AUTOMATYKA • 2010 • Tom 14 • Zeszyt 3/1

Maciej Garbacz*, Mieczysław Zaczyk*

Algorytmy rozmyte w nawigacji kołowego robota mobilnego

1. Wprowadzenie

W artykule zostanie przedstawiona implementacja algorytmu wykorzystującego logikę rozmytą do realizacji ruchu kołowego robota mobilnego Khepera III w nieznanym otoczeniu, z omijaniem występujących w nim przeszkód. Do realizacji tego zadania wykorzystane zostało środowisko programowe Matlab/Simulink [7]. Aby robota można było traktować jako jednostkę autonomiczną, konieczne jest wyposażenie go w czujniki zbierające informacje o otoczeniu [3]. Do prawidłowego działania algorytmu realizującego określony cel konieczna jest znajomość przestrzeni otaczającej robota. W opisanej w artykule aplikacji dla rozpoznawania otoczenia zostały wykorzystane czujniki zbliżeniowe podczerwieni. Przedstawiony algorytm ma za zadanie realizację ruchu robota z omijaniem występujących w obszarze roboczym przeszkód. Bardziej zaawansowane algorytmy nawigacji umożliwiają zadawanie robotowi dowolnego dopuszczalnego położenia w przestrzeni, bądź realizację poleceń typu „jedź wzdłuż ściany" czy „podążaj środkiem wolnej przestrzeni", „idź do celu" [5]. W zadaniach nawigacji możliwe jest również wykorzystywanie dodatkowych urządzeń zewnętrznych, takich jak np. kamera umieszczona ponad obszarem roboczym. Możliwe jest wówczas zbudowanie mapy otoczenia w oparciu o informacje uzyskane z obrazu z takiej kamery. Zaawansowane zagadnienia nawigacji wymagają zaangażowania sztucznej inteligencji. Nadrzędny układ decyzyjny musi odpowiednio reagować na pojawiające się nieraz sprzeczne ze sobą sygnały z podsystemów pomiarowych. Zaprezentowany w niniejszej pracy algorytm 'rozmyty' bazuje jedynie na informacjach pochodzących z czujników zbliżeniowych. Robot na bieżąco śledzi sytuację w otaczającej go przestrzeni, dzięki czemu zdolny jest do reakcji w przypadku zagrożenia kolizji np. z drugim poruszającym się robotem.

* Katedra Automatyki, Akademia Górniczo-Hutnicza w Krakowie

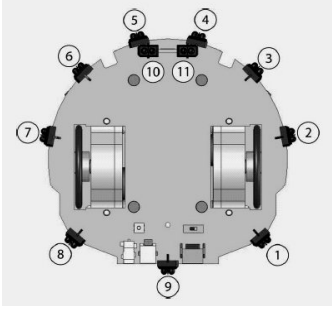
2. Robot Khepera III

2.1. Budowa

Robot (rys. 1) porusza się na dwóch kółkach, umieszczonych w jednej osi, obleczonych gumą dla uzyskania lepszej przyczepności. Dodatkowo Khepera III podparty jest w jednym punkcie, co zapewnia stabilne poruszanie się w przestrzeni roboczej. Dookoła obudowy robota umieszczonych jest 9 czujników podczerwieni oraz dodatkowo 2 pod spodem (czujniki zbliżeniowe), umożliwiających wykrywanie na niedaleką odległość przeszkód, bądź krawędzi stołu, po którym robot się porusza. Robot ma również możliwość zmierzenia odległości od przeszkody, za sprawą pięciu wbudowanych sensorów ultradźwiękowych (sonary). Do napędu wykorzystano dwa wysokiej klasy silniki DC (po jednym dla każdego koła) zapewniające sprawne i dokładne sterowanie ruchem robota. Obydwa koła robota napędzane są silnikami DC sprzężonymi z przekładnią o przełożeniu 43,2:1. Silniki maja własne wbudowane enkodery przyrostowe, umiejscowione na osi silnika, dające 16 impulsów na obrót wału silnika. W sumie daje to 2764,8 impulsów na obrót koła, co odpowiada 21,47 impulsom na milimetr przejechanej drogi (średnica koła wynosi 41 mm, co daje 128,8 mm przejechanej drogi na pełny obrót tarczy koła). Maksymalna osiągalna prędkość robota wynosi 298 mm/s [6].

