Piotr PRZYSTAŁKA, Kamil POLOCZEK
INSTYTUT PODSTAW KONSTRUKCJI MASZYN, POLITECHNIKA ŚLĄSKA,
ul. Konarskiego 18a, 44-100 Gliwice

Szybkie prototypowanie sterownika rozmytego dla robota mobilnego

Dr inż. Piotr PRZYSTAŁKA

Adiunkt w Instytucie Podstaw Konstrukcji Maszyn Politechniki Śląskiej. Obszar zainteresowań naukowo-badawczych autora obejmuje: mechatronikę, automatykę i robotykę, diagnostykę techniczną, informatykę stosowaną. Głównym przedmiotem jego działalności naukowej jest zagadnienie stosowania metod i technik sztucznej inteligencji w sterowaniu i diagnozowaniu różnych klas obiektów technicznych.

e-mail: piotr.przystalka@polsl.pl

Mgr inż. Kamil POLOCZEK

Absolwent wydziału Mechanicznego Technologicznego Politechniki Śląskiej w Gliwicach. Studia magisterskie ukończył na kierunku Automatyka i Robotyka w Instytucie Podstaw Konstrukcji Maszyn. Tytuł mgra inż. obronił w roku 2011 pisząc pracę na temat zastosowania systemów rozmytych do sterowania robotów mobilnych. Obecnie pracuje w branży związanej z automatyka przemysłową.

e-mail: kamil.poloczek@o2.pl

Streszczenie

W artykule przedstawiono zastosowanie techniki szybkiego prototypowania podczas realizacji sterownika rozmytego do unikania kolizji robota mobilnego z przeszkodami. Opisano w jaki sposób można wykorzystać w tym celu środowiska Microsoft® Robotics Developer Studio (MRDS) oraz MATLAB®. Artykuł zawiera badania weryfikacyjne różnych przykładów implementacji sterownika rozmytego. Otrzymane wyniki testów potwierdzają poprawność zaproponowanego podejścia.

Słowa kluczowe: robotyka mobilna, szybkie prototypowanie systemów sterowania, systemy rozmyte, nawigacja robotów mobilnych.

Rapid prototyping of the fuzzy controller for a mobile robot

Abstract

The paper deals with the application of the rapid prototyping technique for control system modeling. The proposed methodology is applied to creating a fuzzy controller that can be used to avoid collisions of a wheeled mobile robot with obstacles. In the paper there is described how to combine and apply two well-known development environments, that is, Microsoft® Robotics Developer Studio and MATLAB® software for rapid prototyping purposes. Important components of the proposed framework such as Visual Programming Language, Visual Simulation Environment and Fuzzy Logic Toolbox are briefly discussed. The main part of the paper focuses on the implementation of a few variants of the fuzzy controller for managing the robot movements in the virtual world (Tab. 1). These controllers differ from each other in such properties as the type of a distance sensor and the number of rules included in the fuzzy knowledge base. The merits and limits of the elaborated solution are considered taking into account the results obtained during verification tests. These experiments were carried out with the use of several environments of different complexity. The results of the verification tests which are included in Tab. 2 show the effectiveness of the proposed approach.

Keywords: mobile robotics, rapid prototyping of control systems, fuzzy systems, mobile robot navigation.

1. Wprowadzenie

Robotyka mobilna to multidyscyplinarna dziedzina nauki i techniki, która obecnie bardzo dynamicznie poszerza swój obszar zastosowań. Za pomocą robotów mobilnych można nie tylko eksplorować trudno dostępny dla człowieka teren, lecz również rozbrajać materiały wybuchowe, poszukiwać i transportować rannych, monitorować strzeżony lub skażony teren oraz można realizować wiele innych operacji, w których człowiek jest mniej efektywny od maszyny [1, 7]. Dziedzina ta aktywnie rozwija się każdego dnia i na bieżąco powstają nowe aplikacje robotów do różnych zadań. Wiąże się z tym potrzeba rozwoju systemów, które pozwolą robotom stać się autonomicznymi lub choćby semi-autonomicznymi [6]. W tym celu coraz częściej wykorzystuje się szerokie spektrum metod i technik sztucznej inteligencji oraz miękkich obliczeń (ang. *soft computing*) takich jak: sztuczne sieci neuronowe, wnioskowanie probabilistyczne, systemy ekspertowe, systemy rozmyte, systemy neuronowo-rozmyte, algorytmy ewolucyjne, itp. Jednym z problemów występujących podczas sterowania robotem mobilnym jest zapewnienie jego bezkolizyjnego poruszania się w statycznym i/lub dynamicznym środowisku. Niniejsza praca jest próbą zastosowania systemów rozmytych oraz techniki szybkiego prototypowania do projektowania i implementacji kontrolera ruchu umożliwiającego bezkolizyjne sterowanie robotem w środowisku statycznym.

