eksperymentalnie umieszczając robota na płycie wykonanej z identycznego materiału co podłoże trasy. Następnie wychylamy płytę w kierunku poprzecznym do robota i wyznaczamy maksymalną wartość kąta α, dla której robot nie zsuwa się w bok. Wartość współczynnika tarcia f_{tr} wynosi

$$f_{tr} = \operatorname{tg} \alpha. \tag{7.5}$$

W przeprowadzonym eksperymencie uzyskano następujące wartości:

- robot z wyłączonym napędem tunelowym – $\alpha = 36°$, $f_{tr} = 0,7265$,
- robot z włączonym napędem tunelowym – $\alpha = 68°$, $f_{tr} = 2,4751$.



Rysunek 7.1: Ścieżka środka modułu czujników dla wybranych algorytmów

Promień zakrętu jest odwrotnie proporcjonalny do krzywizny trasy

$$r = \frac{1}{K}. \tag{7.6}$$

Dla $K = 0$ przyjmujemy

$$r = \infty. \tag{7.7}$$

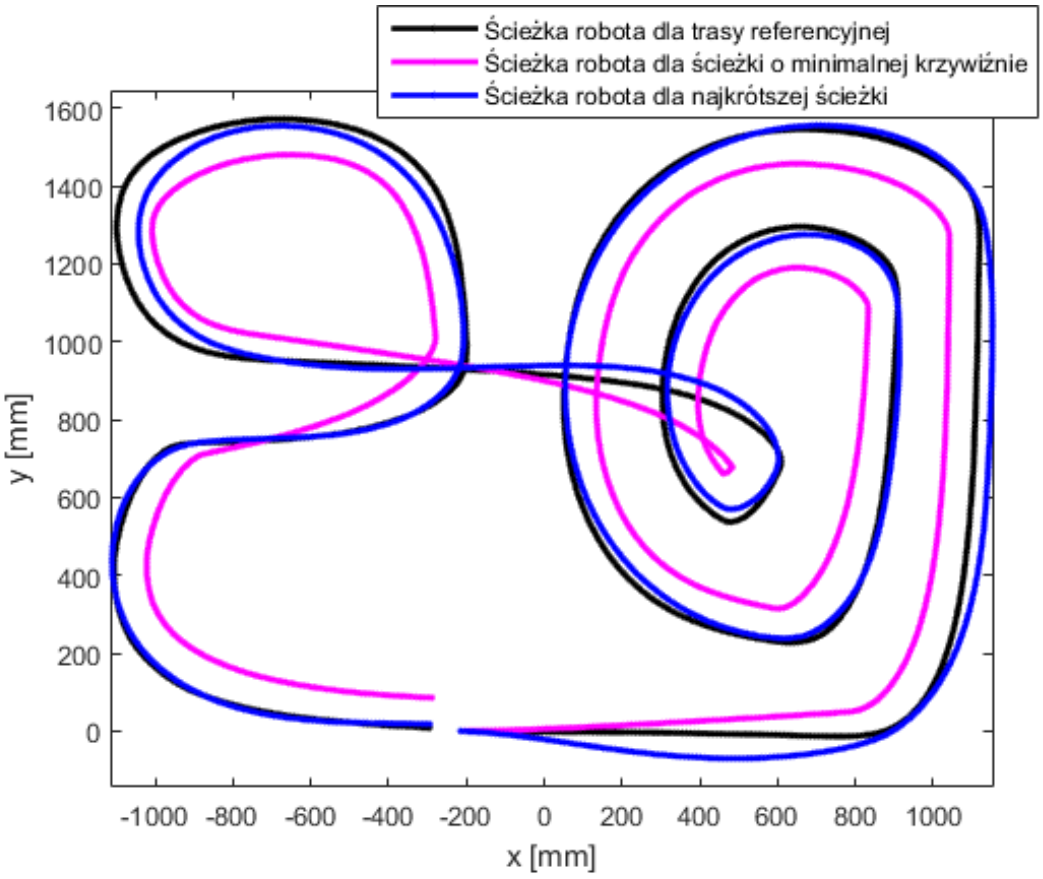
Reasumując, aby przejazd był uznany jako prawidłowy przyjmujemy, że nierówność (7.3) jest spełniona w każdej chwili. W celu jej weryfikacji obliczamy chwilowy promień zakrętu trajektorii robota, którego sposób otrzymania opisano w podrozdziale 6.1, a następnie wyznaczamy maksymalną dopuszczalną prędkość robota i porównujemy z prędkością aktualną.