Rys. 1. Widok robota Khepera III

Rozmieszczenie czujników zbliżeniowych przedstawia rysunek 2. Czujniki zbliżeniowe nie dają informacji o odległości od przeszkody, jedynie o jej bliskości (odczyt rośnie w miarę zbliżania się do przeszkody). Czujniki podczerwieni mogą również pracować jako czujniki światła [6].



Rys. 2. Rozmieszczenie czujników zbliżeniowych (widok od spodu)

Podstawowym sposobem komunikacji pomiędzy komputerem a robotem mobilnym Khepera III jest protokół komunikacji szeregowej RS232. Podczas takiej komunikacji komputer pracuje jako 'master' a robot Khepera jako 'slave'. Każde połączenie z robotem jest inicjowane przez komputer, a komunikacja realizowana jest poprzez przesyłanie komunikatów ASCII. Każde pojedyncze połączenie składa się z dwóch części:

– rozkazu wysyłanego z komputera: rozpoczynającego się dużą literą, po której następują (jeśli są konieczne) numeryczne lub znakowe parametry oddzielone przecinkiem;
– odpowiedzi, wysyłanej z robota do komputera: rozpoczynającej się małą literą (taką jak w rozkazie), po której następują (jeżeli rozkaz dotyczy odczytu stanu czujników) numeryczne parametry odpowiedzi oddzielone przecinkami.

Dostępne rozkazy można podzielić na dwie grupy:

– rozkazy dotyczące konfiguracji robota (ustawienie parametrów protokołu szeregowego, ustawianie parametrów regulatorów położenia i prędkości, ustawianie parametrów sonarów);
– rozkazy związane ze sterowaniem robota (zadawanie pozycji, zadawanie prędkości, odczyt czujników zbliżeniowych, odczyt czujników światła, odczyt odległości z sonarów).

Taki sposób komunikacji z robotem Khepera umożliwia programowanie przy użyciu dowolnego oprogramowania udostępniającego łączność przez port szeregowy COM. Na bazie udostępnionych dla robota rozkazów zrealizowany został zestaw funkcji pozwalających na programowanie w środowisku Matlaba [2]. Robot wyposażony jest standardowo w urządzenie Bluetooth, zapewniające bezprzewodową komunikację poprzez port szeregowy pomiędzy nim a komputerem sterującym.

2.2. Reprezentacja położenia robota

Przyjęto, że współrzędne pozycji Khepery odpowiadają lokalizacji punktu przecięcia osi robota z osią jego kół. Do nawigowania robotem nie wystarcza znajomość jego położenia. Aby móc skierować go w odpowiednim kierunku, trzeba również dysponować wiedzą o jego aktualnej orientacji. Dlatego współrzędne robota są reprezentowane przez 3 zmienne (x, y, α). Aktualna orientacja jest mierzona względem startowej w stronę przeciwną do ruchu wskazówek zegara. Przy wykonywaniu obrotów wokół pionowej osi robota koła przesuwają się po okręgu. Biorąc pod uwagę odległość między kołami robota, na pełny obrót przypada 5934,45 impulsów z enkoderów. Kierunek, w którym porusza się robot, wyznaczany jest z różnicy wartości wskazywanych przez enkodery na obu kołach:

$$\alpha = \frac{enk\_p - enk\_l}{5934,45} \cdot 180^{\circ}$$

Dla określenia aktualnej pozycji robota potrzebna jest wartość enkoderów przed rozpoczęciem i po zakończeniu ruchu. Pokonany dystans jest wyznaczany poprzez porównanie średnich arytmetycznych wskazań enkoderów przed rozpoczęciem i po zakończeniu ruchu.

$$dist = \frac{enk\_l_n + enk\_p_n}{2} - \frac{enk\_l_{n-1} + enk\_p_{n-1}}{2}$$

Aktualne położenie robota wyznaczane jest z zależności:

