Teodora DIMITROVA – GREKOW
POLITECHNIKA BIAŁOSTOCKA, WYDZIAŁ INFORMATYKI,
ul. Wiejska 45 A, 15-351 Białystok

Quasi-autonomiczny pojazd na bazie zestawu Lego Mindstorms NXT

Dr inż. Teodora DIMITROVA - GREKOW

Ukończyła studia na Wydziale Elektroniki Politechniki Sofijskiej, Bułgaria w 1991r., obroniła pracę doktorską na Uniwersytecie Technicznym w Wiedniu w 1997r. Jest adiunktem na Wydziale Informatyki przy Politechnice Białostockiej. Jej zainteresowania naukowe to synteza układów programowalnych, robotyka, mechatronika, analiza i przetwarzanie sygnałów.

e-mail: t.grekow@pb.edu.pl

Streszczenie

W artykule przedstawiono metodę sterowania pojazdem, która jest analogiczna do wydawania poleceń człowiekowi, czyli operuje na abstrakcyjnych pojęciach. Zbadane zostało na ile możliwa jest budowa takiego systemu korzystając z powszechnie dostępnego sprzętu - zestawu Lego Mindstorms NXT. Główne zadania wykonywane przez robota, to budowanie mapy, osiągnięcie pozycji wskazanej przez operatora oraz wykrywanie i rozpoznawanie prostych obiektów. Badania eksperymentalne wykazały, że zastosowany algorytm rozpoznawaniu obiektów, uzyskał wysoką trafność.

Słowa kluczowe: pojazdy quasi-autonomiczne, nawigacja, mapowanie otoczenia, komunikacja człowiek-maszyna, detekcja obiektów, algorytmy SLAM, Mindstorms NXT.

Lego Mindstorms NXT based quasi-autonomous vehicle

Abstract

The presented research evaluates how suitable the Lego Mindstorms NXT sets are for implementing real time algorithms in mobile devices. Therefore, a vehicle control method, based on an abstract commands list, was developed. The main tasks executed by the vehicle are: mapping, reaching a location determined by the operator and recognizing a primitive object. The interface of the project is shown in Fig. 1. It allows the user/operator to observe the map, the parameters of the objects and to send commands from the dynamic generated command list. The implemented mapping and object detection algorithms are shown in Fig. 2. The vehicle construction (Fig. 3) is based on the classical model of 2 steered and one balancing wheel. Tab.1 presents the following test results: the achieved position accuracy, object recognition precision and the time of task completion. The odometry [2, 3] of the construction is not good enough. Navigation could be improved by adding some more precise sensors [4], [5]. The efficiency of the object detection algorithm reaches 75%, which is relatively suitable for the vehicle tasks. The test experience shows that the Lego NXT set is not an easy platform to create a general purpose vehicle, because of the many sensor and motor problems. However, the general positive results speak for its partial suitability by experimental verification of different modern attempts requiring vehicles, avoiding great hardware costs.

Keywords: quasi-automonous vehicle, mapping, object detection, navigation, SLAM algorithms, Mindstorms NXT, human machine communication.

1. Wstęp

Na współczesnym rynku szeroko dostępnych automatów dominują dwa rodzaje urządzeń: urządzenia wymagające bezpośredniej kontroli przez człowieka oraz urządzenia niemal w pełni automatyczne. Do pierwszych należą wszelkie zdalnie sterowane zabawki oraz większość pojazdów mechanicznych, a także roboty chirurgiczne [1]. Do prawidłowej pracy wymagają one ciągłej uwagi i nadzoru człowieka. Niewątpliwą zaletą takiego podejścia jest gwarancja, że urządzenie będzie działać w każdych warunkach (do których zostało zaprojektowane). Wadą kontroli bezpośredniej jest jednak angażowanie człowieka. Z założenia roboty istnieją po to, aby przejąć zadania, których człowiek nie chce się podejmować, ze względów ekonomicznych (jednostka czasu pracy człowieka kosztuje zwykle więcej niż automatu) lub ergonomicznych (ang. *Dull, Dirty and Dangerous*). Przykładem może być kierowanie pojazdem, które wymaga ciągłego, wielogodzinnego skupienia na monotonnym zadaniu. Roboty wykonane w ten sposób mogą być postrzegane jako urządzenia rozszerzające zdolności człowieka. Interakcja z nimi następuje w taki sam sposób, jak z ciałem właściciela lub narzędziami. Każda komenda wysłana do urządzenia zostaje wykonana natychmiast, oraz aby wykonać zadanie należy poświęcać mu ciągłą uwagę.

