Sprawozdanie

STERO

Lab 1

Maciej Groszyk, Mateusz Zembroń

Interpolacja liniowa sterowania pozycyjnego na podstawie wyłącznie zadawanych prędkości

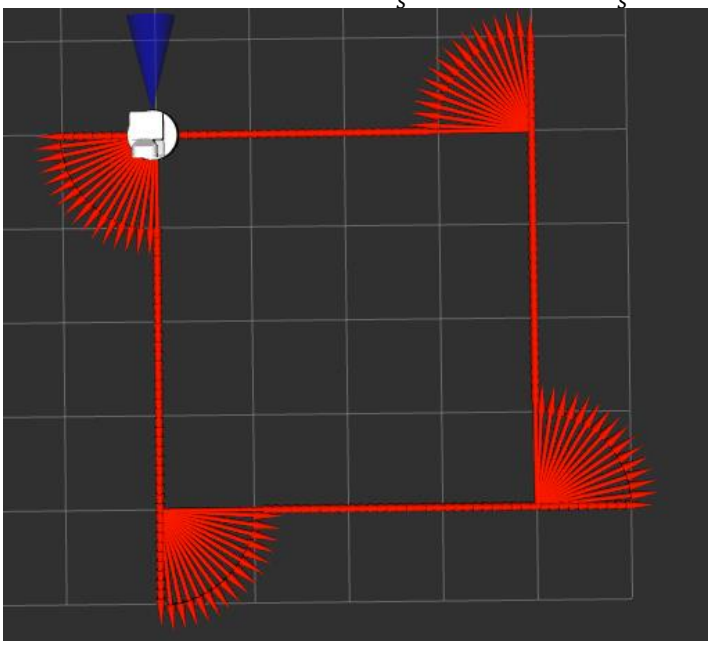
- Program (*„sq_publisher.py”*)

Napisany program zadawał prędkość naprzemiennie liniową oraz kątową w taki sposób aby uzyskać ruch robota po kwadracie. W tej wersji prędkość była zadawana niezależnie od tego gdzie znajdował się robot (czas ruchu w każdym etapie był założony z góry). Dane były publikowane na temacie *„key_vel”’* przez stworzony węzeł o nazwie *„stero_vel_pub”*.

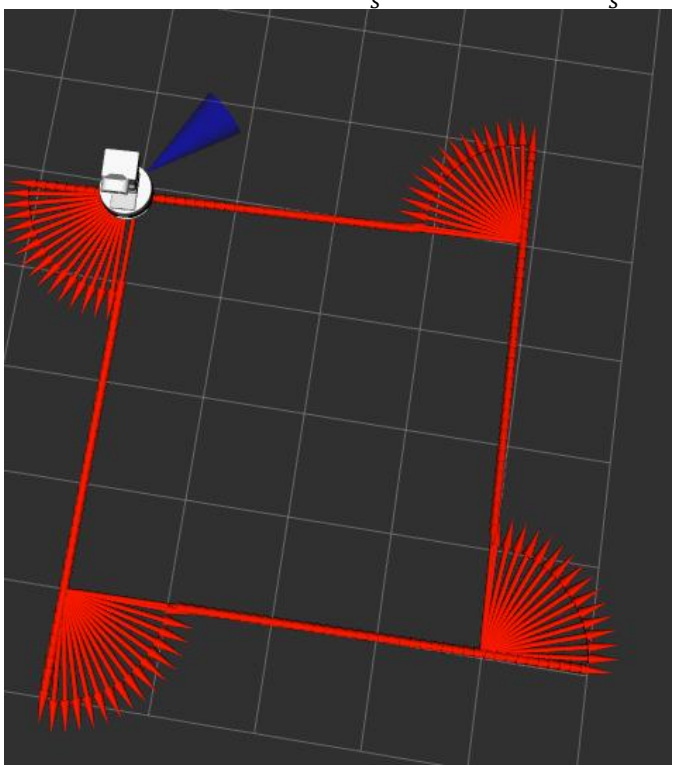
- Testy

Program sprawdzono dla różnych zestawów prędkości (liniowych i kątowych). Efekty działania programu można zaobserwować m.in. w programie symulacyjnym RViz:

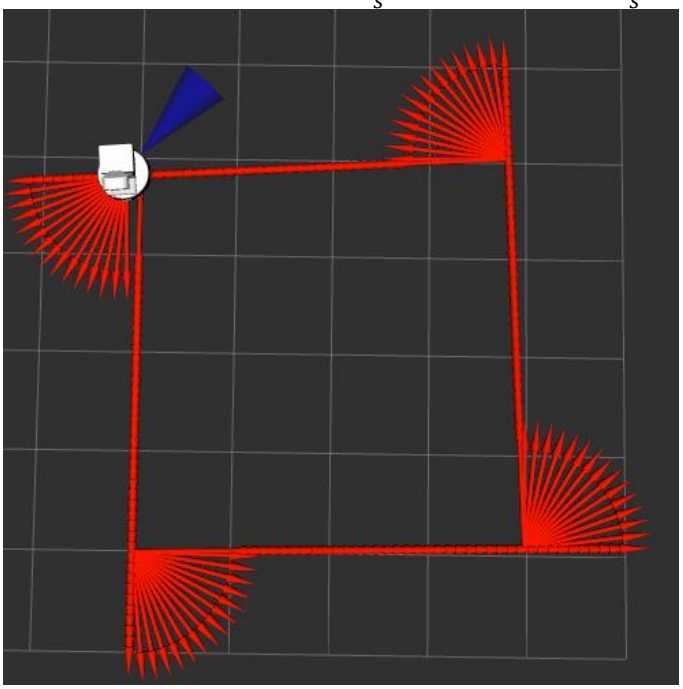
Prędkość kątowa: $0{,}05\frac{rad}{s}$, liniowa $0{,}12\frac{m}{s}$