7.2 Symulacje przejazdów

Pozycję środka modułu czujników dla mapy trasy oraz algorytmów najkrótszej ścieżki i minimalnej krzywizny dla symulowanego przejazdu przedstawiono na rysunku 7.1. Z kolei na rysunku 7.2 pokazano odpowiadającą tym ścieżkom ścieżki ruchu środka osi robota. Rozpatrzmy dwa przypadki, gdy dysponujemy powolnym oraz bardzo szybkim robotem[1].

Dla powolnego przejazdu przyjmijmy prędkość postępową środka osi robota $v_r = 0,2\frac{m}{s}$. Rezultaty uzyskane w symulacji przedstawiono w tabeli 7.1. W tym przypadku, decydu-

[1]Ponieważ w pracy tej pominięto kwestię dostosowywania prędkości jazdy robota do aktualnej krzywizny toru przyjmujemy, że prędkość ta jest stała w trakcie trwania przejazdu.



Rysunek 7.2: Ścieżka robota dla symulowanych przejazdów

Tabela 7.1: Parametry przejazdu robotem powolnym

Ścieżka	Długość trasy [m]	Czas [s]
referencyjna	13,08	65,4
najkrótsza	10,965	54,83
o minimalnej krzywiźnie	12,99	64,95

Tabela 7.2: Parametry przejazdu robotem szybkim

Ścieżka	Długość trasy [m]	Czas [s]
referencyjna	13,08	∞
najkrótsza	10,965	∞
o minimalnej krzywiźnie	12,99	12,99

Tabela 7.3: Maksymalne prędkości i parametry przejazdu z wyłączonym napędem tunelowym

Ścieżka	Długość trasy [m]	$v_{max}\left[\frac{m}{s}\right]$	Czas [s]	Zmiana
referencyjna	13,08	0,536	24,4	–
najkrótsza	10,965	0,274	40,01	$+63,97\%$
o minimalnej krzywiźnie	12,99	0,704	18,45	$-24,39\%$

jącym czynnikiem jest długość trasy, jaką musi pokonać robot. Najlepszy czas przejazdu uzyskano dla algorytmu najkrótszej ścieżki. Długość trasy referencyjnej i ścieżki o minimalnej krzywiźnie jest porównywalna. Algorytm minimalnej krzywizny zyskuje na zakrętach, gdyż skraca tor jazdy w tych miejscach. Wypracowana przewaga nad przejazdem po referencyjnej ścieżce jest jednak niwelowana z powodu wymaganego przygotowania pozycji przed kolejnymi zakrętami i szerszym opuszczaniem ich. Stąd też oba przejazdy uzyskały porównywalny wynik. Sumarycznie czas przejazdu po zastosowaniu optymalizacji uległ poprawie o około 16% i, czego należało się spodziewać, jest proporcjonalny do długości trasy.

Dla szybkiego przejazdu zaproponowano prędkość środka osi robota $v_r = 1\frac{m}{s}$. Jak się okazało prędkość ta przekracza maksymalną dopuszczalną prędkość v_{max} zarówno dla najkrótszej ścieżki jak i trasy referencyjnej, nawet z włączonym napędem tunelowym. Wyniki symulacji zaprezentowano w tabeli 7.2. Jak można było się spodziewać, dla ścieżki o minimalnej krzywiźnie uzyskano pięciokrotne skrócenie czasu przejazdu i był to jedyny przejazd, w którym robot nie wypadł z trasy.

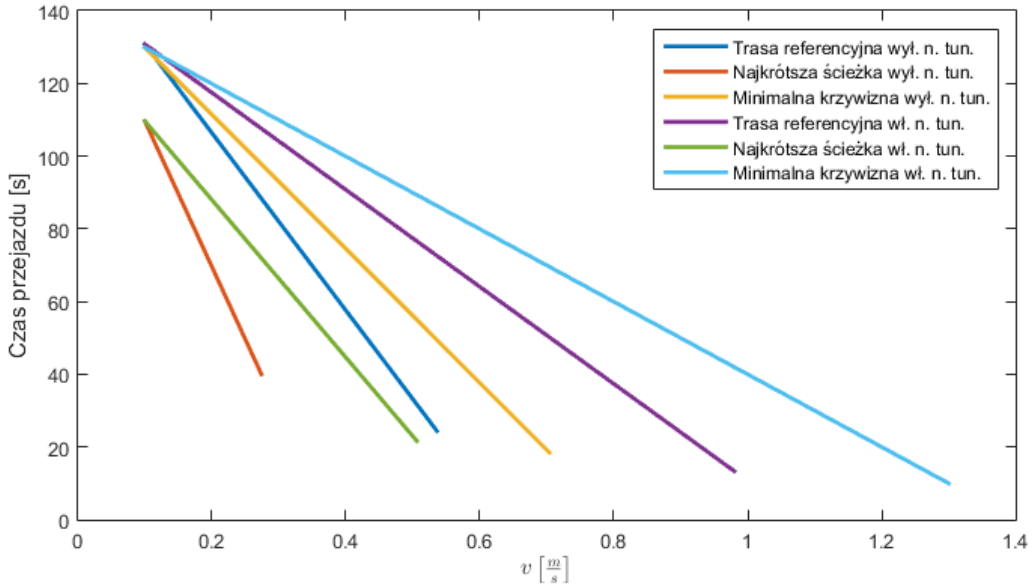
Podsumowując, najkrótsza trasa sprawdza się dla wolnych przejazdów, gdy siły odśrodkowe są stosunkowo małe i nie przekraczają dopuszczalnej wartości nawet na ostrych zakrętach. Niestety dla szybkich przejazdów nie może być stosowana. Optymalizacja toru do minimalnej krzywizny ma zastosowanie przy szybszych przejazdach z racji wygładzania zakrętów. Brak tej własności eliminuje pozostałe wyznaczone tory przejazdu.