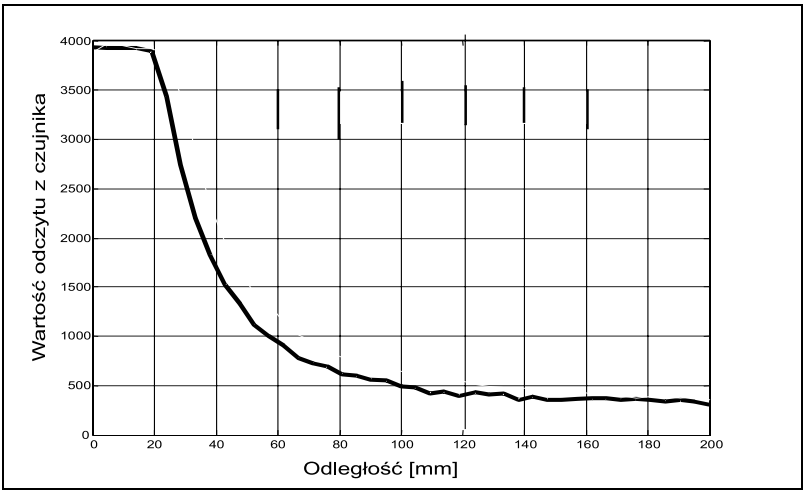
$$x_n = x_{n-1} + \frac{dist \cdot \cos(\alpha)}{21,47}$$

$$y_n = y_{n-1} + \frac{dist \cdot \sin(\alpha)}{21,47}$$

gdzie dzielenie przez 21,47 wynika z przeliczenia liczby impulsów z enkoderów na każdy milimetr przebytej drogi.

3. Implementacja algorytmu ruchu

Dla potrzeb implementacji i testowania algorytmów nawigacji i sterowania robota Khepera III w nieznanym otoczeniu wybrane zostało środowisko Matlab/Simulink (R2008b) [7] oraz *Fuzzy Logic Toolbox* [8]. Komunikacja z robotem wykonywana jest poprzez pakiet funkcji zrealizowanych dla tego środowiska [2]. W zaimplementowanym algorytmie 'rozmytym' dla oceny interakcji z otoczeniem wybrane zostały czujniki zbliżeniowe, których charakterystyka przedstawiona została na rysunku 3 (12-bitowe czujniki dają wartość od 0 do 4095, a odległość podana jest w milimetrach).