Istnieje wiele odmian systemów rozmytych, które wypracowano na gruncie logiki rozmytej zaproponowanej przez L. Zadeha [9] - tj. np. system Mamdaniego-Assilana, system Takagi-Sugeno-Kanga, system Tsukamoto, systemy rozmyte bazujące na zbiorach rozmytych typu 2 [5]. Analizując dostępne źródła literatury dotyczące sterowników robotów mobilnych bazujących na systemach rozmytych, można dostrzec powszechne zastosowanie metody wnioskowania Mamdaniego-Assilana. Jednym z ciekawych zastosowań tej techniki jest system sterowania robota mobilnego omijającego przeszkody i jadącego wg wcześniej zdefiniowanej trasy [3]. Innym przykładem wykorzystania systemu rozmytego do omijania przeszkód jest sterowanie omnikierunkowym robotem mobilnym posiadającym możliwość swobodnego poruszania się w dowolnym kierunku [8]. Tu robot sam wyznacza drogę, po której się przemieszcza aby dotrzeć do wyznaczonego celu. Inną z metod sterowania za pomocą systemu rozmytego jest wykorzystanie go do układu regulacji nadążnej [2]. W przytoczonym przykładzie zadaniem systemu rozmytego jest kompensacja nieliniowości sterowanego robota.

W dalszej części artykułu opisano metodykę tworzenia sterownika rozmytego dla kołowego robota mobilnego. Głównym założeniem opisanych tu badań było opracowanie takiego sposobu tworzenia sterownika robota, który pozwoli na minimalizację czasu potrzebnego na jego zaprojektowanie i implementację na docelowej platformie mobilnej.

2. Szybkie prototypowanie rozmytego sterownika dla robota mobilnego

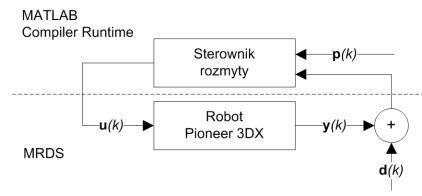
Prezentowana metodyka tworzenia sterownika bazuje na dwóch środowiskach programistycznych: Microsoft® Robotics Developer Studio (MRDS) [4] oraz MATLAB® [10]. Pierwsze z nich jest środowiskiem przeznaczonym do rozwoju algorytmów i systemów sterowania robotów mobilnych. Umożliwia łatwą instalację systemu sterowania na urządzeniach fizycznych oraz oferuje możliwość wcześniejszego przetestowania algorytmu sterującego w środowisku symulacyjnym. MRDS oferuje przykładowe wzorce programowe, które można odpowiednio modyfikować do własnych potrzeb. Taka metodyka postępowania pozwala zaoszczędzić wiele czasu podczas tworzenia oprogramowania. Dodatkowo MRDS posiada moduł Visual Programming Language (VPL) pozwalający programować i debugować programy w sposób intuicyjny. Wykorzystuje się w tym celu proste i konfigurowalne