Przeciwnym podejściem jest wcześniejsze zaprogramowanie robota tak, aby wykonywał swoje działania automatycznie. Stosuje się je w robotycznych zabawkach (np. Sony AIBO), odkurzaczach (iRobot Roomba), lub w robotach przemysłowych. Roboty takie muszą być zaprogramowane przed oddaniem ich do użytku, a następnie działają bez nadzoru człowieka. Jest to ich największa zaleta, jednak niesie ze sobą pewne ograniczenia. Aby zaprogramować takie urządzenie do prawidłowej pracy we wszystkich okolicznościach, należy wziąć te niedostatki pod uwagę. Z tego powodu najczęściej są one wykorzystywane w kontrolowanych środowiskach.

Niniejsza praca pokazuje podejście pośrednie: urządzenia, działają nie całkiem automatycznie ani całkiem pod kontrolą. Zwykle automatyzowane są proste czynności. Ponieważ istnieje niewiele pojazdów tego typu, powstał pomysł podjęcia próby wypełnienia tej luki. Przedstawiono projekt systemu: pojazd którego sterowanie wysokiego poziomu zakłada wbudowana autonomiczność. Pokazano niektóre techniki użyte do realizacji części programistycznej oraz przeprowadzone testy głównych funkcjonalności. Istotnym punktem badań jest to na ile można liczyć na zestawy edukacyjne Lego NXT przy weryfikacji skuteczności podobnych projektów.

2. Sterowanie pojazdem

Opracowany został system sterowania pojazdem, w podobny sposób, w jaki wydaje się polecenia człowiekowi - operując na abstrakcyjnych pojęciach. System ten został zaimplementowany z użyciem realnego robota edukacyjnego Lego Mindstorms NXT. Celem tej implementacji jest przede wszystkim zbadanie w jakim stopniu nadają się te zestawy do weryfikacji algorytmów nawigacyjnych oraz rozpoznawania obiektów. Zaplanowany w celach badań scenariusz zakłada że:

- robot i operator są w oddzielnych pomieszczeniach;
- robot buduje mapę otoczenia (zadanie domyślne);
- robot identyfikuje obiekty (rozpoznaje wybrane kształty geometryczne);
- robot przesyła dane operatorowi (przede wszystkim aktualizuje mapę otoczenia);
- robot wykonuje polecenia operatora.

Założenie, że eksperymentalna część pracy będzie oparta na zestawie Lego Mindstorms NXT, wymaga zastosowania pewnej umowy. Aby dostosować złożoność oprogramowania do możliwości sprzętowych, przyjęte zostały dwa upraszczające założenia dotyczące otoczenia, w którym robot będzie się poruszać. Otoczenie powinno być:

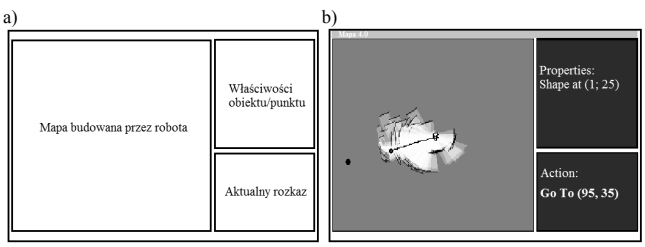
- dwuwymiarowe 2D - ze względu na skalę pojazdu i zakres zastosowań, można przyjąć, iż robot zawsze będzie się poruszał w jednej płaszczyźnie, a ewentualne nierówności podłoża nie wpłyną znacząco na jego postrzeganie otoczenia;
- statyczne - przy dostępnym zestawie sensorów i aktorów, każda zmiana w otoczeniu miałaby za duży wpływ na rozpoznanie te-

renu przez robota. Można zakładać, że praca w tym kierunku będzie podjęta w dalszych badaniach, chociaż wyniki niniejszej pracy nie są optymistycznie w tej kwestii.

W celu realizacji założeń projektu, został napisany program składający się z kilku równoległych wątków:

- tworzenie mapy otoczenia z pomocą danych odczytanych z sensorów (częstotliwość 3Hz);
- detekcja obiektów (co 30s);
- komunikacja z użytkownikiem: przekazywanie bieżących danych o otoczeniu oraz wykonywanie poleceń użytkownika.

Graficzny interfejs programu (rys. 1) pozwala użytkownikowi na obserwację budowanej przez robota mapy (zadanie wykonywane domyślnie), śledzenie parametrów obiektów (robot, namierzone przedmioty itd.) oraz wydawanie rozkazów robotowi. Na rys. 1 jest pokazany wygląd ogólny interfejsu (rys. 1a) oraz zrzut ekranu podczas wykonania przykładowego zadania (rys. 1b). Punkt docelowy dla robota, wybrany przez operatora zaznaczono czarnym kółkiem. Do momentu zaznaczenia akcji robot samodzielnie tworzy mapę - kolor biały wskazuje na zmapowaną wolną przestrzeń.



Rys. 1. Graficzny interfejs użytkownika: a) ogólny wygląd; b) wykonanie rozkazu GoTo (x, y)
Fig. 1. GUI a) general view; b) execution of a task GoTo(x,y)

Zaimplementowane zostały podstawowe polecenia własnego protokołu komunikacji, wyświetlające mapę oraz pozycję robota. Komunikacja odbywała się jednokierunkowo za pomocą kolejki FIFO, a najważniejsze polecenia, które użytkownik wydaje robotowi to dojechanie do punktu [x,y] oraz zidentyfikowanie obiektu.

Wybór zastosowanego algorytmu został podyktowany dwoma głównymi kryteriami:

- redukcja błędu linearnego;
- minimalizacja obciążenia obliczeniowego.

Po zrobieniu przeglądu istniejących podejść, najbardziej obiecujące rezultaty wykazało użycie równoczesnej lokalizacji i mapowania, czyli algorytm SLAM [4]. Ogólnie mówiąc algorytm ten porównuje informacje docierające z otoczenia z dotychczasową wiedzą na jego temat. Przyjmując za d_{i+1} dane na temat otoczenia pobrane z pozycji p_{i+1}, a za m_i mapę otoczenia, próbuje znaleźć takie p_{i+1}, aby informacje d_{i+1} nie przeczyły wiedzy zawartej w m_i. Po znalezieniu tej pozycji, integracja d_{i+1} z m_i daje kolejną mapę m_{i+1}.

