Piotr ŚWIĄTEK BRZEZIŃSKI, Paweł KOSIŃSKI, **Rafał OSYPIUK**
ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY W SZCZECINIE, KATEDRA AUTOMATYKI PRZEMYSŁOWEJ I ROBOTYKI
ul. 26 Kwietnia 10, 71-126 Szczecin

Integracja czujników inercyjnych z konstrukcją robota humanoidalnego cz. II

Mgr inż. Piotr ŚWIĄTEK-BRZEZIŃSKI

W latach 2005 – 2011 studiował Elektronikę i Telekomunikację na Wydziale Elektrycznym, Zachodniopomorskiego Uniwersytetu Technologicznego w Szczecinie. Jego głównym obszarem zainteresowań jest projektowanie systemów sterowania z wykorzystaniem metod sztucznej inteligencji. Obecnie jest doktorantem w Katedrze Automatyki Przemysłowej i Robotyki.

e-mail: psb@zut.edu.pl

Mgr inż. Paweł KOSIŃSKI

W latach 2005 – 2010 studiował Elektronikę i Telekomunikację na Wydziale Elektrycznym Zachodniopomorskiego Uniwersytetu Technologicznego w Szczecinie. Jego zainteresowania naukowe koncentrują się na projektowaniu i sterowników wbudowanych oraz systemach nawigacji bezwładnościowej. Obecnie jest doktorantem w Katedrze Automatyki Przemysłowej i Robotyki..

e-mail: pkosinski@zut.edu.pl

Streszczenie

W niniejszym artykule przedstawiono algorytmy sterowania robotem humanoidalnym firmy Futaba. Zaprezentowano wykorzystanie sztucznych sieci neuronowych, jako alternatywnego sposobu obliczenia kinematyki odwrotnej oraz wykorzystanie środowiska Microsoft Robotics Developer Studio do tworzenia złożonych, wielowątkowych aplikacji szeroko stosowanych w robotyce. Ponadto pokazano zastosowanie środowiska symulacyjnego VSE (Visual Simulation Environment) w procesie prototypowania algorytmów sterujących.

Słowa kluczowe: Robot humanoidalny, kinematyka odwrotna, Microsoft Robotics Developer Studio.

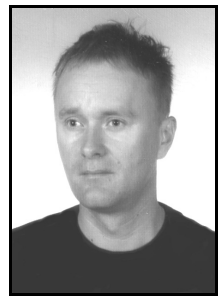
Integration of inertial sensors with humanoid robots body, part II

Abstract

This paper presents a control system for a Futaba humanoid robot [1]. It is equipped with a controller board based on a micro-controller with an ARM7TDMI core, a full set of inertial sensors and a 2.4 GHz wireless communication module. The controller uses the wireless communication module to send information about the robot state to a PC on which the controlling application is run. This paper focuses on the software part of the presented system (the hardware part has been presented in the first part of this paper). In order to develop a controlling algorithm, an analysis of robot kinematics was made and equations for direct kinematics were derived in consistency with the Denavit-Hartenberg convention. To eliminate the necessity of designating equations for inverse kinematics, which can be very complex due to the kinematic redundancy of the robot, an artificial neural network was used (Fig. 2). The application was written using the Microsoft Robotics Developer Studio designed for creating complex, multithread, distributed and scalable applications used in robotics. The application uses the data acquired by radio to implement the walking and balance-keeping algorithms. For visualization of the robot movement, testing and development of the algorithms without the risk of damaging the robot, a simulation in the Visual Simulation Environment, a part of the Microsoft Robotics Developer Studio, was created. A 3D model of the robot was used in this simulation (Fig. 4).

Keywords: humanoid robots, inverse kinematics, Microsoft Robotics Developer Studio.