bloki funkcyjne stanowiące tzw. serwisy. Programowanie taką metodą opiera się na przeciąganiu i upuszczaniu tych serwisów w przestrzeni programowania oraz ich odpowiednim łączeniu. Możliwe jest łączenie obu metod programowania, tzn. edycji już istniejących bloków funkcyjnych z programowaniem serwisu od podstaw. Gotowe oprogramowanie danego urządzenia przed wdrożeniem można przetestować w dostępnym środowisku symulacyjnym Visual Simulation Environment (VSE). Posiada ono pełne wsparcie wizualizacji 3D oraz obliczeń z uwzględnieniem praw fizyki. Pozwala to na wyeliminowanie błędów w algorytmie sterowania jeszcze na etapie wstępnej weryfikacji modelu robota. Środowisko symulacyjne można w pełni zaprogramować, tak aby odzwierciedlało w jak największym stopniu warunki występujące w rzeczywistości. Dotyczy to zarówno samego robota, jak i otoczenia w jakim ma on pracować. Istnieje również możliwość zastosowania gotowych szablonów w stosunku do robotów. MRDS posiada w swoich zasobach zdefiniowane klasy robotów o często wykorzystywanej strukturze mechanicznej, które po modyfikacji głównych własności oraz nadaniu odpowiedniego modelu wizualnego, stanowią obiekt symulacji odpowiadający pożądanemu celowi. Jeżeli taka opcja nie wystarcza, możliwa jest całkowita modyfikacja kodu, aby dopasować symulowany obiekt do jego odpowiednika w rzeczywistości. Po etapie programowania i testowania sterownika, MRDS wspiera komunikację z urządzeniem za pomocą narzędzi opartych na protokole TCP/IP.

MATLAB® wykorzystano do projektowania i implementacji systemu rozmytego umożliwiającego realizację sterownika robota. Stosuje się w tym celu specjalny przybornik dostępny z pakietem MATLAB® - *Fuzzy Logic Toolbox*™. Interfejs tego przybornika pozwala w sposób intuicyjny zaprojektować i przetestować system rozmyty (np. z blokiem wnioskowania wg Mamdaniego lub Sugeno). Projektowanie polega na odpowiednim użyciu trzech głównych modułów przybornika: *FIS Editor* - za pomocą, którego można określić ilość i nazwy zmiennych lingwistycznych stanowiących odpowiednio wejścia i wyjścia projektowanego systemu oraz sprecyzować metodę i operatory wnioskowania rozmytego; *Membership Function Editor* – który pozwala na dodawanie oraz edycję funkcji przynależności zdefiniowanych zmiennych lingwistycznych; *Rule Editor* - służy do konstruowania zbioru reguł systemu. W celu przetestowania poprawności działania projektowanego systemu, przybornik *Fuzzy Logic Toolbox*™ udostępnia narzędzie do podglądu funkcjonowania utworzonego zbioru reguł. Służy do tego *Rule Viewer*, w którym można edytować wartości wejściowe systemu i obserwować zachowanie się poszczególnych reguł.

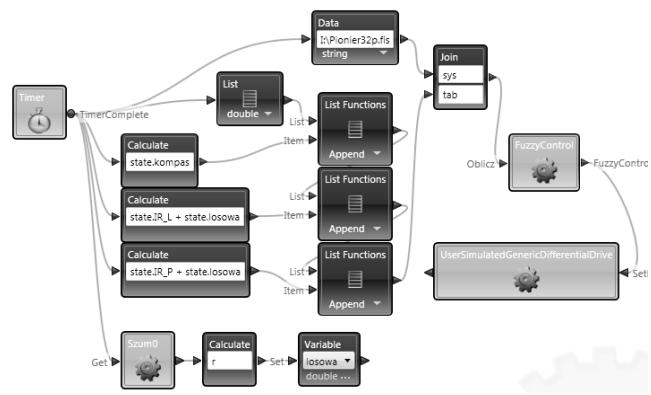
Na rys. 1 przedstawiono w jaki sposób zrealizowana jest wymiana informacji pomiędzy przedstawionymi aplikacjami. Informacje uzyskane za pomocą czujników opisują stan robota i jego lokalnego otoczenia $\mathbf{y}(k)$, czyli np. jego aktualne położenie, orientacja, odległości zmierzone do przeszkód, itp. Dodatkowo do pomiarów z wirtualnych czujników dodawany jest szum $\mathbf{d}(k)$, który zniekształca informacje o stanie otoczenia robota. Informacje te stanowią wartości wejściowe sterownika rozmytego. Drugą zmienną wejściową są parametry sterowania $\mathbf{p}(k)$, będące dodatkowymi informacjami niezbędnymi do pracy sterownika, np. takimi jak punkt docelowy, do którego ma zmierzać robot lub sygnały z manualnego kontrolera operatora sterującego pojazdem. Na podstawie tych informacji sterownik bazując na przyjętym sposobie wnioskowania wyznacza na wyjściu sygnał sterujący $\mathbf{u}(k)$. Wyjście systemu rozmytego stanowi sygnał wejściowy, który używany jest do wysterowania układów napędowych robota. Tym samym w zależności od aktualnego stanu robota i parametrów sterowania, wyznaczany jest sygnał sterujący jego ruchem.