Na koniec dla każdej z wyznaczonych tras określimy maksymalne prędkości przejazdu i uzyskane dzięki nim czasy przejazdu. Maksymalne prędkości przejazdu v_{max} z wyłączonym napędem tunelowym zebrano w tabeli 7.3, a z włączonym w tabeli 7.4. W obu przypadkach optymalizacja dla minimalnej krzywizny skutkowała poprawą czasu przejazdu o około 25%. Na rysunku 7.3 przedstawiono zbiorcze porównanie osiąganych rezultatów w funkcji prędkości v dla prezentowanych strategii z wyłączonym oraz włączonym napędem tunelowym. Różnica w rezultatach między dwoma przypadkami wynika ze wzrostu współczynnika tarcia kół, co skutkuje także wzrostem maksymalnych prędkości przejazdu i poprawą czasu pokonania trasy

Symulacje powtórzono dla drugiej trasy testowej, którą wraz z najkrótszą ścieżką oraz tą o minimalnej krzywiźnie zaprezentowano na rysunku 7.4. Rezultaty uzyskane w przejeździe z wyłączonym napędem tunelowym przedstawiono w tabeli 7.5 a z włączonym w tabeli 7.6. Wyniki ponownie potwierdziły, że przejazd po trasie o minimalnej krzywiźnie pozwala uzyskać najlepszy czas przejazdu ze wszystkich prób. Wynik poprawił się

Tabela 7.4: Maksymalne prędkości i parametry przejazdu z włączonym napędem tunelowym

Ścieżka	Długość trasy [m]	$v_{max}\left[\frac{m}{s}\right]$	Czas [s]	Zmiana
referencyjna	13,08	0,98	13,34	–
najkrótsza	10,965	0,506	21,67	$+62,44\%$
o minimalnej krzywiźnie	12,99	1,3	9,99	$-25,11\%$



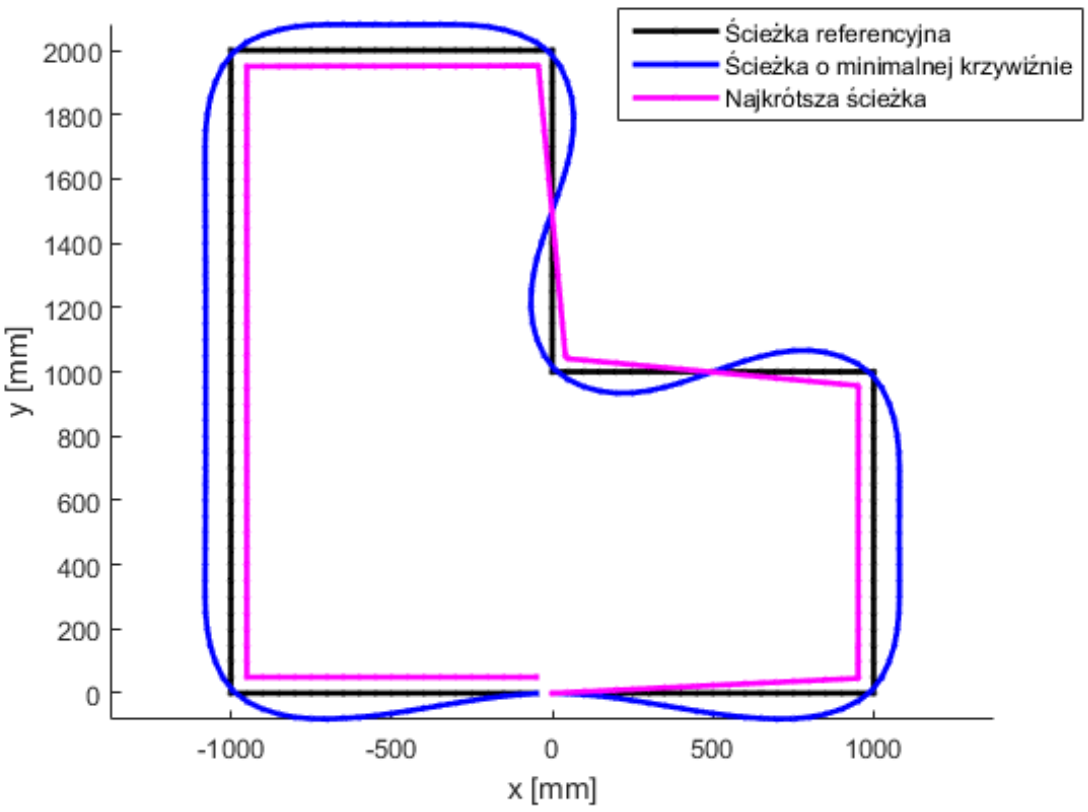
Rysunek 7.3: Porównanie rezultatów dla trasy testowej