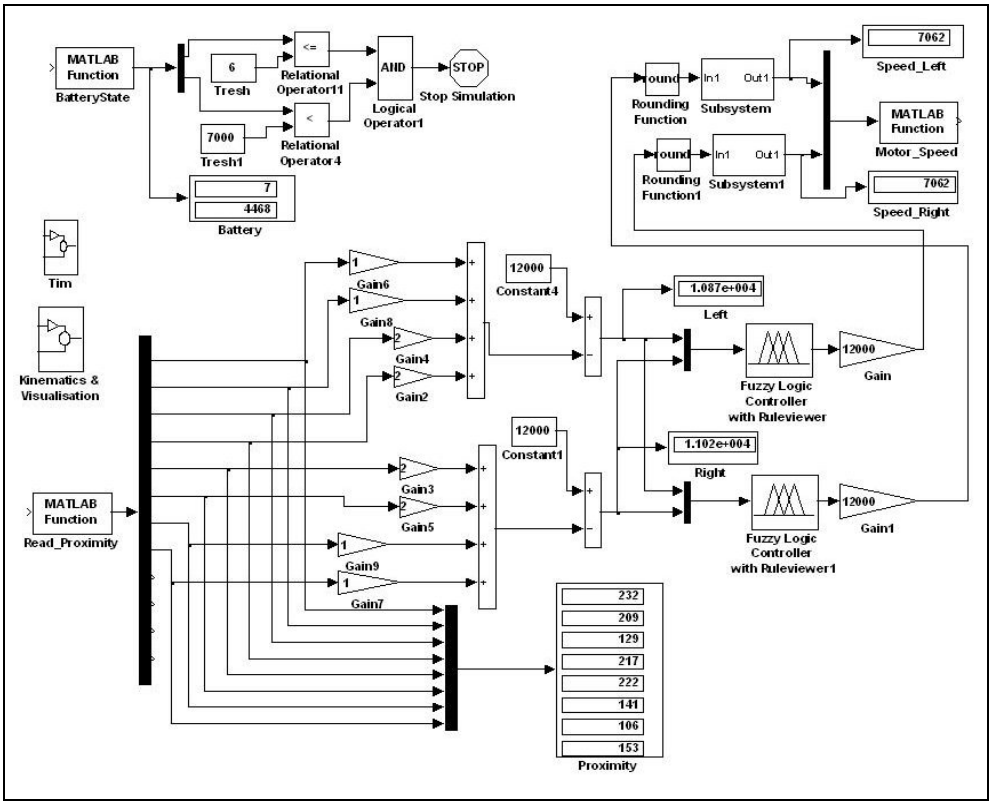


Rys. 3. Charakterystyka czujników zbliżeniowych

Poniżej przedstawiony został algorytm poruszania się robota Khepera III w nieznanym otoczeniu z omijaniem przeszkód wykorzystujący logikę rozmytą, bazujący na idei Braitenberga. Idea tzw. „pojazdu Braitenberga" [1] polega na bezpośrednim połączeniu modułów percepcji i wykonywania ruchu, czyli połączeniu czujników z elementami wykonawczymi (napędami). Każde takie połączenie ma przypisane wagi. W zależności od zastosowanych czujników i wag robot może wykonywać różne zadania.

3.1. Algorytm 'rozmyty'

Schemat blokowy algorytmu zaimplementowany w Simulinku przedstawiono na rysunku 4. Odczyt czujników zbliżeniowych realizowany jest przez blok *'Read_Proximity'* (w zaimplementowanym algorytmie wykorzystano czujniki od 1 do 8) a wysyłanie zadanej prędkości dla kół robota przez blok *'Motor_Speed'*. Bloki '*Fuzzy Logic Controller*' zawierają funkcje opisujące algorytm 'rozmyty'. W podsystemie *'Kinematics & Visualisation'* rozwiązywane jest zadanie kinematyczne dla uzyskania pozycji i orientacji robota oraz wizualizacji ruchu. Podsystem *'Tim'* zawiera s-funkcję zapewniającą działanie układu z zadanym czasem próbkowania.



Rys. 4. Schemat blokowy układu z algorytmem „rozmytym"

Regulatory rozmyte zostały stworzone za pomocą narzędzia „fuzzy" znajdującego się w pakiecie Fuzzy Logic Toolbox [8], będącym dodatkiem do oprogramowania Matlab/Simulink. Umożliwia ono ręczny dobór parametrów regulatora, czyli funkcji przynależności dla wejść i dla wyjść, reguł opisujących sposób działania regulatora, czy sposobu defuzyfikacji. W ten sposób utworzono regulatory typu Mamdani o dwóch wejściach i jednym wyjściu. Dla realizacji procesu defuzyfikacji wybrana została metoda środka ciężkości.

Zaimplementowany algorytm jest bardzo 'intuicyjny':

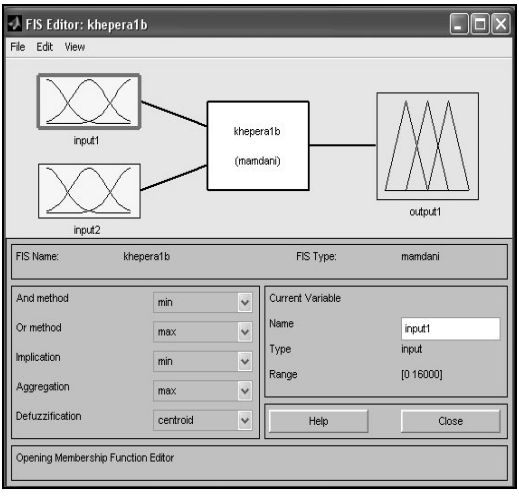
„jeżeli odległość od przeszkody z lewej strony robota jest mała a z prawej strony jest duża to robot powinien skręcać w prawo (prędkość lewego koła duża a prawego mała)"

„jeżeli odległość od przeszkody z lewej strony robota jest duża a z prawej strony jest mała to robot powinien skręcać w lewo (prędkość lewego koła mała a prawego duża)"

„jeżeli odległość od przeszkody z lewej strony robota jest duża i z prawej strony jest duża to robot powinien jechać prosto (prędkość lewego i prawego koła duża)".