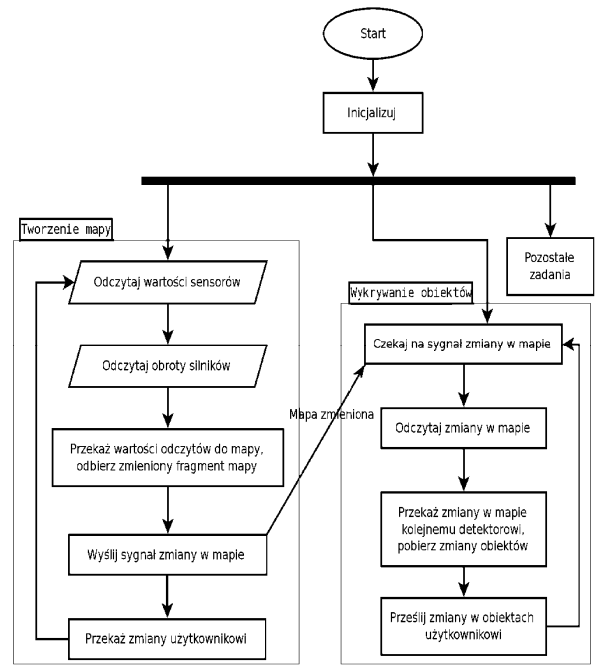
Ponieważ w świecie rzeczywistym zawsze istnieje niepewność pomiarów, nie można się spodziewać, że da się znaleźć pozycję idealnie odpowiadającą poprzedniej wiedzy. Z tego powodu w każdym zaawansowanym algorytmie SLAM można zaobserwować pewne elementy analizy statystycznej.

Użyty algorytm SLAM został zaprojektowany w taki sposób, aby akceptował wszelkie sensory odległości. Zapewnia to projektowi uniwersalność i otwiera szerokie możliwości przyszłym eksperymentom. Testy robota przeprowadzone zostały z wykorzystaniem sensora ultradźwiękowego.

Zaimplementowane algorytmy tworzenia mapy oraz wykrywania obiektów są pokazane na rys. 2.

Z punktu widzenia programisty, całościowy projekt składa się z dwóch wątków: wyświetlanie elementów interfejsu graficznego oraz odbieranie danych od użytkownika. W ten sposób możliwe jest zaimplementowanie natychmiastowych reakcji interfejsu na dane przesyłane z robota.

Ze względu na ograniczenia używanej biblioteki graficznej (GTK), dane odbierane przez drugi wątek nie mogą być wyświetlane przez niego na ekranie. Stąd, wątek dynamicznie generuje funkcje wywołujące odpowiednie procedury (rys. 2), a następnie przekazuje te funkcje bibliotece GTK.



Rys. 2. Schemat działania wątku tworzącego mapy oraz wykrywającego obiektów
Fig. 2. Algorithms of mapping and object identification

Lista akcji, które operator (użytkownik) może wykonać składa się z dwóch zasadniczo różniących się grup: akcje operujące na obiektach oraz akcje operujące na punktach.

Druga grupa akcji przeznaczona jest do zadań skierowania robota do konkretnego punktu mapy.

Lista zadań jest generowana automatycznie (w chwili wybrania obiektu) na podstawie danych przesłanych z robota, stąd może być różna dla każdego obiektu.

3. Doświadczenia eksperymentalne

Opracowane algorytmy i interfejs użytkownika, wsparte zapleczem komunikacyjno–sterującym zostały zaimplementowane oraz praktycznie przetestowane na robocie Lego Mindstorms NXT.

Eksperymentalna konstrukcja robota (rys. 3) została zbudowana w oparciu o podstawowy model ruchu: robot posiada dwa koła napędowe, sterowane niezależnie oraz koło stabilizujące. Atutem takiej konstrukcji jest możliwość obrotu w miejscu.

Główne założenia przeprowadzanych testów to:

- a priori wiadome są wielkość i kształt pomieszczenia do którego jest wpuszczany robot;
- przedmioty w pomieszczeniu są prostopadłościanami o różnych rozmiarach, przy czym długość najmniejszej ściany wynosi 10cm, a największej 50cm.

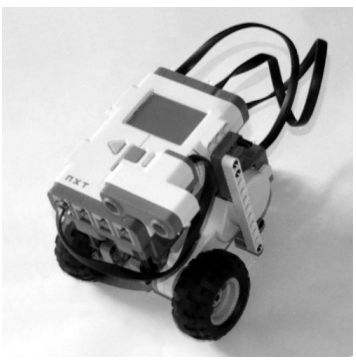
Zadania robota to:

- robot ma sam ustalić swoją początkową pozycję i dalej weryfikować mapę;
- robot ma dotrzeć do punktu o współrzędnych wskazanych przez użytkownika za pomocą interfejsu graficznego;
- robot ma rozpoznać N przedmiotów $1 \leq N \leq 4$ i przesłać ich usytuowania.