Dr inż. Rafał OSYPIUK



W latach 1994 - 1999 studiował Automatykę i Robotykę na Politechnice Szczecińskiej. W roku 2004 w tej samej dyscyplinie obronił prace doktorską i został zatrudniony na stanowisku adiunkta w Instytucie Automatyki Przemysłowej. Ściśle współpracuje z Institute for Robotics and Process Control w Braunschweig. Głównym obszarem jego badań są odporne systemy sterowania pozycją oraz siłą dla manipulatorów przemysłowych.

e-mail: rafal.osypiuk@zut.edu.pl

1. Wstęp

Ogromny postęp w dziedzinie mechaniki i elektroniki oraz rozwój badań nad modelowaniem mechanicznych właściwości ludzkiego ciała sprawiły, że konstrukcje robotów humanoidalnych coraz bardziej przypominają swój pierwowzór [2]. Oczekuje się, że w niedalekiej przyszłości roboty tego typy zastąpią ludzi w pracach, które są monotonne lub odbywają się w niebezpiecznych warunkach. W szczególności roboty humanoidalne mogą znaleźć zastosowanie w obszarach takich jak opieka nad osobami w podeszłym wieku, praca w elektrowniach jądrowych czy eksploracja kosmosu. Aby to jednak było możliwe, dalszy rozwój musi nastąpić w dziedzinie konstrukcji i algorytmów stabilnego poruszania się robotów humanoidalnych [3]. Stało się to motywacją dla autorów do budowy platformy sprzętowo-programowej przeznaczonej do opracowywania i testowania algorytmów sterowania dwunożnymi robotami. Platforma ta składa się z robota humanoidalnego Futaba RBT-1 [1] wyposażonego w specjalnie zaprojektowany sterownik mikroprocesorowy oraz zestaw czujników inercyjnych do nawigacji bezwładnościowej. Projekt oraz realizacja sprzętowa została opisana w pierwszej części artykułu [4] i stanowiła bazę do dalszych prac, związanych z implementacją algorytmów sterowania ruchem oraz modelowaniem i wizualizacją procesu.

2. Kinematyka prosta

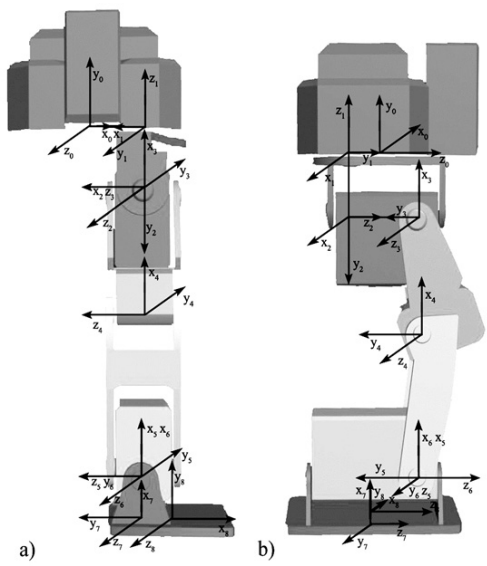
Noga robota humanoidalnego Futaba RBT-1 stanowi otwarty łańcuch kinematyczny o sześciu sterowanych stopniach swobody. Znalezienie pozycji oraz orientacji stopy robota, znajdującej się na końcu tego łańcucha, przy znajomości kątów zgięć każdego z przegubów, nazywa się zadaniem prostym kinematyki [5]. Równania kinematyki prostej dla rozważanej konstrukcji zawsze mają jedno, ściśle określone rozwiązanie tj. każdemu wektorowi reprezentującemu kąty zgięć przegubów przyporządkowana jest dokładnie jedna pozycja i orientacja stopy.

Przy wyznaczaniu kinematyki prostej posłużono się notacją Denavita-Hartenberga [5]. Sposób przyporządkowania układów współrzędnych do poszczególnych członów robota przedstawiony jest na rys. 1. Orientacja układu bazowego O_0 oraz układu skojarzonego ze stopą robota O_8 zostały przyjęte w taki sposób, aby zachować zgodność z orientacją układów współrzędnych w środowisku symulacyjnym VSE Microsoft Robotics Development Studio.

3. Kinematyka odwrotna

Znajomość równań kinematyki prostej pozwala na określenie położenia i orientacji członów robota na podstawie wartości jego zmiennych przegubowych. Jednak z punktu widzenia algorytmu