Prędkość kątowa: $0{,}12\frac{rad}{s}$, liniowa $0{,}288\frac{m}{s}$

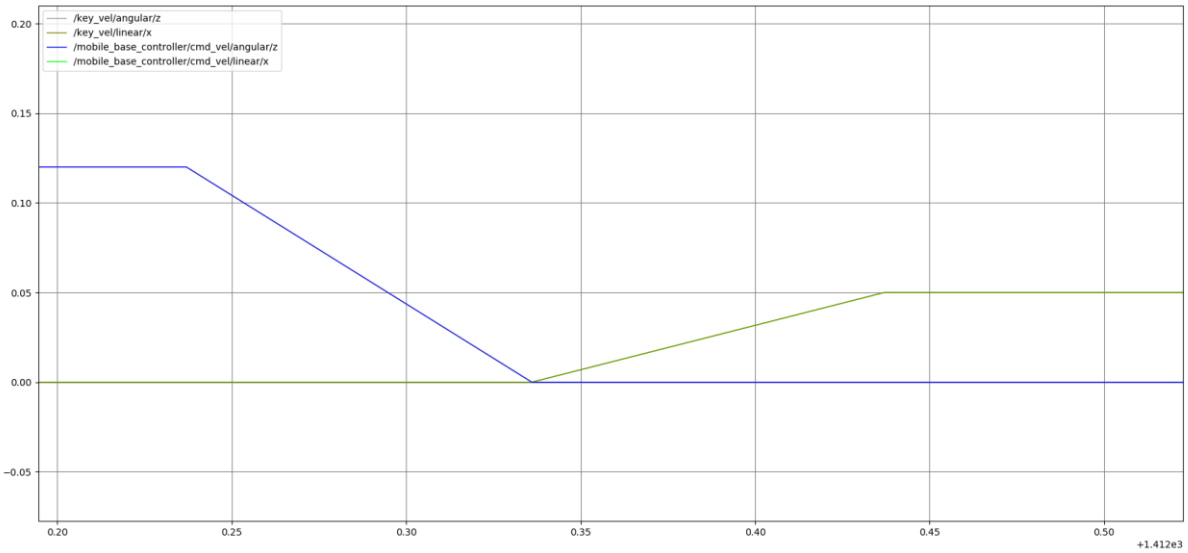


Prędkość kątowa: $0{,}288\frac{rad}{s}$, liniowa $0.6912\frac{m}{s}$

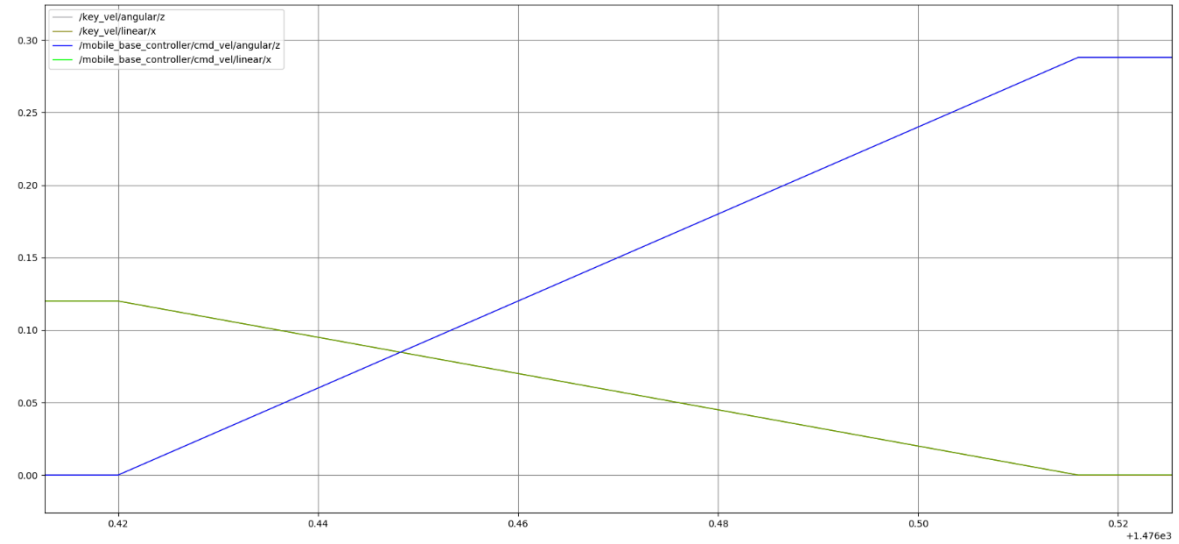


Sprawdzono również zgodność prędkości zadanej w programie z prędkością symulowanego robota. Poniżej zamieszczono przebiegi tych wartości podczas zmiany trybu ruchu robota z jazdy prosto na obrót:

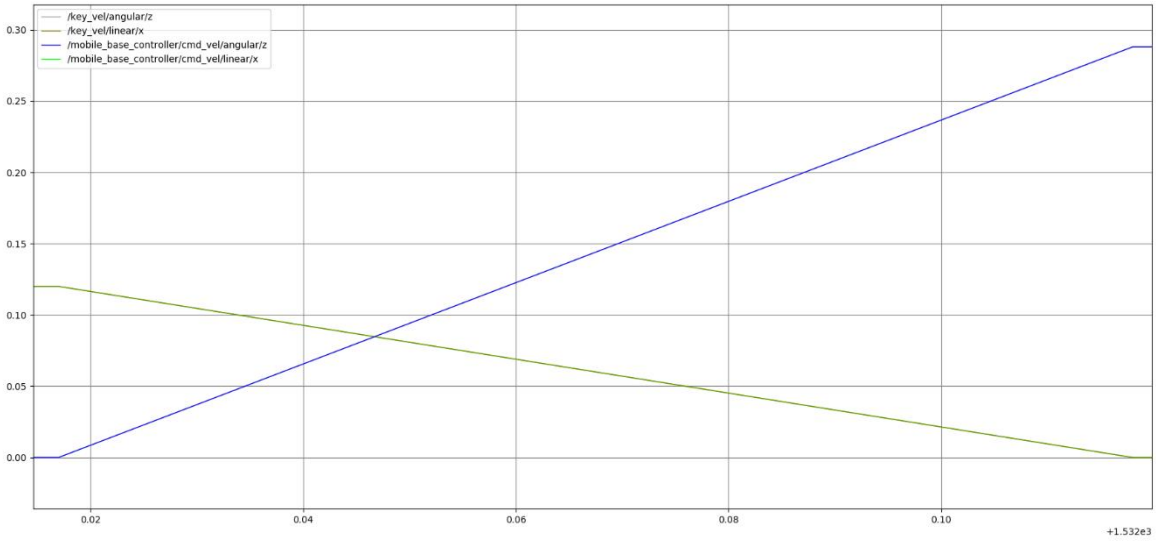
Prędkość kątowa: $0{,}05\,\frac{rad}{s}$, liniowa $0{,}12\,\frac{m}{s}$



Prędkość kątowa: $0{,}12\,\frac{rad}{s}$, liniowa $0{,}288\,\frac{m}{s}$



Prędkość kątowa: $0{,}288\frac{rad}{s}$, liniowa $0.6912\frac{m}{s}$



Uwaga!
Na przebiegu oznaczono kolorem:
- szarym – zadawaną prędkość kątową
- ciemnozielonym – zadawaną prędkość liniową
- niebieskim – odczytaną liniową kątową robota
- jasnozielonym – odczytaną liniową prędkość robota

! Na zdjęciach widoczne są jedynie dwa przebiegi, ponieważ prędkości zadane i odczytane zgadzały się ze sobą z dużą dokładnością!

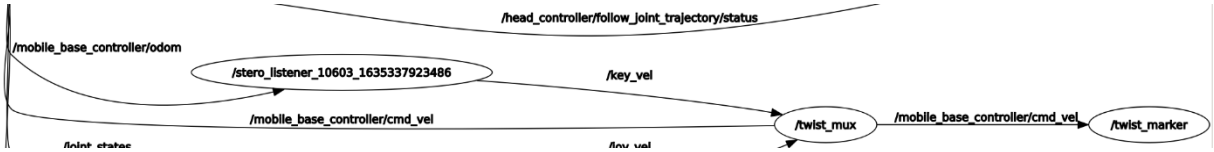
Zgodnie z symulacjami, w trybie interpolacji liniowej punktów na podstawie wyłącznie zadawanej prędkości, robot poruszał się z bardzo dużą dokładnością zgodnie z jego zamierzoną trajektorią. Jednak taka forma ruchu jest mało odporna na jakiekolwiek zmiany środowiska – brak sprzężenia zwrotnego położenia robota.

