ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2003
Seria: AUTOMATYKA z. 138 Nr kol. 1581

Krzysztof JASKOT
Politechnika Śląska

ZASTOSOWANIE SIECI NEURONOWYCH DO STEROWANIA ROBOTEM MOBILNYM

Streszczenie. W artykule przedstawiono zastosowanie sieci neuronowych do sterowania robotem mobilnym. Przez sterowanie w tym przypadku będziemy rozumieć zdolność do unikania kolizji z przeszkodami podczas poruszania się w nieznanym i dynamicznie zmieniającym się środowisku, jakim jest otoczenie robota. Przedstawiono również różne sposoby wykorzystania czujników jako źródła sygnałów wejściowych sieci neuronowej.

AN APPLICATION OF NEURAL NETWORK TO CONTROL OF A MOBILE ROBOT

Summary. The paper presents an application of neural network to control of a mobile robot. A mobile robot is a machine that can operate in a human-made environment. By control in this case we understand to be able to avoid collisions with obstacles while driving in unknown environment. Presented algorithm generates collision free path for the robot using neural network as a brain. Results of simulations and real application are also shown.

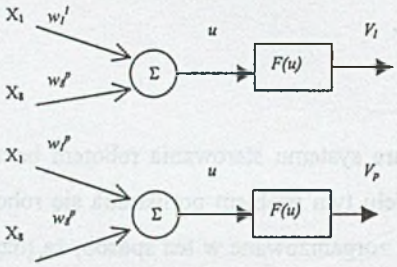
1. Wprowadzenie

Klasyczna architektura systemu sterowania robotem bazuje na tak zwanym modelu sekwencyjnym [4]. W modelu tym problem poruszania się robota dekomponowany jest na oddzielne zadania (funkcje) zorganizowane w ten sposób, że rozpoczęcie wykonania danego zadania jest uwarunkowane wykonaniem zadania poprzedniego. Wadą takiego rozwiązania jest konieczność użycia złożonego systemu sensorycznego do budowy modelu otoczenia, zapotrzebowanie na dużą moc obliczeniową realizującą zadanie budowy modelu otoczenia oraz planowania ruchu, a także brak szybkiej reakcji systemu na zmiany w konfiguracji otoczenia. Innym podejściem do sterowania robotami jest idea, która została przedstawiona

przez Braitenberga [1]. Rozpoczął on swoje badania z wykorzystaniem prostych robotów wyposażonych w pojedynczy czujnik i jedno koło. Następnie w czasie prac powiększał złożoność budowy robotów przez dokładanie kolejnych czujników, kół napędowych. Pokazał także, że niekonieczne jest tworzenie robotów inteligentnych, ale takich, które zachowują się inteligentnie. Były to próby naśladowania zachowania organizmów żywych takich, jak owady. W pracy pokazano modyfikację algorytmu Braitenberga za pomocą neuronu ADALINE. Pokazano również przykład wykorzystania sieci BAM (Bi-directional Associative Memory) do sterowania robotem. Wszystkie prezentowane poniżej algorytmy były testowane z wykorzystaniem małego robota mobilnego Khepera [6]. Robot ten wyposażony jest standardowo w osiem czujników odległości oraz układ napędowy składający się z dwóch silników. We wszystkich prezentowanych sieciach neuronowych na wejścia podawano sygnały pomiarowe z czujników.

2. Model reaktywny – algorytm Braitenberga

Istotą tego rozwiązania jest bezpośrednie połączenie modułów percepcji i wykonywania ruchów. Klasycznym przykładem modelu reaktywnego jest algorytm Braitenberga [1] – rys.1. Idea tego algorytmu opiera się na bezpośrednim połączeniu czujników z elementami wykonawczymi. Każde połączenie posiada wagę. W zależności od rodzaju zastosowanych czujników i wartości wag robot może wykonywać różne zadania: podążać do źródła światła lub go unikać, omijać przeszkody, itp.