Zgodnie ze schematem przedstawionym na rys. 1, wnioskowanie rozmyte realizowane jest w środowisku uruchomieniowym MATLABa a realizacja ruchu z uwzględnieniem dynamiki robota przeliczana jest w MRDS. Obliczenia związane z pomiarem stanu robota i jego otoczenia oraz dodawaniem szumów (tylko w przypadku symulacji) wykonywane są w MRDS. Następnie za pomocą specjalnie zaprogramowanego serwisu przesyłane są do środowiska uruchomieniowego MATLAB®. W środowisku tym na podstawie przesłanych informacji oraz zgodnie z wykorzystywanym systemem rozmytym, generowane są sygnały wyjściowe systemu. Dane te z powrotem przesyłane są do MRDS, stanowiąc tym samym sygnały sterujące robotem.



Rys. 1. Współdziałanie aplikacji MRDS z środowiskiem uruchomieniowym MATLABa
Fig. 1. Interaction between MRDS and MATLAB Compiler Runtime

Integracja MATLABa z MRDS polega więc na utworzeniu serwisu korzystającego z wcześniej skompilowanej biblioteki funkcji opracowanych za pomocą *Fuzzy Logic Toolbox*™ z zastosowaniem kompilatora skojarzonego z MATLABem. Dostosowując serwis do współdziałania z biblioteką przyjęto następująca jego budowę: informacjami wejściowymi jest wektor z danymi wejściowymi oraz ścieżka do pliku z wykorzystywanym systemem rozmytym, natomiast wyjście serwisu stanowi wektor z uzyskanymi obliczeniami. Przykładowe zastosowanie serwisu w języku VPL przedstawiono na rys. 2.



Rys. 2. Przykładowe zastosowanie serwisu FuzzyControl w VPL
Fig. 2. Exemplary application of the FuzzyControl service in VPL

Przyjęty sposób działania serwisu pozwala utworzyć serwis realizujący dowolny sterownik rozmyty zaprojektowany w MATLABie, niekoniecznie służący do sterowania robotem. Uniwersalność dotycząca ilości wejść i wyjść używanego systemu pozwala na stosowanie go do różnych celów. Po stronie programisty chcącego użyć systemu rozmytego w MRDS konieczne jest tylko ustalenie kolejności elementów na liście z danymi, tak aby była zgodna z wykorzystywanym przez niego systemem rozmytym. Następnie definiując ścieżkę dostępu do tego systemu, może on generować automatycznie sygnały sterujące. Lista z danymi wyjściowymi zmienia swój rozmiar odpowiednio do ilości wyjść systemu, a ich kolejność zgodna jest z kolejnością ustaloną przy projektowaniu systemu.

3. Badania weryfikacyjne

Jednym z etapów prototypowania systemu sterowania są badania weryfikacyjne przygotowywanego rozwiązania. W tym celu można wykorzystać moduł symulacyjny VSE środowiska MRDS. W rozdziale tym przedstawiono wyniki testów zaprojektowanych sterowników wykorzystujących model robota Pioneer 3DX udostępniony przez MRDS. Posiada on niewielkie gabaryty - 39cm, 18cm, 40cm oraz masę - 9 kg. Robot do poruszania się wykorzystuje układ napędowy bazujący na dwóch niezależnie sterowanych silnikach napędowych oraz dodatkowo wyposażony jest w jedno