Wydaje się, że dosyć naturalne jest przełożenie tak sformułowanego algorytmu na rzeczywiste sterowanie poprzez wykorzystanie logiki 'rozmytej'.

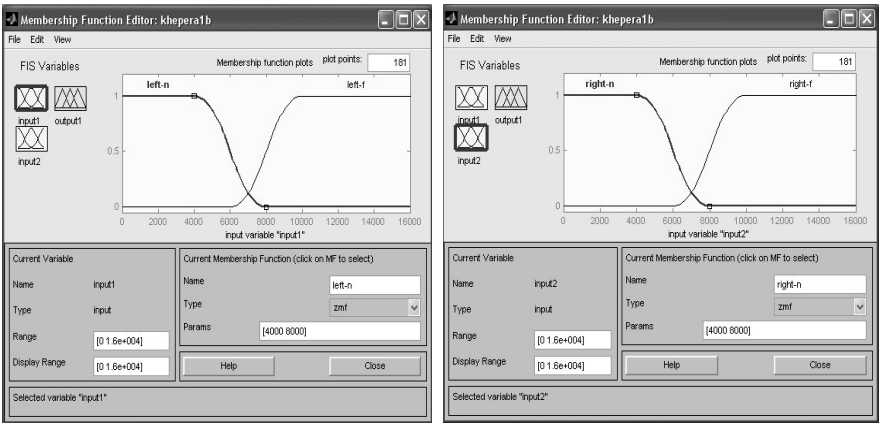
Uwzględniając charakterystyki czujników zbliżeniowych przyjęto (ze względu na zachowanie bezpiecznej odległości od przeszkód), że górny próg sumy wskazań czterech czujników z każdej strony robota będzie wynosił 12 000 (co oznacza odległość od przeszkody około 25 mm). Regulator rozmyty dla każdego koła posiada dwa wejścia powiązane z czujnikami z lewej i prawej strony robota oraz jedno wyjście określające prędkość koła (rys. 5).



Rys. 5. Struktura regulatora rozmytego

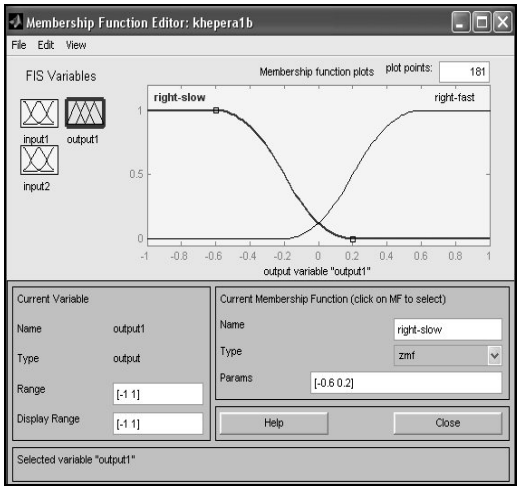
Dla opisu sygnałów wejściowych przyjęte zostały dwie funkcje przynależności (typu *ZMF* i *SMF*) przedstawione na rysunku 6.

Dla czujników z lewej strony są to funkcje: *'left-n'* (przeszkoda blisko) i *'left-f'* (przeszkoda daleko). Podobnie dla czujników z prawej strony: *'right-n'* i *'right-f'*.



Rys. 6. Funkcje przynależności dla wejść

Dla opisu sygnału wyjściowego przyjęto również dwie funkcje opisujące (typu *ZMF* i *SMF*). Przykładowo dla prawego koła (rys. 7) są to: *'right-slow'* (prawe koło wolno) i *'right-fast'* (prawe koło szybko).