Rys. 1. Sieć neuronowa realizująca ideę algorytmu Braitenberga
Fig. 1. Braitenberg's algorithm

Sygnały sterujące dla silnika lewego i prawego są obliczane w następujący sposób:

$$v_l = \sum_{i=1}^{8} x_i w_i^l + v \qquad (1)$$

$$v_p = \sum_{i=1}^{8} x_i w_i^p + v \qquad (2)$$

gdzie: x_i – sygnał z i-tego czujnika,

w_i^l, w_i^p – odpowiednio wagi dla lewego i prawego silnika,

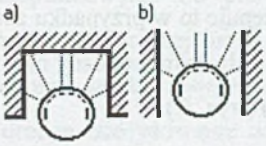
v – zadana prędkość poruszania się.

Można zauważyć, że zależności (1) i (2) są formą łącznego pobudzenia neuronu. Z tego faktu wynika, że jest to sieć jednowarstwowa posiadająca dwa neurony o liniowych funkcjach aktywacji.

Niewątpliwą zaletą tego algorytmu jest jego prostota, lecz wadą to, że nie można przewidzieć, jak w danej sytuacji zachowa się robot.

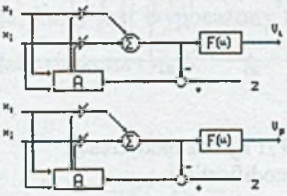
2.1. Modyfikacja algorytmu Braitenberga

W czasie poruszania się robota w nieznanym otoczeniu może się zdarzyć sytuacja, w której następuje niezamierzone zatrzymanie. Zdarza się to w przypadku dostania się robota do pułapki, jaką może być np. zatoczka – rys. 2a. Inną wadą prezentowanego algorytmu jest brak płynnego przejazdu podczas poruszania się wąskimi korytarzami – rys. 2b.



Rys. 2. Przykład pułapki
Fig. 2. Example of the trap

Naturalna wydaje się więc próba modyfikacji sieci neuronowej realizującej algorytm Braitenberga w ten sposób, aby w sytuacji, gdy robot znajdzie się w pułapce, sieć potrafiła zmodyfikować wektor wag. Od wartości wag zależy czułość robota na przeszkody.



Rys. 3. Modyfikacja algorytmu Braitenberga
Fig. 3. Modification of Braitenberg's algorithm

Modyfikacji algorytmu dokonano przez dodanie dwóch dodatkowych elementów - procesora zmian wag i detektora błędu, co zapewniło możliwość uczenia się. Uzupełniony w ten sposób neuron nazywany jest ADALINE (ADAptive LINear Element) [5]. W przypadku zastosowania tego neuronu wektor wag modyfikowany jest w sposób opisany równaniem:

$$W_N = W_S + \eta e X \tag{3}$$

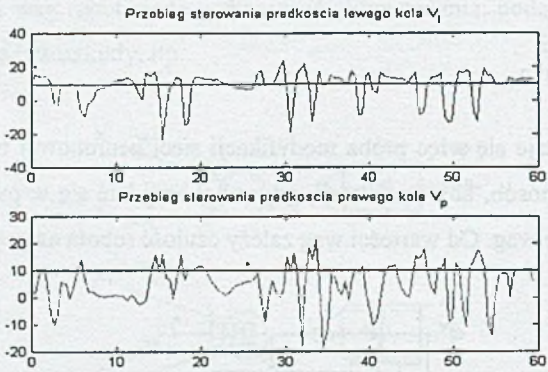
gdzie: W_N, W_S - odpowiednio nowy i stary wektor wag,

$X = \begin{bmatrix} x_1 \\ \vdots \\ x_8 \end{bmatrix}$ - sygnały z czujników,

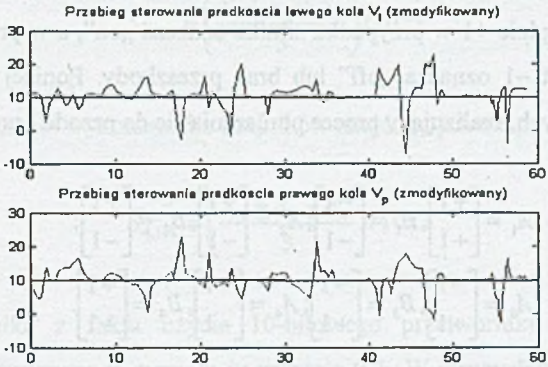
e – błąd, η – współczynnik uczenia (η =0.1).

Poniżej zostały przedstawione przebiegi sygnałów sterowania dla lewego i prawego silnika realizowane przez algorytm bez modyfikacji (rys. 4) oraz algorytm zmodyfikowany (rys. 5).