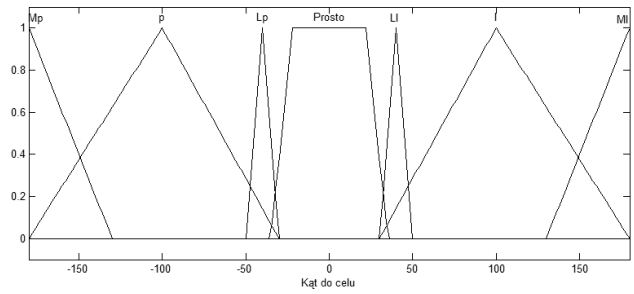
swobodne koło podporowe. W symulacji przyjęto silniki komutatorowe prądu stałego o mocy 12 W. Model robota był odpowiednio modyfikowany pod względem wyposażenia czujników. Dopasowywano go adekwatnie do realizowanej koncepcji sterowania. W tym celu symulowano działanie czujnika skanującego, czujników sonarowych oraz czujników podczerwieni.

Na potrzeby testów utworzono trzy tory przeszkód o różnym stopniu skomplikowania. Położenie i orientacja elementów na torze były uzyskiwane losowo. Tor pierwszy jest torem najmniej skomplikowanym w stosunku do reszty. Posiada elementarne przeszkody w postaci graniastosłupów oraz pachołków. Przestrzenie między przeszkodami są względnie duże i umożliwiają łatwe manewrowanie między nimi. Tor drugi poza przeszkodami występującymi na torze pierwszym, posiada elementy w kształcie litery *L*. Symulują one naroża, na które robot może natrafić podczas poruszania się. Dodatkowo przeszkody są bardziej zagnieżdżone i stanowią utrudnienie przy ich omijaniu. Tor trzeci jest najbardziej skomplikowany ze względu na liczbę i rodzaj przeszkód. Występują na nim dodatkowo elementy w kształcie litery *U* będące zaułkami, które można ominąć tylko poprzez wycofanie się z wnętrza takiego elementu. Przeszkody są bardzo gęsto rozmieszczone stanowiąc czasami całe grupy elementów.

Biorąc pod uwagę działanie symulowanego robota oraz wygenerowane tory przeszkód zaprojektowano trzy sterowniki rozmyte. Różnią się one stopniem zaawansowania oraz wykorzystywanymi układami sensorycznymi. Pierwszy z wariantów sterownika bazuje na informacjach o położeniu względem punktu docelowego oraz informacjach z czujnika skanującego. Czujnik ten jest umieszczony tak, aby skanował teren na wprost i częściowo po bokach robota. Dzięki wysokiej rozdzielczości kątowej czujnika możliwe jest precyzyjne określenie położenia elementu znajdującego się w jego zasięgu. Wykorzystując tą informację skonstruowano system rozmyty bazujący na pomiarze położenia i odległości do najbliższej przeszkody względem robota. Drugi z zaprojektowanych systemów rozmytych również bazuje na pomiarze kąta jaki dzieli robota od punktu docelowego oraz analizie obecności przeszkód w jego otoczeniu. Różnica polega na tym, ze zamiast czujnika skanującego wykorzystuje się trzy czujniki sonarowe (Sonar 1, 2, 3) oraz dwa podczerwieni (IR L, IR P). Czujniki sonarowe swoim zakresem obejmują region przed robotem, dzieląc go na trzy nie nachodzące na siebie strefy. Natomiast czujniki podczerwieni odpowiednio badają przestrzeń po lewej i prawej stronie robota. Trzeci z badanych sterowników bazuje na dokładnie tym samym systemie jak drugi lecz jest zastosowany na robocie wyposażonym w inne czujniki. Czujniki sonarowe zastąpiono czujnikiem skanującym. Modyfikacja polega na tym, że zakres działania czujnika skanującego podzielono na trzy strefy odpowiadające strefom jakie obejmowały czujniki sonarowe (Sektor 1, 2, 3). Następnie pobierana jest wartość średnia zmierzonych odległości w każdej ze stref. Ma to na celu wyznaczenie dokładniejszej informacji o stanie przeszkód wokół robota. Dodatkową funkcjonalnością sterowników rozmytych jest tryb sterowania ręcznego. W trybie tym sterownik płynnie blokuje możliwość kolizji robota z przeszkodami.