Rys. 7. Funkcje przynależności dla wyjścia

Dla opisu działania regulatora ustalone zostały następujące proste reguły (dla prawego koła):

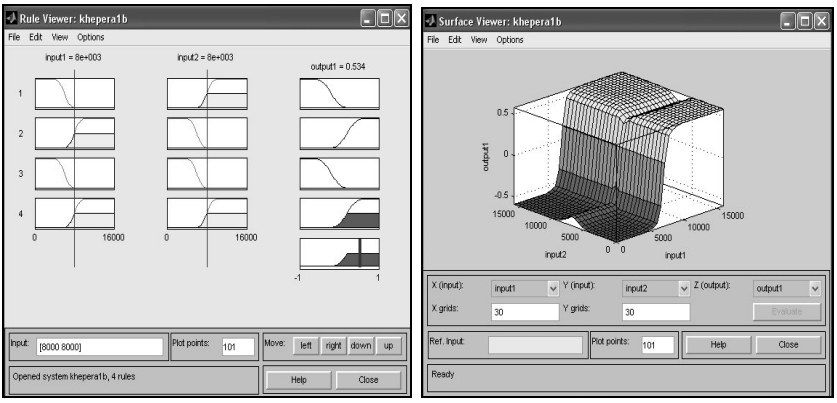
1. **If** (*input1* is *left-n*) **and** (*input2* is *right-f*) **then** (*output1* is *right-slow*)
2. **If** (*input1* is *left-f*) **and** (*input2* is *right-n*) **then** (*output1* is *right-fast*)
3. **If** (*input1* is *left-n*) **and** (*input2* is *right-n*) **then** (*output1* is *right-slow*)
4. **If** (*input1* is *left-f*) **and** (*input2* is *right-f*) **then** (*output1* is *right-fast*)

W podobny sposób ustalone zostały funkcje przynależności dla wejść i wyjścia regulatora lewego koła, a reguły jego działania są następujące:

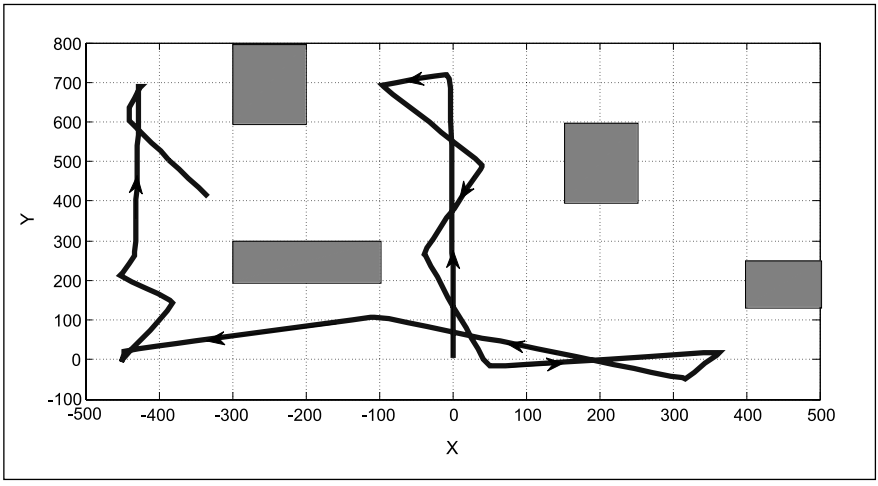
1. **If** (*input1* is *left-n*) **and** (*input2* is *right-f*) **then** (*output1* is *left-fast*)
2. **If** (*input1* is *left-f*) **and** (*input2* is *right-n*) **then** (*output1* is *left-slow*)
3. **If** (*input1* is *left-n*) **and** (*input2* is *right-n*) **then** (*output1* is *left-fast*)
4. **If** (*input1* is *left-f*) **and** (*input2* is *right-f*) **then** (*output1* is *left-fast*)

Dla tak ustalonych funkcji przynależności oraz przyjętych reguł działania uzyskana została zależność sterowania od sygnałów wejściowych przedstawiona na rysunku 8.