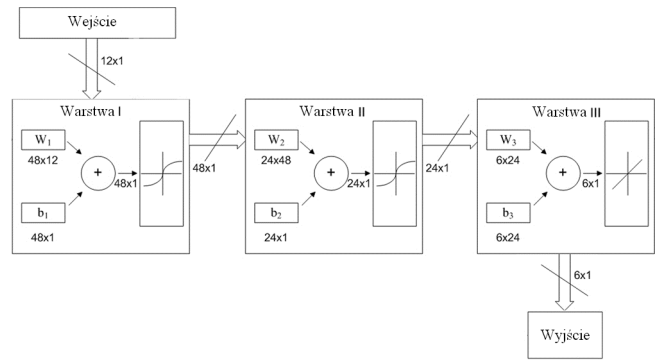
sterowania i planowania trajektorii ruchu, dużo ważniejsza jest znajomość przekształcenia odwrotnego tj. umiejętność określenia, jakie wartości zmiennych przegubowych odpowiadają pożądanej pozycji i orientacji końcówki roboczej. Jest to tak zwane zadanie odwrotne kinematyki [5] i w ogólności jest ono dużo bardziej skomplikowane niż zadanie proste. O ile równania kinematyki prostej mają zawsze jedno, ściśle określone rozwiązanie, tak równania kinematyki odwrotnej mogą mieć jedno, wiele lub w ogóle nie posiadać rozwiązania. W dodatku odwrócenie równań kinematyki prostej, które są zazwyczaj skomplikowanymi, nieliniowymi funkcjami zmiennych przegubowych niejednokrotnie jest zadaniem bardzo trudnym. W przypadku manipulatorów przemysłowych zadanie to upraszcza się poprzez specjalną konstrukcję mechaniczną, pozwalającą na rozbicie zadania odwrotnego kinematyki na dwa prostsze zadania – zadanie odwrotne pozycji oraz zadanie odwrotne orientacji. Niestety podejścia tego nie można zastosować w odniesieni do prezentowanej konstrukcji. Z tego powodu skorzystano z metod sztucznych sieci neuronowych [6], które z powodzeniem były testowana na robocie humanoidalnym PINO [7].



Rys. 1. Przyporządkowanie układów współrzędnych do nogi robota: a) widok od przodu, b) widok z prawej strony
Fig. 1. Assignment of coordinate systems to the robot legs: a) front view, b) right side view

Sztuczne sieci neuronowe (ang. *Artificial Neural Network*, w skrócie ANN), stanowią dynamicznie rozwijającą się dziedzinę wiedzy. Ich powstanie było możliwe dzięki rozwojowi badań nad działaniem biologicznych struktur nerwowych, wśród których niedoścignionym wzorem jest ludzki mózg. Dzięki swym właściwością, do których zaliczyć można: zdolność do generalizacji nabytej wiedzy, umiejętność uczenia się i adaptacji, odporność na uszkodzenia, umiejętność generowania prawidłowego wyniku mimo zaszumionych lub częściowo brakujących danych wejściowych, ANN znalazły bardzo szerokie zastosowanie [8, 9].

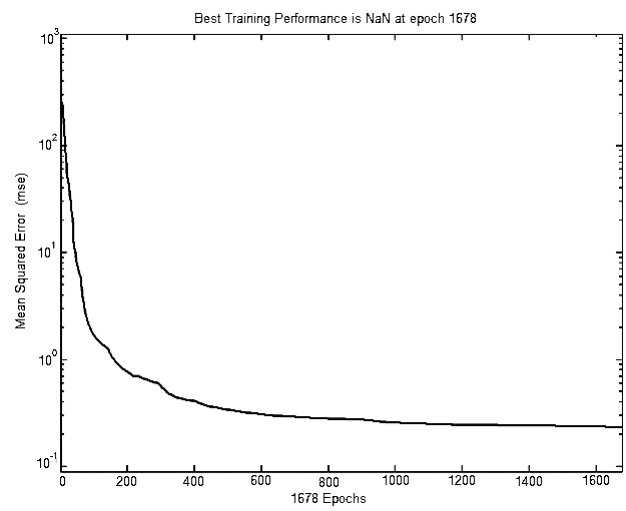
W pracy wykorzystano sztuczną sieć neuronową do obliczenia zadania odwrotnego kinematyki nóg robota. Algorytm zaimplementowano w programie Matlab/Simulink. Architekturę sieci dobrano eksperymentalnie i po wielu próbach najlepsze efekty uzyskano dla trójwarstwowej sieci typu „*feedforward*" (rys. 2) ze wsteczną propagacją błędu. Ilość neuronów w warstwach ukrytych to odpowiednio 48 i 24. Warstwa wyjściowa liczy 6 neuronów. Dla warstw ukrytych, jako funkcję przejścia zastosowano tangens hiperboliczny, natomiast dla warstwy wyjściowej funkcję liniową, ponieważ dane wyjściowe sieci przybierają wartości z poza zakresu (-1, 1). Schemat struktury sieci przedstawiono na rys. 2. Dane wejściowe sieci stanowi wektor 12x1 złożony z 9 elementów macierzy orientacji stopy robota oraz współrzędnych *x, y, z* określających jej położenie. Wynikiem działania sieci jest wektor 6x1 zawierający wartości zmiennych przegubowych odpowiadających żądanej orientacji i położeniu stopy.