Tabela 7.5: Maksymalne prędkości i parametry przejazdu z wyłączonym napędem tunelowym dla drugiej trasy testowej

Ścieżka	Długość trasy [m]	$v_{max}\left[\frac{m}{s}\right]$	Czas [s]	Zmiana
referencyjna	6,98	0,715	9,76	–
najkrótsza	6,51	0,732	8,89	$-8,91\%$
o minimalnej krzywiźnie	7,48	1,288	5,81	$-40,47\%$

Tabela 7.6: Maksymalne prędkości i parametry przejazdu z włączonym napędem tunelowym dla drugiej trasy testowej

Ścieżka	Długość trasy [m]	$v_{max}\left[\frac{m}{s}\right]$	Czas [s]	Zmiana
referencyjna	6,98	1,32	5,29	–
najkrótsza	6,51	1,35	4,82	$-8,88\%$
o minimalnej krzywiźnie	7,48	2,37	3,16	$-40,26\%$



Rysunek 7.4: Tor przejazdu dla drugiej trasy testowej

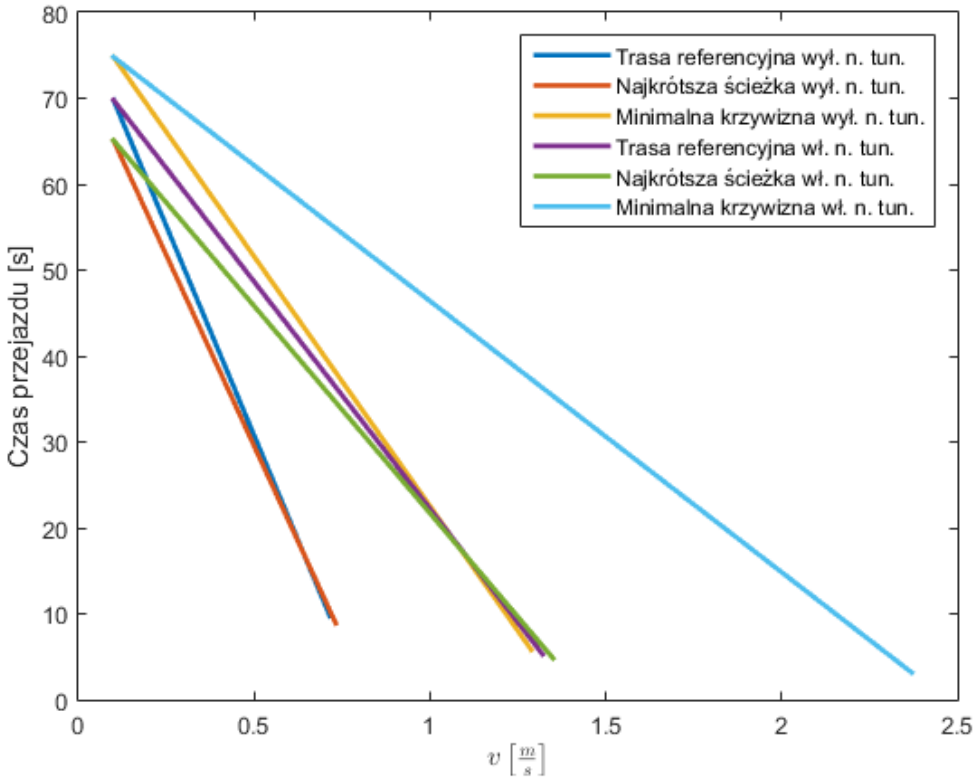
o około 40%. Z kolei dla przejazdu po najkrótszej ścieżce poprawa wyniosła około 9%. Na rysunku 7.5 pokazano zbiorcze porównanie dla drugiej trasy testowej.