Aby zwiększyć kompleksowość testów, przedmioty zostają zawszę umieszczone na lub obok linii łączącej robota z punktem docelowym. Takie założenie pozwala jednym testem zbadać kilka aspektów funkcjonalności systemu. W ten sposób każdy rozkaz przejścia do punktu o współrzędnych (x,y), czyli **GoTo(x,y)**,

zapewnia możliwość przetestowania identyfikacji. Nie pozwoliło to precyzyjnie ocenić czasu pracy algorytmu SLAM, ponieważ wykluczone zostały przypadki trywialne. Jednak testy sprawdziły dokładność:

- mapy budowanej;
- ustalenia pozycji początkowej;
- osiągniętej pozycji docelowej;
- wykrywania obiektów na podstawie mapy.



Rys. 3. Model eksperymentalny robota Lego Mindstorms NXT
Fig. 3. Experimental model of the robot Lego Mindstorms NXT

Przeprowadzone eksperymenty pokazały dużą zależność wyników od usytuowania przedmiotu w stosunku do ścian pomieszczenia. Ponieważ ten problem jest aktualnie rozwiązywany i dotychczasowe postępy pracy rokują jego kompletną eliminację, testy zostały przeprowadzone pod warunkiem, że wszystkie prostopadłościany usytuowane są równolegle do ścian pokoju.

Aby przedmioty zostały prawidłowo rozpoznane, musiałyby się one różnić wymiarami minimum o 5cm. Granica ta mogłaby zostać przekroczona, choć wpłynęłoby to mocno na czas identyfikacji. W tym celu testy zostały przeprowadzone w kilku różnych konfiguracjach. Pierwsza kolumna tablicy Tab.1 nazwana **RRmin** prezentuje minimalną różnicę między rozmiarami przedmiotów rozpoznawalnych. Skuteczność rozpoznawania obiektów zaprezentowana jest w trzeciej kolumnie.

Ocena dokładności pozycji początkowej **Start** oraz pozycji końcowej **Stop** (tab.1) została wyliczona na bazie rzeczywistych pozycji zmierzonych przed i po zakończeniu każdego testu i podawanych przez robota współrzędnych. Wszędzie podane są uśrednione wartości, przy czym każda seria testów była powtarzana 10 razy dla poszczególnych konfiguracji przedmiotów. Dokładność map tworzonych przez robota była oceniana subiektywnie pod kątem poprawności topologicznej. W każdym przypadku mapa była zadowalająco poprawna.

Tab. 1. Wyniki testów
Tab. 1. Test results

RRmin, cm	Start, %	Stop, %	Rozpoznanie obiektów, %	Czas, s
5	95	49	56	509
10	91	62	60	257
15	92	69	69	224
20	93	73	75	208

Minimalna różnica między rozmiarami (RRmin) nie może mieć wpływu na dokładność pozycji początkowej i końcowej, choć z wyników testów nasuwa się pewna zależność: im mniejsza różnica RRmin tym mniejsza dokładność pozycji końcowej. Jednak tu niewątpliwa jest rola nakładającego się błędu odometrycznego [2, 3]. Ostatnia kolumna przedstawia czas przejazdu. Dla wszystkich badanych przypadków punkt początkowy i końcowy są takie same, a odległość między nimi po linii prostej wynosi 180cm.

4. Wnioski

Doświadczenia zdobyte w czasie konstrukcji i testów robota wskazują na to, że zestaw Lego NXT nie jest łatwą platformą do stworzenia robota ogólnego przeznaczenia ze względu na mnogość problemów wynikających z niedoskonałości sensorów i silników.