Rys. 2. Schemat struktury sieci neuronowej
Fig. 2. Schematic structure of a neural network

Dane uczące sieć zostały wygnerowane przy pomocy równań kinematyki prostej. Składają się one z 4096 wektorów wejściowych zawierających dane o pozycji i orientacji stopy robota oraz odpowiadającym im wektorów zawierających pożądane wartości zmiennych przegubowych. Podczas generacji danych uczących wzięto pod uwagę ograniczenia konstrukcyjne robota (wartości minimalne i maksymalne kątów dla każdego z przegubów). W celu ograniczenia ilości danych uczących, a co za tym idzie wielkości sieci oraz czasu nauki, ograniczono zakresy zmienności kątów przegubów. Jest to możliwe, ponieważ podczas chodu lub utrzymywania równowagi przez robota wykorzystywany jest tylko fragment przestrzeni osiągalnej przez jego stopę. Zastosowanie takiego podejścia powoduje, iż sieć dużo lepiej uczy się realizować zadanie odwrotne kinematyki wewnątrz pożądanego fragmentu obszaru roboczego, niż miałoby to miejsce podczas nauki realizacji tego zadania dla pełnego zakresu kątów.

Po przetestowaniu różnych algorytmów uczących [10], między innymi metody Levenberga-Marquardta, metody quasi-Newtonowskiej czy metody gradientu prostego, wybrany został algorytm LM. Algorytm ten okazał się najszybszy, gdzie już po 150 iteracjach błąd średniokwadratowy sieci wynosił mniej niż 1 stopień. Proces uczenia się sieci został przedstawiony na rys. 3. Wykres prezentuje zmianę wartości błędu średniokwadratowego pomiędzy wyjściem sieci, a wartościami pożądanymi w kolejnych epokach uczenia.



Rys. 3. Postęp uczenia sieci neuronowej
Fig. 3. Progress of the neural network learning

Dla prezentowanej sieci uzyskano błąd średnio kwadratowy na poziomie 0.2 stopnia. Sieć wykazała także poprawne działanie w przypadku danych nieprezentowanych jej wcześniej. W tym przypadku średni błąd wyznaczenia każdego z 6 kątów nie był większy niż 1 stopień, co przekłada się na niedokładność pozycji stopy robota na poziomie ok. 1mm dla każdej z osi.

4. Microsoft Robotics Developer Studio

Rosnący stopień złożoności systemów zrobotyzowanych wymusza na programistach poszukiwanie nowych, wydajnych metod prototypowania. W tym celu firma Microsoft udostępniła środowisko programistyczne Microsoft Robotics Developer Studio (MRDS) [11]. Jest to darmowe, oparte o platformę .NET, środowisko umożliwiające projektowanie, implementację oraz debugowanie wysoce skalowalnych, wielowątkowych, rozproszonych aplikacji przeznaczonych (ale nie ograniczających się) do zadań związanych z szeroko pojętą robotyką. MRDS ułatwia rozwiązanie problemów przed jakimi stają programiści piszący złożone aplikacje jak np.: koordynacja wielu wątków, reakcja na asynchronicznie pojawiające się sygnały, obsługa błędów czy przenośność programów na różne platformy sprzętowe. Oprócz środowiska uruchomieniowego oraz wielu innych narzędzi, platforma deweloperska MRDS oferuje także duży zbiór przykładów, instrukcji oraz obszerną dokumentację.