Interpolacja liniowa sterowania pozycyjnego na podstawie danych z odometrii

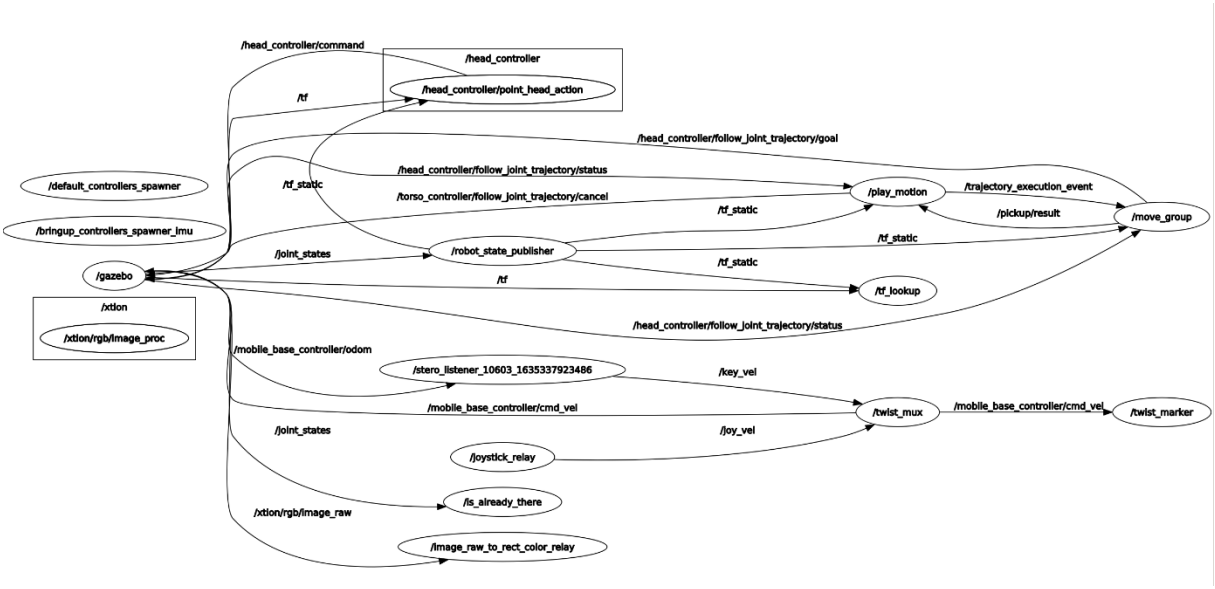
- Program (*„odometry_node.py”*)

Napisany program opiera się o węzeł *„stero_listener”* odczytujący aktualną pozycje robota (poprzez subskrypcję na temacie *„mobile_base_controller/odom”*). Zaimplementowany algorytm decyduje o zadaniu prędkości kątowej lub liniowej w zależności od odczytanej odległości od ostatniego rogu interpolowanego kwadratu. Następnie prędkość zadawana jest robotowi poprzez publikację danych na temacie *„key_vel”.*

Struktura programu wygenerowana za pomocą *„rqt_graph”*



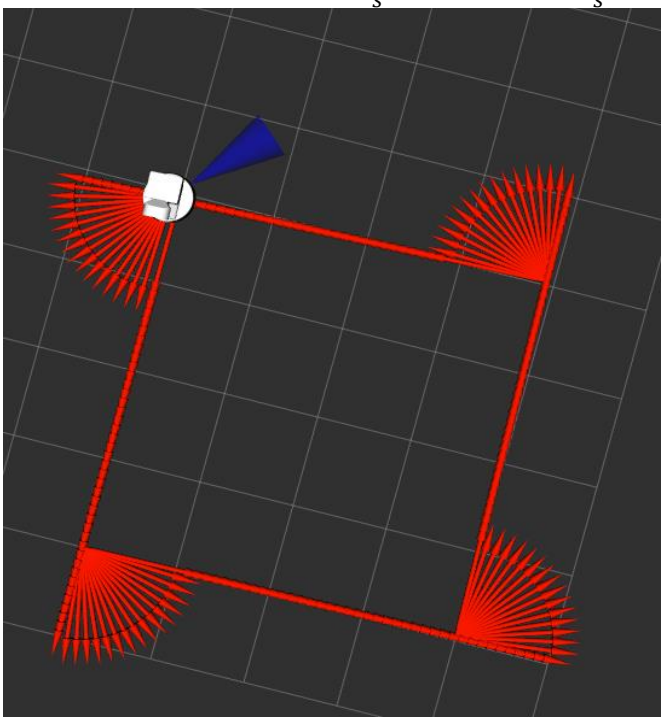
Struktura całego systemu (wygnereowana w ten sam sposób)



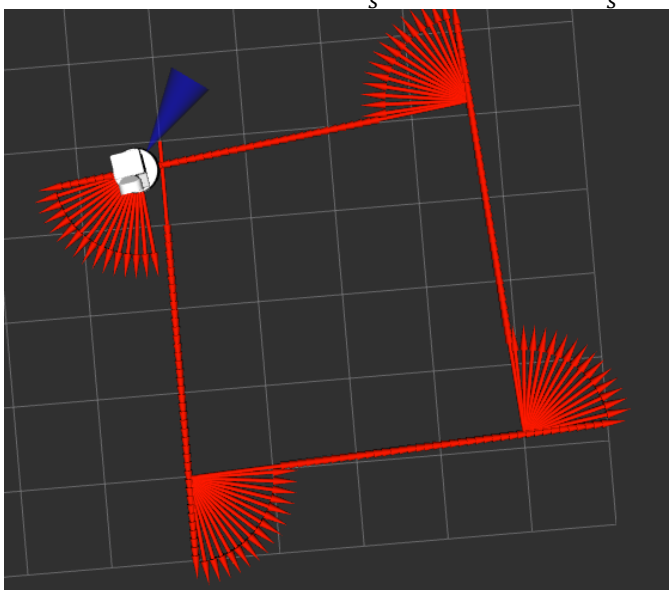
- Testy

Działanie programu sprawdzono jak poprzednio za pomocą symulacji. Efekty interpolacji w programie RViz:

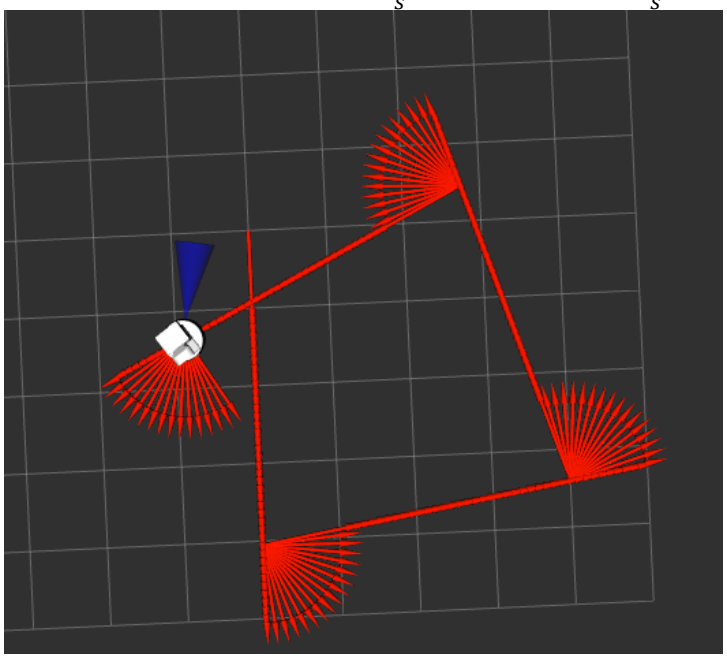
Prędkość kątowa: $0{,}05\,\frac{rad}{s}$, liniowa $0{,}12\,\frac{m}{s}$