7.3 Przejazd właściwy

Celem przejazdu właściwego jest pokonanie trasy po optymalnej ścieżce. Diagram funkcyjny algorytmu zaprezentowano na rysunku 1.3. Do realizacji zadania wymagana jest trajektoria, po której powinien poruszać się robot. Najstosowniejszym rozwiązaniem byłoby wygenerowanie tejże trajektorii na komputerze oraz przesłanie jej do robota, który zapisałby dane w pamięci przed rozpoczęciem przejazdu. Następnie robot samodzielnie realizuje przejazd po zadanej trasie bez komunikacji z komputerem.

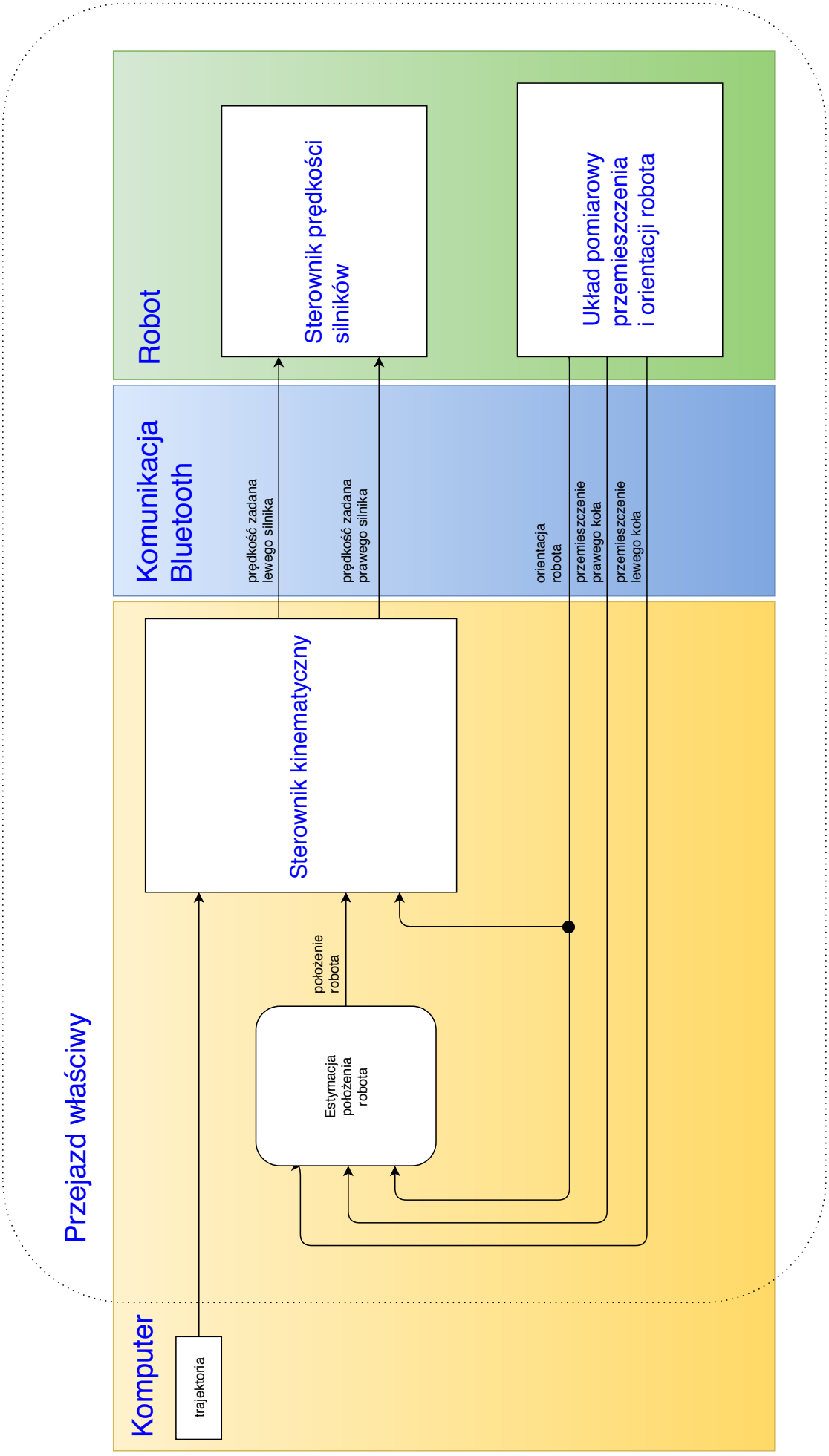
Niestety w naszym przypadku nie jest to możliwe z powodu niewystarczających możliwości obliczeniowych oraz ograniczonej pamięci mikrokontrolera. Stąd też wymagana jest stała komunikacja Bluetooth między komputerem, na którym uruchomiony jest program pełniący rolę sterownika kinematycznego dla zadania śledzenia trajektorii, a robotem, który to zadanie realizuje. Robot wykonuje cyklicznie pomiary przemieszczeń lewego i prawego koła oraz orientacji i przesyła dane do komputera. Program komputerowy na podstawie tych pomiarów estymuje nowe położenie robota i przy użyciu algorytmu śledzenia trajektorii generuje nowe prędkości obrotowe na koła i wysyła je do robota. Niniejszy podział przedstawia rysunek 7.6.