Do projektowania poszczególnych wariantów systemów rozmytych w powyższych sterownikach wykorzystano opisany przybornik *Fuzzy Logic Toolbox*™. Za pomocą dostępnych w nim narzędzi utworzono odpowiednie zmienne lingwistyczne i ich funkcje przynależności. Następnie zbudowano bazę reguł realizującą działanie sterownika oraz wyeksportowano kompletne systemy do plików **.fis*. Przykładową zmienną lingwistyczną i jej funkcje przynależności przedstawiono na rys. 3.

Odpowiednio podzielono zakresy wartości zmiennych, wykorzystując wiedzę literaturową [3, 8] oraz własne doświadczenie. W przypadku zmiennej *Kąt do celu*, podzielono jej zakres następująco: przyjęto, że cel może znajdować się na wprost robota (*Prosto*), lekko z lewej (*Ll*), z lewej (*l*), mocno z lewej (*Ml*) oraz analogicznie dla prawej strony (*Lp, p, Mp*). Dodatkowo każdy z sterowników posiada możliwość sterowania ręcznego. Realizują to trzy zmienne: *Sterowanie* – sprawdzając aktualny rodzaj trybu sterowania (ręczne/autonomiczne) oraz *Pad X* i *Pad Y*, analizując kierunek wychylenia drążka analogowego kontrolera sterującego robotem. Komplet zmiennych wejściowych i wyjściowych poszczególnych sterowników rozmytych przedstawiono w tab. 1.



Rys. 3. Funkcje przynależności zmiennej lingwistycznej *Kąt do celu*
Fig. 3. Fuzzy membership functions of the linguistic variable *Kąt do celu*

Tab. 1. Główne cechy sterowników rozmytych
Tab. 1. The main properties of fuzzy controllers

Sterownik 1	**Sterownik 2**	**Sterownik 3**
Liczba reguł		
22	70	70
Wejścia		
$y_1(k)$ – Kąt do celu	$y_1(k)$ – Kąt do celu	$y_1(k)$ – Kąt do celu
$y_2(k)$ – Kąt do przeszkody	$y_2(k)$ – Odległość do celu	$y_2(k)$ – Odległość do celu
$y_3(k)$ – Odległość do przeszkody	$y_3(k)$ – Sonar 1	$y_3(k)$ – Sektor 1
$y_4(k)$ – Odległość do celu	$y_4(k)$ – Sonar 2	$y_4(k)$ – Sektor 2
$p_1(k)$ – Sterowanie	$y_5(k)$ – Sonar 3	$y_5(k)$ – Sektor 3
$p_2(k)$ – Pad X	$y_6(k)$ – IR L	$y_6(k)$ – IR L
$p_3(k)$ – Pad Y	$y_7(k)$ – IR P	$y_7(k)$ – IR P
-	$p_1(k)$ – Sterowanie	$p_1(k)$ – Sterowanie
-	$p_2(k)$ – Pad X	$p_2(k)$ – Pad X
-	$p_3(k)$ – Pad Y	$p_3(k)$ – Pad Y
Wyjścia		
$u_1(k)$ – Napęd L	$u_1(k)$ – Napęd L	$u_1(k)$ – Napęd L
$u_2(k)$ – Napęd P	$u_2(k)$ – Napęd P	$u_2(k)$ – Napęd P

Dla tak przyjętych zmiennych lingwistycznych za pomocą edytora reguł utworzono bazy wiedzy poszczególnych sterowników. Przykładowa reguła z bazy wiedzy wygląda następująco:

$R^{(1)}$ = **JEŻELI** *Kąt do celu* jest *Lp* **I** *Odległość do przeszkody* nie jest *Bardzo blisko* **I** *Odległość do celu* nie jest *Bardzo blisko* **I** *Sterowanie* jest *Off* **TO** *Napęd L* jest *Prawe_obr* **I** *Napęd P* jest *Lewe_obr*.