Analizując przedstawione przebiegi sterowania lewym i prawym silnikiem można zauważyć, że w przypadku algorytmu bez modyfikacji przebieg sterowania (rys. 4) jest o wiele bardziej gwałtowny niż występuje to w przypadku zastosowania algorytmu zmodyfikowanego. Można także zauważyć, że algorytm poddany modyfikacji potrafi lepiej utrzymywać wartość zadaną z prędkości (V=10) (rys. 5.)



Rys. 4. Przebieg sterowania silnikami V_l i V_p bez modyfikacji
Fig. 4. Plot of motors control without modification



Rys. 5. Przebieg sterowania silnikami V_l i V_p po modyfikacji

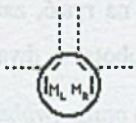
Fig. 5. Plot of motors control with modification

3. Dwukierunkowa pamięć asocjacyjna – sieć BAM

Pamięć asocjacyjna (skojarzeniowa) jest jednym z podstawowych atrybutów ludzkiego mózgu. Pamięć taka ma dwie istotne cechy, odróżniające ją w sposób zasadniczy od pamięci systemów technicznych, np. komputerowych. Po pierwsze, jak wynika z jej nazwy, informacje zarejestrowane w pamięci asocjacyjnej mogą być dostępne poprzez podanie na wejściu systemu informacji skojarzonej, selekcjonującej jedną z zapamiętanych wiadomości na drodze asocjacji, a więc na drodze zdecydowanie odmiennej od dostępu adresowego, charakterystycznego dla systemów obliczeniowych. Po drugie, ślad pamięciowy, zwany czasami engramem nie ma w pamięci asocjacyjnej ścisłej lokalizacji, gdyż każda zarejestrowana informacja zlokalizowana jest w istocie w całej pamięci, na zasadzie kolektywnego działania wszystkich jej elementów [5].

3.1. Uczenie sieci BAM i przykład jej działania

Wyobraźmy sobie robota, który jest wyposażony tylko w dwa czujniki i dwa silniki. Konfiguracja taka została przedstawiona na rys. 6.



Rys. 6. Konfiguracja robota

Fig. 6. Configuration of the robot

Powiązania między czujnikami i silnikami zostały przedstawione w postaci zbioru 4 par wektorów **A** i **B**, gdzie +1 w przypadku silnika oznacza „on", a w przypadku czujnika – przeszkodę. Natomiast –1 oznacza „off" lub brak przeszkody. Poniżej przedstawiony jest zbiór wektorów uczących, realizujący proces poruszania się do przodu i unikania kolizji:

$$A_1 = \begin{bmatrix} +1 \\ +1 \end{bmatrix}, B_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}; A_2 = \begin{bmatrix} +1 \\ -1 \end{bmatrix}, B_2 = \begin{bmatrix} +1 \\ -1 \end{bmatrix};$$
$$A_3 = \begin{bmatrix} -1 \\ +1 \end{bmatrix}, B_3 = \begin{bmatrix} -1 \\ +1 \end{bmatrix}; A_4 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, B_4 = \begin{bmatrix} +1 \\ +1 \end{bmatrix}. \tag{4}$$

Proces uczenia wag (kodowanie wag) nie odbywa się poprzez uczenie w sensie zwykłym, ale na drodze obliczeniowej, przy wykorzystaniu danych wektorów uczących **A** i **B** (4) stowarzyszonych ze sobą [3]. Macierz wag **W** jest obliczana jako suma iloczynów wektorów **A** i $\mathbf{B}^T$:

$$W = \sum_{i=1}^{N} A_i B_i^T \tag{5}$$

Macierz wag określona wzorem (5) dla danych przedstawionych wyżej przyjmuje postać:

$$W = \begin{bmatrix} 0 & -4 \\ -4 & 0 \end{bmatrix} \tag{6}$$

Macierz **W** (6) określa parametry sieci i stanowi jej pamięć. Każdy wektor otrzymany jako odpowiedź podlega normalizacji, tzn. elementy większe od zera przyjmują wartość +1, a mniejsze –1. Przykład odpowiedzi sieci na wektor A_1:

$$R = W^T A_1 = \begin{bmatrix} -4 \\ -4 \end{bmatrix} \triangleq \begin{bmatrix} -1 \\ -1 \end{bmatrix} \tag{7}$$

Jak łatwo zauważyć, otrzymaliśmy jako odpowiedź wektor B_1 (4).