Przeprowadzone badania na rzeczywistym robocie uwidoczniły wady niniejszego rozwiązania. Komunikacja Bluetooth z urządzeniem zewnętrznym obarczona jest pewnym opóźnieniem, które negatywnie wpływa na realizację zadania. Stan robota w momencie

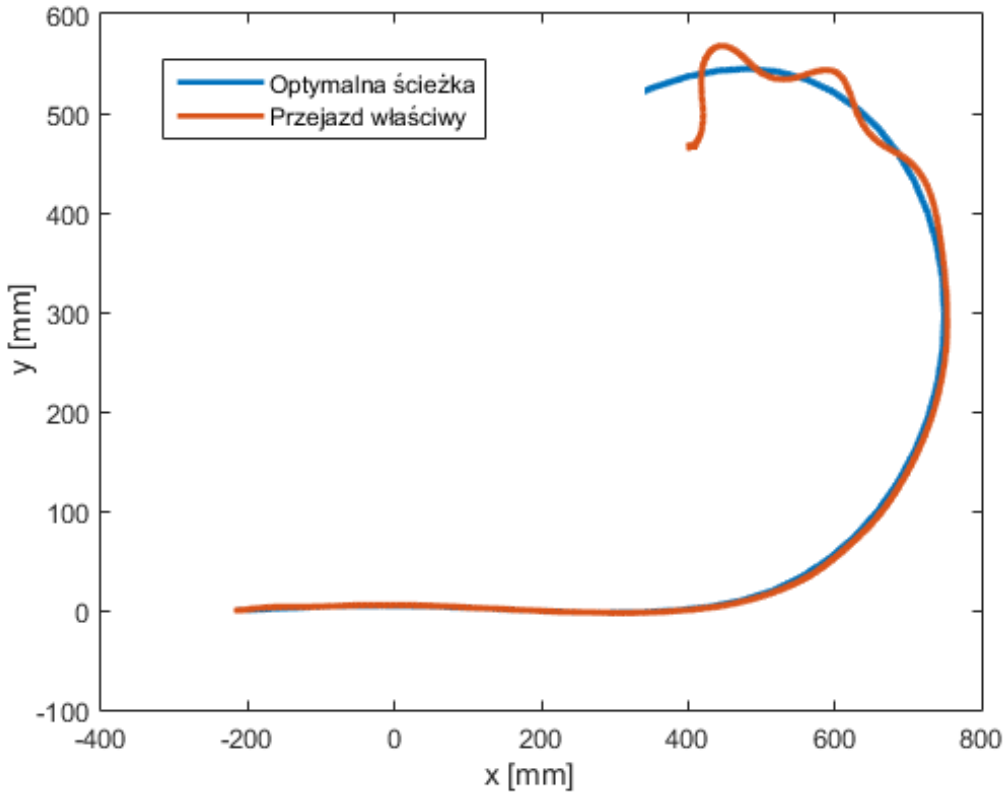


Rysunek 7.5: Porównanie rezultatów dla drugiej trasy testowej

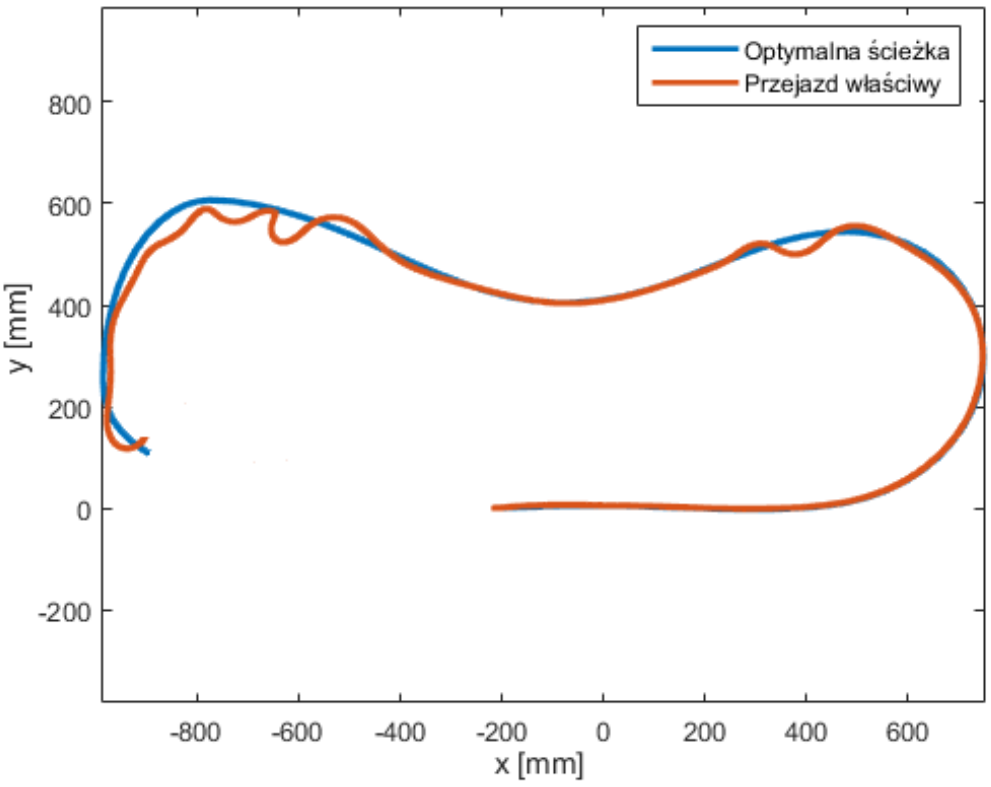
pomiarów z czujników różni się od tego, gdy zadawane jest nowe sterowanie na podstawie tychże danych. Kolejny problem stanowi czas reakcji jednostki DMP. Obrót – zwłaszcza dla dużych prędkości kątowych – uwidacznia się w pomiarach z pewnym opóźnieniem, co uniemożliwia efektywne wykorzystanie opracowanego rozwiązania i wymusza bardziej zachowawczą jazdę. Przykłady prób przejazdu po zadanej trajektorii z prędkością postępową robota $v_r = 0,4\frac{m}{s}$ zaprezentowano na rysunkach 7.7 i 7.8. Możemy zaobserwować rosnące oscylacje położenia robota względem trajektorii referencyjnej, gdy robot pokonuje zakręt.



Rysunek 7.6: Przejazd właściwy z uwzględnieniem układów realizujących zadania



Rysunek 7.7: Porównanie zadanej optymalnej ścieżki i wykonanego przejazdu



Rysunek 7.8: Porównanie zadanej optymalnej ścieżki i drugiego wykonanego przejazdu

Rozdział 8

Podsumowanie

Celem pracy było zaproponowanie algorytmów optymalizacji toru przejazdu robota klasy linefollower po trasie w najkrótszym możliwym czasie. Przedstawiono algorytmy wyznaczania najkrótszej ścieżki oraz ścieżki o minimalnej krzywiźnie. Następnie przygotowano środowisko do wykonania badań oraz porównano własności