W sterownikach rozmytych wykorzystano sposób wnioskowania Mamdaniego-Assilana, realizując koniunkcję poprzez wyznaczanie wartości minimum funkcji oraz implikację poprzez maksimum. Wyostrzanie realizowane jest metodą bisector, dzielącą pole pod funkcją wynikową na dwa sektory o równych polach powierzchni. Miejsce ich łączenia wyznacza wyostrzony wynik systemu rozmytego. Dodatkowo w testach wykorzystano sterownik nr 4 bazujący na logice klasycznej. Jest to modyfikacja sterownika udostępnionego przez autorów książki [4].

Weryfikacja działania poszczególnych sterowników polegała na wykonaniu serii zadań postawionych robotowi. Ich celem było dotarcie do punktu docelowego znajdującego się w przeciwległym rogu toru względem pozycji startowej robota. Tym samym robot dla każdego z torów przeszkód wykonywał cztery przejazdy (dwie przekątne, każda oddzielnie w obu kierunkach). Podczas wyko-

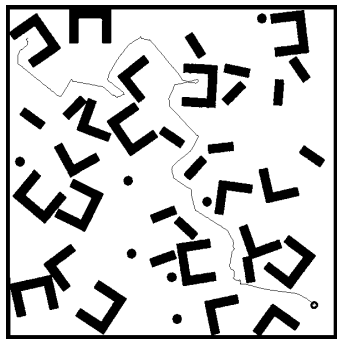
nywania zadania mierzone były wybrane parametry ruchu, po czym poddawane były analizie.

Głównym sterownikiem, na który poświecono najwięcej pracy jest sterownik wg wariantu drugiego. Dzięki temu udało sie zaprojektować system rozmyty sterujący bezkolizyjną pracą robota oraz realizujący wyznaczone zadania we wszystkich testowanych wariantach. W dalszej części artykułu przeprowadzono analizę porównawczą otrzymanych wyników badań. W tab. 2 przedstawiono wartości uśrednione miar zastosowanych do oceny poprawności działania testowanych sterowników.

Tab. 2. Zestawienie porównawcze wyników
Tab. 2. Comparative summary of the results

	Sterownik 1	Sterownik 2	Sterownik 3	Sterownik 4
Tor 1				
Czas [s]	128,5	135,75	134,75	166,75
Droga [m]	26,62	26,58	26,79	31,35
Energia [Wh]	0,33	0,34	0,35	0,53
Skuteczność [%]	100	100	100	100
Tor 2				
Czas [s]	146	170,75	149	205
Droga [m]	28,17	31,5	28,89	42,84
Energia [Wh]	0,38	0,44	0,39	0,69
Skuteczność [%]	50	100	75	100
Tor 3				
Czas [s]	-	260,75	169,25	442
Droga [m]	-	44,66	31,62	85,16
Energia [Wh]	-	0,66	0,44	1,43
Skuteczność [%]	0	100	100	25

Różnice w wynikach dotyczących toru pierwszego w przypadku systemów rozmytych nie są znaczące. Jedynie sterownik bazujący na logice klasycznej niewiele dłużej wykonuje zadanie. Większe rozbieżności można dostrzec w przypadku bardziej złożonych torów przeszkód. Podczas testów na torze drugim sterowniki zrealizowane wg wariantu nr 1 i 3 ujawniły problemy z omijaniem przeszkód. Natomiast sterownik utworzony wg wariantu drugiego uzyskiwał gorsze czasy względem pozostałych sterowników rozmytych, niemniej jednak bezkolizyjnie docierał do celu we wszystkich przypadkach testu. Tor trzeci stanowił zbyt trudne środowisko pracy dla sterownika pierwszego, poprzez to robot wykorzystujący go, ani razu nie dotarł do wyznaczonego celu. Natomiast sterownik czwarty pomimo, iż bezkolizyjnie przemieszczał się wśród przeszkód, to ich układ był zbyt skomplikowany by mógł doprowadzić robota do celu. Tylko w jednym przypadku udało sie to osiągnąć w poniżej 500 sekund. Sterownik wykonany wg wariantu drugiego jako jedyny bezkolizyjnie dla wszystkich testowanych wariantów zdołał dotrzeć do punktu docelowego.