Prędkość kątowa: $0{,}12\,\frac{rad}{s}$, liniowa $0{,}288\,\frac{m}{s}$

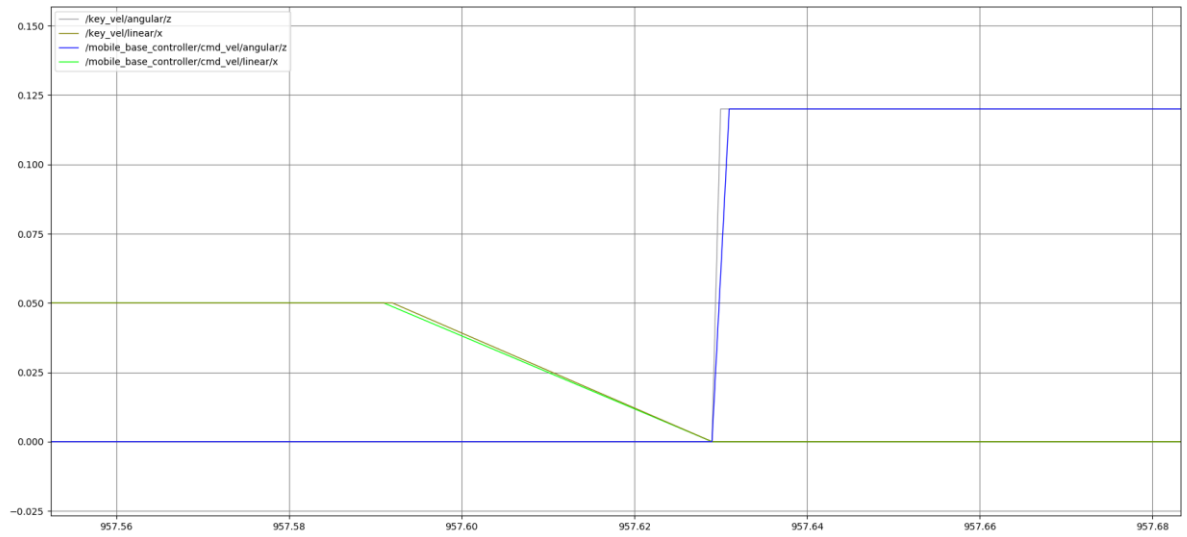


Prędkość kątowa: $0{,}288\,\frac{rad}{s}$, liniowa $0.6912\,\frac{m}{s}$

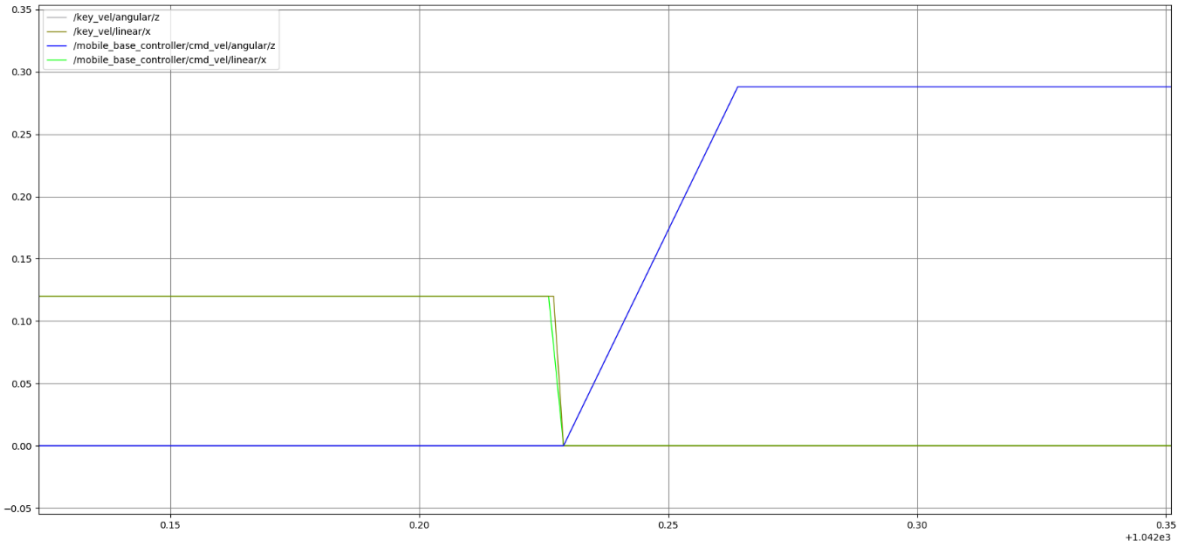


Analogicznie jak wcześniej sprawdzeniu podlegała zgodność zadanej prędkości z faktyczną prędkością robota:

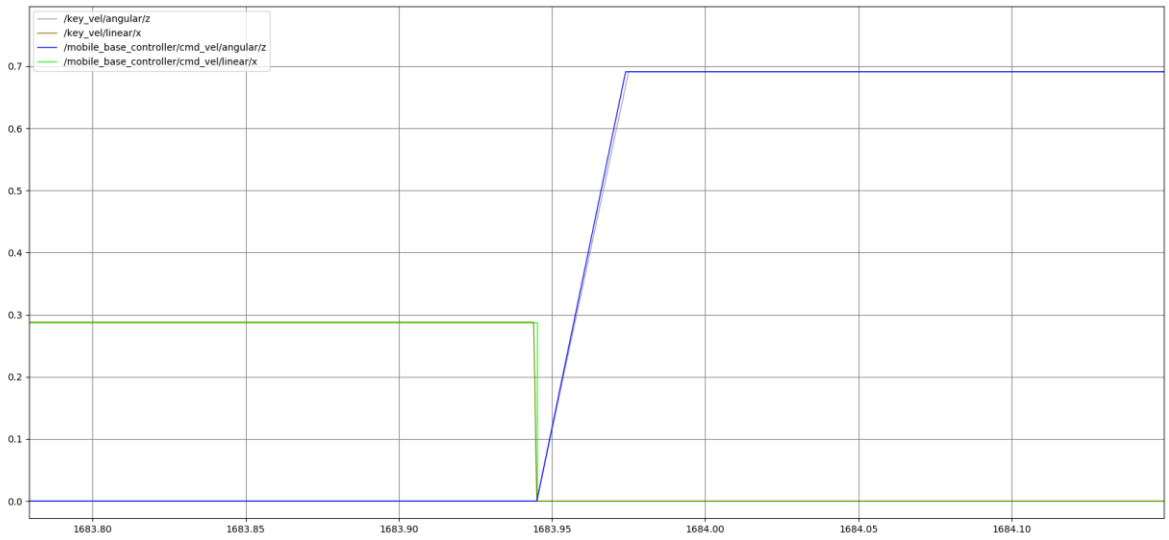
Prędkość kątowa: $0{,}05\,\frac{rad}{s}$, liniowa $0{,}12\,\frac{m}{s}$