zaimplementowanych algorytmów. Przeprowadzone symulacje, zakładające stałą prędkość przejazdu, potwierdzają przewidywaną poprawę czasu przejazdu. Dysponując powolnym robotem optymalny jest przejazd po najkrótszej ścieżce. Uzyskany wynik jest o 16% lepszy względem ścieżki referencyjnej. Z kolei dla szybkiej konstrukcji optymalny tor ruchu to ścieżka o minimalnej krzywiźnie. Osiągnięty rezultat jest około 25% lepszy od przejazdu referencyjnego. Dodatkowo pokazano, że w przypadku krętych tras minimalizacja krzywizny toru ruchu może być jedynym sposobem na przejazd z dużą prędkością.

Przeprowadzone badania pozwalają na sformułowanie zasad postępowania o różnym stopniu złożoności w praktycznym aplikowaniu opisywanych rozwiązań, które to zasady można zapisać w postaci następujących punktów.

1. Dla powolnej konstrukcji zalecany jest przejazd po najkrótszej ścieżce.

2. Dla szybkiego robota pierwszą metodą optymalizacji jest przejazd po referencyjnej trasie z nadrzędnym profilerem prędkości, który efektywnie wykorzystuje długie proste przyspieszając na nich i zapewnia utrzymanie na trasie przez redukcję prędkości przed ciasnymi zakrętami. Algorytm opisano w pracy [7]. Bazuje na informacji o pokonanym dystansie.

3. Kolejną propozycją optymalizacji dla szybkiej konstrukcji jest przejazd po ścieżce o minimalnej krzywiźnie. To rozwiązanie zaprezentowane w niniejszej pracy charakteryzuje się większą złożonością, ponieważ wymaga iteracyjnego wyznaczania położenia względem punktu startowego, jednakże pozwala na szybsze pokonywanie zakrętów kosztem ewentualnego wydłużenia trasy.