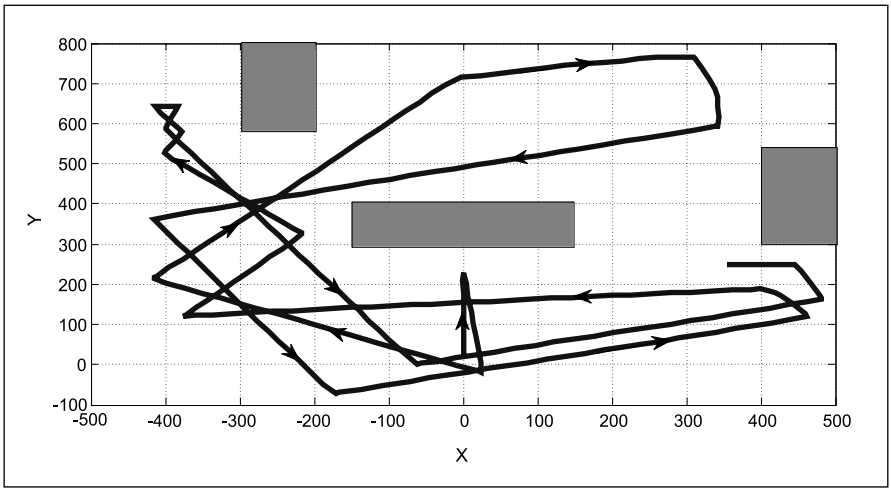
Przeprowadzono wiele testów potwierdzających poprawność działania algorytmu. Przykładowe przebiegi ruchu robota w nieznanym otoczeniu z wykorzystaniem przedstawionych regulatorów rozmytych pokazane zostały na rysunkach 9 i 10.



Rys. 8. Zmiany sygnału sterującego w funkcji sygnałów wejściowych



Rys. 9. Ilustracja działania algorytmu



Rys. 10. Ilustracja działania algorytmu

4. Podsumowanie

Przedstawione w pracy rozważania pokazują możliwości wykorzystania bardzo wygodnego i uniwersalnego środowiska Matlab/Simulink do implementacji i testowania algorytmów nawigacji i sterowania dla mobilnego robota kołowego Khepera III. Realizacja tego zadania była możliwa przez wykorzystanie stworzonego pakietu funkcji sterujących. Wybrane środowisko badawcze pozwala na łatwą zmianę i dobór parametrów algorytmów i testowanie wpływu tych zmian na zachowanie robota. Wykorzystanie logiki rozmytej dla implementacji algorytmu zachowania się mobilnego robota w nieznanym otoczeniu wydaje się bardzo naturalne i korzystne. Uzyskane wyniki testów przeprowadzonych dla zaprezentowanego algorytmu ruchu robota w nieznanym otoczeniu z omijaniem przeszkód można uznać za zadowalające.

Literatura

[1] Braitenberg V., *Vehicles: Experiments in synthetic psychology.* MIT Press., Cambridge, 1984.
[2] Garbacz M., Zaczyk M., *Robot mobilny Khepera III – oprogramowanie dla środowiska MATLAB.* Automatyka (półrocznik AGH), t. 12, z. 3, 2008, 759–767.
[3] Garbacz M., *Laboratoryjny robot mobilny Khepera II.* Automatyka (półrocznik AGH), t. 9, z. 3, 2005, 393–400.
[4] Giergiel J., Hendzel Z., Żylski W., *Kinematyka, dynamika i sterowanie mobilnych robotów kołowych*. PWN, Warszawa, 2002.

[5] Trojnacki M., Szynkarczyk P., *Autonomia robotów mobilnych – stan obecny i perspektywy rozwoju.* Pomiary Automatyka Robotyka, 9/2008, 5–9.
[6] *Khepera III User Manual*, ver. 2.1, K-Team S.A., Switzerland, 2008.
[7] *Matlab 7.7 User Guide*, The Mathworks Inc., 2008.
[8] *Fuzzy Logic Toolbox User Guide*, The Mathworks Inc., 2008.
[9] Piegat A., *Modelowanie i sterowanie rozmyte*. Akademicka Oficyna Wydawnicza EXIT, Warszawa, 1999.