Prędkość kątowa: $0{,}12\,\frac{rad}{s}$, liniowa $0{,}288\,\frac{m}{s}$



Prędkość kątowa: $0{,}288\frac{rad}{s}$, liniowa $0.6912\frac{m}{s}$

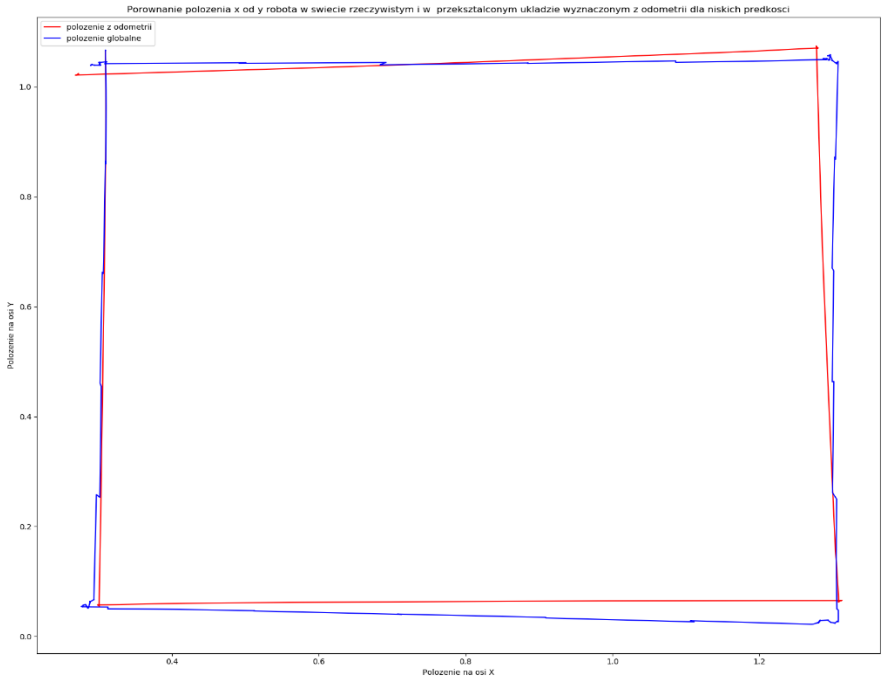


Uwaga!
Na przebiegu oznaczono kolorem:
- szarym – zadawaną prędkość kątową
- ciemnozielonym – zadawaną prędkość liniową
- niebieskim – odczytaną liniową kątową robota
- jasnozielonym – odczytaną liniową prędkość robota

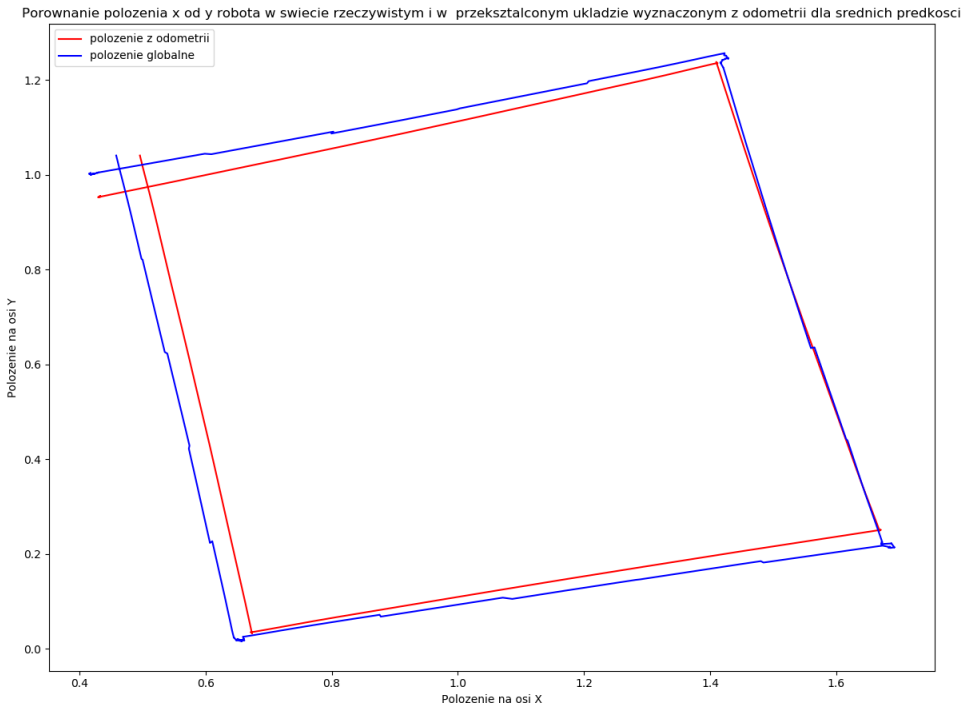
Obserwując przebiegi można zauważyć, że występowało nieznaczne opóźnienie między zadaną prędkością, a faktyczną prędkością robota. Większą różnicę widać w symulowanym ruchu robota. Przy wysokiej prędkości interpolacja kwadratu była mocno zniekształcona.