Środowisko uruchomieniowe (ang. runtime) MRDS korzysta z trzech środowisk niższego rzędu: Concurrency and Coordination Runtime (CCR), Decentralized Software Services (DSS) oraz Common Language Runtime (CLR). CCR to, oparty na wiadomościach model programistyczny, pozwalający aplikacji koordynować asynchroniczne procesy oraz wykorzystywać współbieżność w bardzo efektywny sposób. DSS pozwala na budowanie aplikacji składających się z luźno połączonych usług. Usługi mogą działać na różnych węzłach sieci, dlatego DSS dostarcza infrastrukturę komunikacyjną umożliwiającą działać im tak, jakby były wykonywane lokalnie. CLR wspiera CCR oraz DSS, oraz umożliwia dostęp do platformy .NET [12, 13].

4.1. Aplikacja sterująca

Aplikacja sterująca robotem została napisana w języku C# w środowisku Microsoft Robotics Developer Studio przy pomocy zintegrowanego środowiska programistycznego Microsoft Visual Studio 2008. Aplikacji postawiono trzy podstawowe cele:

- Sterowanie robotem Futaba RBT-1 w czasie rzeczywistym przy wykorzystaniu łączności bezprzewodowej.
- Uruchomienie i kontrolę symulacji w środowisku symulacyjnym MRDS VSE.
- Zapewnienie interfejsu użytkownika.

Aplikacja potrafi komunikować się z robotem poprzez wirtualny port COM oraz specjalnie zaprojektowany moduł komunikacji radiowej 2,4GHz wpięty do portu USB komputera. Na podstawie otrzymanych danych o stanie serwonapędów oraz wychyleniu, obliczane są dane sterujące wysyłane do robota. Aplikacja działa w dwóch trybach:

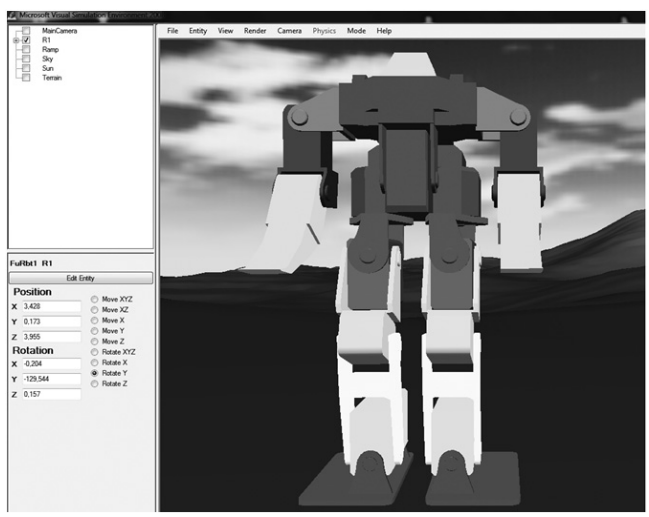
- Tryb symulacji – nie ma połączenia radiowego z rzeczywistym robotem i jako obiekt sterowania używany jest model robota wewnątrz symulacji.
- Tryb sterowania – aplikacja komunikuje się z rzeczywistym robotem tzn. odbiera dane o stanie robota i wysyła dane sterujące realizując zamkniętą pętle sterowania. W tym trybie symulacja służy jako wizualizacja stanu faktycznego robota.

Dzięki takiemu podejściu możliwe jest opracowanie i testowanie algorytmów sterowania robotem przy wykorzystaniu jego wirtualnego modelu, a następnie łatwe ich przeniesienie na rzeczywisty obiekt. Możliwość ta jest bardzo cenna z wielu względów. Po pierwsze znacznie przyspiesza pracę, ponieważ nie wymaga dostępu do fizycznego urządzenia. Po drugie minimalizuje ryzyko uszkodzenia robota w wyniku źle działającego algorytmu, gdyż błędy można wykryć i eliminować już na poziomie symulacji. Dodatkową zaletą takiego rozwiązania jest łatwość debugowania programu.

4.2. Symulacja

Integralną częścią pakietu MRDS jest środowisko symulacyjne VSE (Visual Simulation Environment). Za jego pomocą możliwe jest zbudowanie trójwymiarowego, wirtualnego świata oraz umieszczenie w nim symulowanych obiektów. W celu symulacji interakcji między tymi obiektami VSE używa silnika fizyki AGEIA PhysiX. Dużą zaleta tego rozwiązania jest możliwość sprzętowej akceleracji obliczeń przez zastosowanie specjalnej karty PhysiX PPU lub karty graficznej wyposażonej w technologię CUDA. Ponadto VSE używa środowiska uruchomieniowego Microsoft DirectX oraz bibliotek Microsoft XNA [12].