4. Ostatnią propozycją jest połączenie dwóch poprzednich metod. Algorytm wyznaczałby trajektorię w dwóch etapach. Pierwszy polega na wyznaczeniu ścieżki o minimalnej krzywiźnie a drugi na optymalizacji profilu prędkości.

Realizacja proponowanych optymalizacji na rzeczywistym robocie była utrudniona. Metoda zakładała wykonywanie obliczeń na komputerze, w tym generowanie optymalnej ścieżki i zdalne sterowanie robotem w trakcie właściwego przejazdu. Metoda ta nie sprawdza się z powodu opóźnień w transmisji danych między robotem a komputerem. Ponadto,

informacja o orientacji robota z jednostki inercyjnej z racji małej częstotliwości generowania kolejnych pomiarów sprawdza się jedynie dla wolnych przejazdów. Zaobserwowano także opóźnienia w reakcji jednostki dla większej prędkości kątowej robota i chwilowe znaczne różnice między faktyczną a wyliczoną orientacją robota.

Proponowane dalsze działania, które mogą rozwiązać zaprezentowane problemy to nowy sposób wyznaczania orientacji robota, który uwzględniałby zarówno pomiary z jednostki inercyjnej, a także odczyty z enkoderów poprzez na przykład filtr Kalmana lub Magwicka. Kolejnym krokiem jest wyposażenie robota w szybszy mikrokontroler z jednostką zmiennoprzecinkową oraz większą pamięcią, aby pomieścić dane o wygenerowanej trajektorii i uporać się z obliczeniami w ograniczonym czasie, po to, by można było zrezygnować z komunikacji między komputerem a robotem w trakcie przejazdu. Komunikacja ograniczałaby się jedynie do wysłania przed startem optymalnej trajektorii przejazdu wyznaczonej na komputerze.

Bibliografia

[1] Robotic Arena 2017 - Regulamin konkurencji Linefollower Turbo. www.roboticarena.pl/static/ra_site/regulaminy/LF_Turbo.pdf

[2] Robotex 2017 - Regulamin konkurencji Line following. www.robotex.ee/wp-content/uploads/2017/06/robotex_line_following_rules_ENG.pdf

[3] R.S. Sharp D. Casanova. On minimum time vehicle manoeuvring: the theoretical optimal time. Praca doktorska, Uniwersytet Cranfield, 2000.

[4] Trójwymiarowy symulator wyścigów samochodowych TORCS www.wikipedia.org/wiki/TORCS

[5] L. Cardamone, D. Loiacono, P.L. Lanzi, A.P. Bardelli. Searching for the Optimal Racing Line Using Genetic Algorithms. *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, s. 388-394, 2010.

[6] Baton. Algorytm linefollowera w C – dla początkujących i nie tylko. www.forbot.pl/blog/algorytm-linefollowera-c-poczatkujacych-id2722

[7] W. Lipieta. Badania porównawcze algorytmów śledzenia trasy dla robotów klasy linefollower. Praca magisterska, Politechnika Wrocławska, 2017.

[8] Mikrokontroler STM32F103RCT6. Dokumentacja techniczna. https://www.st.com/resource/en/datasheet/cd00191185.pdf

[9] Moduł Bluetooth HC-06. Dokumentacja techniczna. www.olimex.com/Products/Components/RF/BLUETOOTH-SERIAL-HC-06/resources/hc06.pdf

[10] Silnik Pololu HPCB 10:1 z przedłużonym wałem. Dokumentacja techniczna. www.pololu.com/product/3071

[11] Enkoder AMS AS5040. Dokumentacja techniczna. www.ams.com/documents/20143/36005/AS5040_DS000374_3-00.pdf

[12] Układ InvenSense MPU-6050. Dokumentacja techniczna. www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[13] Transoptor odbiciowy KTIR0711s. Dokumentacja techniczna. www.farnell.com/datasheets/2307824.pdf

[14] K. Tchoń, A. Mazur, I. Dulęba, R. Hossa, R. Muszyński. Manipulatory i roboty mobilne: modele, planowanie ruchu, sterowanie. Problemy Współczesnej Nauki, Teoria i Zastosowania. Robotyka. Akademicka Oficyna Wydawnicza PLJ, 2000.

[15] F. Braghin, F. Cheli, S.Melzi, E. Sabbioni. Race driver model. *Computers and Structures*, s. 1503 – 1516, 2008.

[16] N. Poeiro, P. Pugliese. Sviluppo di un modello di pilota da corsa. 2005.

[17] A. Mazur. Sterowanie oparte na modelu dla nieholonomicznych manipulatorów mobilnych. Oficyna Wydawnicza Politechniki Wrocławskiej, 2009.