Dodatkową formą sprawdzenia algorytmu było przetestowanie go na realnym robocie Tiago. Robot miał za zadanie przejechać po interpolowanym kwadracie o boku długości 1m. Podczas testów nagrano lokalizacje robota na podstawie samej odometrii oraz fuzji różnych czujników(położenie globalne). Porównanie ich widoczne jest na wykresach poniżej:

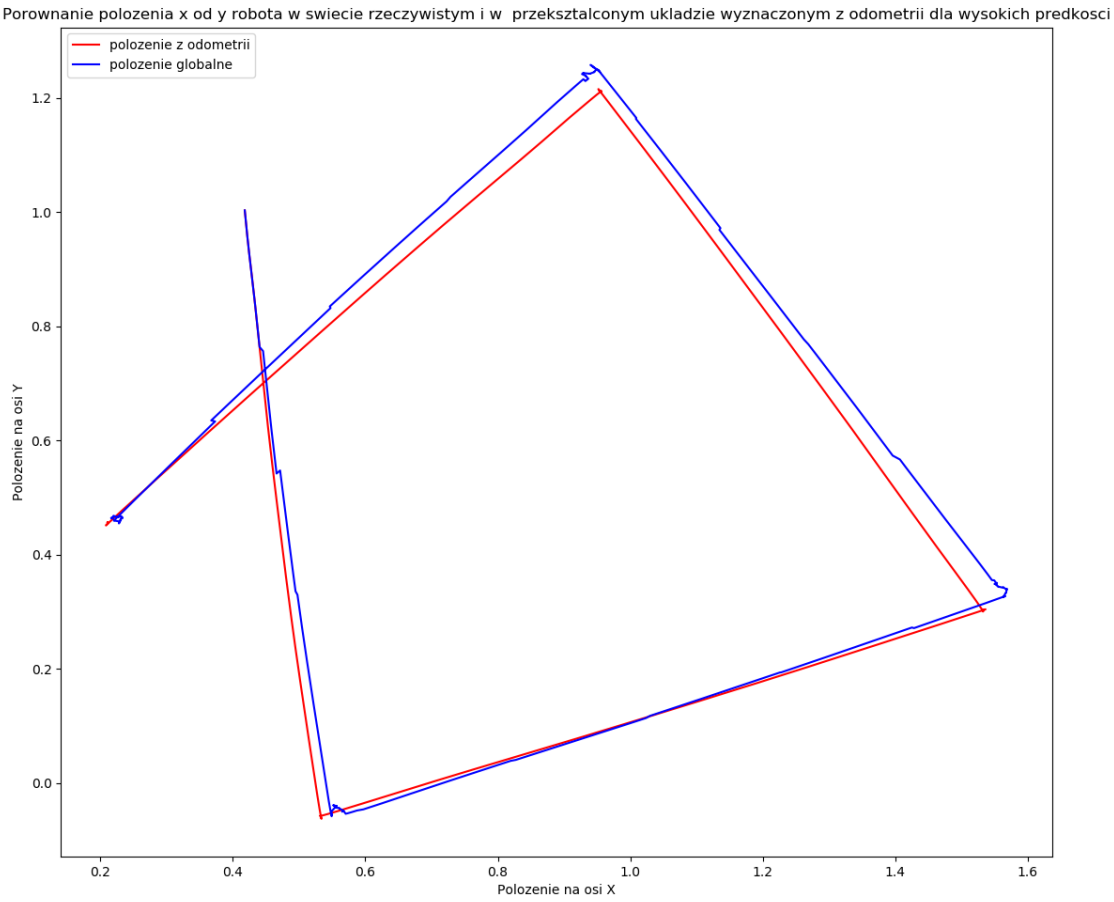
Prędkość kątowa: $0{,}05\,\frac{rad}{s}$, liniowa $0{,}12\,\frac{m}{s}$



Prędkość kątowa: $0{,}12\,\frac{rad}{s}$, liniowa $0{,}288\,\frac{m}{s}$



Prędkość kątowa: $0{,}288\frac{rad}{s}$, liniowa $0.6912\frac{m}{s}$



Dla każdego z przejazdów wyznaczono skumulowaną sumę modułów błędów oraz błąd średni.

Błąd średni został wyliczony zgodnie ze wzorem:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y - \gamma|$$

Gdzie:

- y – położenie robota z odomertrii
- γ – położenie robota w globalnym układzie współrzędnych
- n – liczba próbek

Tabela błędów uzyskanych w przejazdach

	Błąd całkowity			Średni błąd		
Prędkości w przejazdach	Położenia X	Położenia Y	Rotacji Z	Położenia X	Położenia Y	Rotacji Z
$L = 0{,}05\frac{rad}{s}$ A= $0{,}12\frac{m}{s}$	118,622m	134,92m	12074°	0,016m	0,018m	1,66°
$L = 0{,}12\frac{rad}{s}$ A= $0{,}288\frac{m}{s}$	106,1m	82,875m	4177°	0,03m	0,023m	1,18°
$L = 0{,}288\frac{rad}{s}$ A= $0{,}6912\frac{m}{s}$	21,19m	29,29m	1403,5°	0,011m	0,015m	0.37°

Wnioski

W realnych zastosowaniach rzadko można użyć ustalonego schematu ruchu. Z tego powodu, pomimo, iż algorytm oparty na odometrii gorzej poradził sobie z postawionym zadaniem to ma on o wiele szersze zastosowanie. Sama odometria w porównaniu z położeniem pochodzącym z fuzji czujników dawała bardzo zbliżone do niego rezultaty. Świadczy to o wysokiej jakości