Podstawowymi elementami symulacji są jednostki (ang. entity), które reprezentują obiekty znajdujące się w wirtualnym świecie. Aby symulacja jak najlepiej odzwierciedlała rzeczywistość, dla każdej jednostki można zdefiniować szereg parametrów takich jak: wymiary, masa, gęstość, rozkład masy, współczynnik tarcia statycznego i dynamicznego, prędkość, położenie, orientacja itd.. Przez ustawienie odpowiednich flag można również sprawić, aby jednostka mogła się poruszać tylko w określonych osiach, wyłączyć wpływ grawitacji na jednostkę lub sprawić, że będzie ona ignorowana przez silnik fizyczny. Pozwala to tworzyć bardzo skomplikowane i realistyczne symulacje, których złożoność ogranicza jedynie moc obliczeniowa użytego komputera.



Rys. 4. Widok okna symulacji VSE
Fig. 4. View of the VSE simulation window

W celu zasymulowania robota Futaba RBT-1 w środowisku VSE, sporządzono jego trójwymiarowy model w programie 3DS MAX. Model składa się z 21 części. Każda z części została zapisana w postaci osobnego pliku OBJ, który jest otwartym formatem plików używanym w grafice 3D. VSE umożliwia importowanie obiektów zapisanych w tym formacie. Elementy robota połączone są ze sobą przy pomocy złącz (ang. joints) o jednym stopniu swobody, których właściwości statyczne i dynamiczne można dowolnie zmieniać dostosowując je do parametrów serwonapędów użytych w prawdziwym robocie. Okno symulacji przedstawione zostało na rys. 4.

5. Wnioski

W artykule przedstawiono sposób rozwiązania zadania odwrotnego kinematyki w przypadku robota humanoidalnego poprzez zastosowanie sztucznych sieci neuronowych. Zastosowanie SSN sprawia, że analityczne wyznaczanie równań kinematyki odwrotnej, co z uwagi na złożoność budowy robotów humanoidalnych

jest bardzo trudne, nie jest konieczne. Dzięki zaproponowanej architekturze sieci oraz sposobie jej uczenia, uzyskano zadowalające wyniki ze średnią niedokładnością pozycjonowania stopy robota na poziomie 1mm dla każdej z osi wewnątrz obszaru roboczego. Zaproponowana sieć była zdolna do generalizacji nabytej wiedzy dla danych wejściowych, które nie były jej wcześniej prezentowane, co ma decydujące znaczenie przy praktycznym wykorzystaniu tej metody.

W artykule zaprezentowano również wykorzystanie współczesnych narzędzi programistycznych w postaci środowiska Microsoft Robotics Developer Studio do budowy rozproszonego, skalowalnego systemu sterowania, wizualizacji i symulacji robota humanoidalnego w czasie rzeczywistym.

Przedstawione w artykule zagadnienia stanowią elementy pracy dyplomowej [14], która zajęła I miejsce w ogólnopolskim konkursie o nagrodę Siemensa dla absolwentów w 2011 roku. Krótki film prezentujący główne osiągnięcia pracy został opublikowany w serwisie YouTube [15].

6. Literatura

[1] http://www.robotshop.com/PDF/RBT-1_Operation_Manual.pdf, instrukcja obsługi robota Futaba RBT-1, 2007.

[2] Zaier R.: The Future of Humanoid Robots - Research and Applications, InTech, DOI 10.5772/1407, 2012.

[3] Harada K., Yoshida E., Yokoi K.: Motion Planning for Humanoid Robots, Springer, ISBN: 1849962197, 2010.

[4] Kosiński P., Świątek-Brzeziński P., Osypiuk R.: Integracja czujników inercyjnych z konstrukcją robota humanoidalnego cz. I, Pomiary Automatyka Kontrola, PAK, 2012.

[5] Spong M. W., Vidyasagar M.: Dynamika i sterowanie robotów, Wydawnictwa Naukowo-Techniczne, 1997.

[6] Lope J., Zarraonandia T., Gonzalez-Careaga R., Maravall D.: Solving the Inverse Kinematics in Humanoid Robots: A Neural Approach, Department of Articial Intelligence, Universidad Politecnica de Madrid, 2009.