Sensory odległości oferujące jeden odczyt naraz, który jest często obarczony znacznym błędem, nie pozwalają na korzystanie z wyrafinowanych metod określania pozycji robota. W związku z tym nawigacja nie jest wystarczająco dobra, aby można było przy jej pomocy wykonywać długie bądź złożone czynności. Sytuacja ta mogłaby się poprawić po dodaniu sensorów, niewchodzących w skład zestawu Lego NXT, oferujących dokładniejsze pomiary.

Algorytm rozpoznawania obiektów, użyty w pracy, mimo swojej prostoty osiągnął wysoką trafność. Oznacza to, że wykrywanie obiektów nie musi być zadaniem kosztownym obliczeniowo. Istnieje możliwość poprawy zdolności wykrywania szerszej listy obiektów poprzez dodanie kolejnych sensorów z zestawu do robota, jak również poprzez alternatywną strategię skanowania obiektów [5].

Jeśli wziąć pod uwagę projekt, jako całość, można mieć pewność, że interfejs użytkownika podobny do gier RTS (ang. *Real-Time Strategy*) nadaje się do zastosowania w robotach mobilnych. Podział na akcje i obiekty pozwala na zautomatyzowanie czynności, jednak nie zmniejsza możliwości człowieka do kontroli robota. Wydawanie poleceń w formie łączenia obiektu i zadania sprawdza się, jako metoda tworzenia intuicyjnego interfejsu, upodabniając interakcję do języka mówionego, w którym polecenia składają się z podmiotu i orzeczenia.

W niektórych przypadkach spontaniczna aktywacja czujnika robota powodowała oznaczenie pustej przestrzeni, jako zablokowanej. Napotykając taką przeszkodę na swojej mapie, robot ma znacznie większe szanse na stwierdzenia braku możliwości manewru. Dzięki zastosowanemu algorytmowi budowy mapy, po pewnym czasie udaje się te błędy skorygować, jednakże, ponieważ robot nie ma możliwości stwierdzenia, że odczyt jest fałszywy, mapa naprawiana jest dopiero po dłuższym czasie pracy.

Skuteczność użytego algorytmu rozpoznawania obiektów, osiągając poziom 75%, jest stosunkowo dobra dla zadania, z którym ten robot miał do czynienia.

Pozytywne wyniki takiej próby wskazują, że można liczyć na zestawy Lego NXT w kwestii eksperymentalnej weryfikacji różnych nowoczesnych podejść wymagających pokładów mobilnych, a jednocześnie można ominąć duże koszty sprzętowe.

Artykuł jest finansowany z pracy statutowej S/WI/4/2008.

5. Literatura

[1] Haidegger T., Benyó B., Kovács L., Benyó Z.: Force Sensing and Force Control for Surgical Robots, Proc. of the 7th IFAC Symposium on Modelling and Control in Biomedical Systems, Denmark, 2009.
[2] Tsubouchi T., Shigematsu B., Yuta S., Ohno K.: Differential GPS and odometry-based outdoor navigation of a mobile robot, Advanced Robotics, VSP and Robotics Society of Japan, Vol. 18, 2004.
[3] Borenstein J., Everett H.R. and Feng L.: Where am I? Sensors and Methods for Mobile Robot Positioning, The University of Michigan, 1996.
[4] Pfingsthorn M., Slamet B., Visser A.: A Scalable Hybrid Multi-robot SLAM Method for Highly Detailed Maps, RoboCup 2007, LNAI 5001, pp. 457-464, Springer-Verlag Berlin Heidel-berg 2008.
[5] Deyle T., Nguyen H., Reynolds M., Kemp Ch.C.: RF Vision: RFID RSSI Images for Sensor Fusion and Mobile Manipulation, IEEE Intelligent Robots and Systems, IROS 2009.

otrzymano / received: 13.05.2011
przyjęto do druku / accepted: 04.07.2011 *artykuł recenzowany*