Rys. 4. Przykładowa ścieżka ruchu robota uzyskana za pomocą sterownika wg wariantu nr 2
Fig. 4. Exemplary path of the robot obtained with the use of the second controller

Na rys. 4 przedstawiono przykładową ścieżkę ruchu robota z kontrolerem wg wariantu 2-go na torze nr 3.

4. Wnioski i podsumowanie

W artykule przedstawiono metodykę szybkiego prototypowania sterownika rozmytego do unikania kolizji kołowego robota mobilnego z przeszkodami w środowisku statycznym. Taki sterownik może być użyty do sterowania robotem w sposób autonomiczny lub może zostać zastosowany do wspomagania pracy operatora np. podczas słabej widoczności obrazu z kamery robota. Zaproponowany sposób prototypowania sterownika robota mobilnego z użyciem systemu rozmytego Mamdaniego-Assilana znacznie skraca czas potrzebny na jego realizację. Jest to możliwe dzięki integracji dwóch środowisk programistycznych - Microsoft® Robotics Developer Studio oraz MATLABa®. Znaczącą zaletą tego podejścia jest to, że sterownik może zostać przetestowany już na etapie wstępnego prototypu robota w wersji symulacyjnej. Dodatkowo środowisko, które zostało wykorzystane umożliwia migrację sterownika na robota mobilnego wyposażonego w dowolny system np. poprzez użycie funkcjonalności MATLABa® polegającej na generowaniu kodu w języku ANSI C z użyciem pakietu Real-Time Workshop®. Możliwe jest również wykorzystanie sterownika w sposób bezpośredni na rzeczywistym robocie z zainstalowanym środowiskiem MRDS.

Przedstawione w artykule badania potwierdzają fakt, że proponowane podejście pozwala szybko zaprojektować i zaimplementować sterownik rozmyty, który charakteryzuje się wysoką skutecznością działania. Ponadto, elastyczność i uniwersalność opisanych narzędzi pozwalają na intuicyjne zaadaptowanie sterownika do robota, który wyposażony może być w zestaw czujników o różnym sposobie ich rozmieszczenia lub różnej ich liczbie. Dzięki temu możliwe jest prowadzenie procesu optymalizacji z uwzględnieniem różnych kryteriów, np. podczas projektowania takich elementów systemu sterownia robota jak opisany w tym artykule sterownik ruchu.

5. Literatura

[1] Czupryniak R., Szynkarczyk P., Trojnacki M.: Tendencje rozwoju mobilnych robotów lądowych (2). Przegląd robotów mobilnych do zastosowań specjalnych. Pomiary Automatyka Robotyka, 7-8/2008.
[2] Giergiel M., Żylski W., Hendzel Z.: Modelowanie i sterowanie mobilnych robotów kołowych, PWN, Warszawa, 2002.
[3] Grefe W.: Integrating collision avoidance, lane keeping, and cruise control with an optimal controller and fuzzy controller. Virginia Polytechnic Institute and State University, Virginia, 2005.
[4] Johns K., Taylor T.: Professional Microsoft Robotics Developer Studio, Wiley Publishing, 2008.
[5] Łęski J.: Systemy neuronowo-rozmyte. Wydawnictwo Naukowo-Techniczne, Warszawa, 2008.
[6] Siegwart R., Nourbakhsh I.R., Scaramuzza D.: Introduction To Autonomous Mobile Robots, Second Edition. MIT Press, 2011.
[7] Trojnacki M., Szynkarczyk P., Andrzejuk A.: Tendencje rozwoju mobilnych robotów lądowych (1). Przegląd robotów mobilnych do zastosowań specjalnych. Pomiary Automatyka Robotyka, 6/2008.
[8] Tzafestas S., Zavlangas P., Althoefer K.: Fuzzy obstacle avoidance and navigation for omnidirectional mobile robots. ESIT 2000, Aachen, 2000.
[9] Zadeh L.: Fuzzy sets. Information and Control, (8):338–353, 1965.
[10] The MathWorks: https://www.mathworks.com/help/, 2013.

otrzymano / received: 27.03.2013
przyjęto do druku / accepted: 01.11.2013 *artykuł recenzowany / revised paper*