[7] Severin E. O.: Robot Companions: MentorBots and Beyond, McGraw-Hill/TAB Electronics, 2003.

[8] Kwaśnicka H.: Ewolucyjne projektowanie sieci neuronowych.: Oficyna Wydawnicza Politechniki Wrocławskiej, 2007.

[9] Osowski S.: Sieci neuronowe w ujęciu algorytmicznym.: Wydawnictwa Naukowo-Techniczne, 1997.

[10] Kelly C.T.: Iterative Methods for Optimization. SIAM Press, Philadelphia, 1999.

[11] Microsoft. Witryna firmy Microsoft. http://msdn.microsoft.com

[12] Johns K., Taylor T.: Professional Microsoft Robotic Developer Studio.: Wiley Publishing, 2008.

[13] Morgan S.: Programming Microsoft Robotic Studio, Microsoft Press, 2008.

[14] Świątek Brzeziński P., Kosiński P.: Integracja zaawansowanych sensorów z konstrukcją robota humanoidalnego, ZUT w Szczecinie, praca dyplomowa, 2011.

[15] http://www.youtube.com/watch?v=MUFJVlG_CIM

otrzymano / received: 23.03.2012
przyjęto do druku / accepted: 03.12.2012 *artykuł recenzowany / revised paper*

INFORMACJE

Informacje dla Autorów

Redakcja przyjmuje do publikacji tylko prace oryginalne, nie publikowane wcześniej w innych czasopismach. Redakcja nie zwraca materiałów nie zamówionych oraz zastrzega sobie prawo redagowania i skracania tekstów oraz streszczeń.

Artykuły naukowe publikowane w czasopiśmie PAK są formatowane jednolicie zgodnie z ustaloną formatką zamieszczoną na stronie redakcyjnej www.pak.info.pl. Dlatego artykuły przekazywane redakcji należy przygotowywać w edytorze Microsoft Word 2003 (w formacie DOC) z zachowaniem:

- wielkości czcionek,
- odstępów między wierszami tekstu,
- odstępów przed i po rysunkach, wzorach i tabelach,
- oznaczeń we wzorach, tabelach i na rysunkach zgodnych z oznaczeniami w tekście,
- układu poszczególnych elementów na stronie.

Osobno należy przygotować w pliku w formacie DOC notki biograficzne autorów o objętości nie przekraczającej 450 znaków, zawierające podstawowe dane charakteryzujące działalność naukową, tytuły naukowe i zawodowe, miejsce pracy i zajmowane stanowiska, informacje o uprawianej dziedzinie, adres e-mail oraz aktualne zdjęcie autora o rozmiarze 3,8 x 2,7 cm zapisane w skali odcieni szarości lub dołączone w osobnym pliku (w formacie TIF).

Wszystkie materiały:

- artykuł (w formacie DOC),
- notki biograficzne autorów (w formacie DOC),
- zdjęcia i rysunki (w formacie TIF lub CDR),

prosimy przesyłać w formie plików oraz dodatkowo jako wydruki na białym papierze (lub w formacie PDF) na adres e-mail: wydawnictwo@pak.info.pl lub pocztą zwykłą, na adres: Redakcja Czasopisma Pomiary Automatyka Kontrola, Asystent Redaktora Naczelnego mgr Agnieszka Skórkowska, ul. Akademicka 10, p.21A, 44-100 Gliwice.

Wszystkie artykuły naukowe są dopuszczane do publikacji w czasopiśmie PAK po otrzymaniu pozytywnej recenzji. Autorzy materiałów nadesłanych do publikacji są odpowiedzialni za przestrzeganie prawa autorskiego. Zarówno treść pracy, jak i wykorzystane w niej ilustracje oraz tabele powinny stanowić dorobek własny Autora lub muszą być opisane zgodnie z zasadami cytowania, z powołaniem się na źródło cytatu.

Przedrukowywanie materiałów lub ich fragmentów wymaga pisemnej zgody redakcji. Redakcja ma prawo do korzystania z utworu, rozporządzania nim i udostępniania dowolną techniką, w tym też elektroniczną oraz ma prawo do rozpowszechniania go dowolnymi kanałami dystrybucyjnymi.