W przypadku zastosowania wyżej opisanej sieci do sterowania rzeczywistym robotem, którego konfiguracja przedstawiona jest na rys.6, zaistniała potrzeba pogrupowania sześciu czujników znajdujących się z przodu robota w dwa wejścia sieci neuronowej. Zostało to zrealizowane w sposób opisany poniżej.

Sygnały z trzech czujników znajdujących się po lewej stronie robota zostały uśrednione i w przybliżeniu swoim zasięgiem pokrywają obszar około 90°:

$$\overline{S}_{1,2,3} = \left(\frac{S_1 + S_2 + S_3}{3} \right) / 1023 \tag{8}$$

Ta sama operacja została wykonana dla czujników znajdujących się po prawej stronie robota:

$$\overline{S}_{4,5,6} = \left(\frac{S_4 + S_5 + S_6}{3} \right) / 1023 \tag{9}$$

Liczba 1023 wynika z faktu użycia 10-bitowego przetwornika A/C i ma na celu znormalizowanie otrzymanego sygnału w zakresie 0-1. W przypadku użycia rzeczywistych czujników sygnał, który otrzymujemy, jest zaszumiony. Prześledźmy, jak zachowa się sieć, gdy na wejściu pojawi się taki sygnał. Wejściem naszej sieci w tym przypadku jest wektor **A** składający się z dwóch elementów będących sygnałami z czujników opisanych zależnościami (8) i (9):

$$A = \begin{bmatrix} \overline{S}_{1,2,3} \\ \overline{S}_{4,5,6} \end{bmatrix} \tag{10}$$

Przykład sygnałów otrzymanych z czujników:

$$A = \begin{bmatrix} 0.3 \\ 0.3 \end{bmatrix} \tag{11}$$

$$R = W^T A = \begin{bmatrix} -1.2 \\ -1.2 \end{bmatrix} \triangleq \begin{bmatrix} -1 \\ -1 \end{bmatrix} \tag{12}$$

Jak widać z powyższego, odpowiedzią układu jest wektor B_1 (4). Dla innych danych wejściowych:

$$A = \begin{bmatrix} 0.02 \\ -0.7 \end{bmatrix} \tag{13}$$

$$R = W^T A = \begin{bmatrix} 2.8 \\ -0.08 \end{bmatrix} \triangleq \begin{bmatrix} +1 \\ -1 \end{bmatrix} \tag{14}$$

Odpowiedzią układu jest wektor B_2 (4).

Jak widać z powyższych przykładów, mimo dużego zaszumienia sygnału wejściowego (wektory A (11),(13)), sieć podała prawidłową odpowiedź (wektor stowarzyszony).

W przypadku realizacji sterowania robotem za pomocą programu napisanego np. w C++ możemy instrukcje warunkowe *if, then, else* zastąpić przez użycie macierzy W, która posiada zapis wszystkich możliwych zachowań robota. W ten sposób struktura programu ulega dużemu uproszczeniu, a w niektórych sytuacjach możliwy jest także wzrost szybkości działania programu.

4. Zastosowanie sieci BAM do sterowania robotem

Poniżej został przedstawiony sposób praktycznej realizacji procesu sterowania robotem mobilnym z wykorzystaniem sieci BAM. W celu sterowania robotem można posłużyć się następującą procedurą:

```
W=[0 -4;-4 0]                          ; pamięć robota obliczona z (5)
while 1,
  S = kproximity(ref)                  ; odczyt z czujników
  s1 = ((S(1) + S(2) + S(3))/3)/1023   ; grupowanie sygnałów zgodnie z (8)
  s2 = ((S(4) + S(5) + S(6))/3)/1023   ; grupowanie sygnałów zgodnie z (9)
if s1 > 0.5                            ; progowanie
      s1 = 1;
elseif  s1 < 0.5
      s1 = -1;
end;
if s2 > 0.5                            ; progowanie
      s2 = 1;
elseif s1 < 0.5
      s2 = -1;
end;
  R = W'*[s1;s2]                       ; wyznaczenie sterowania zgodnie z (7)
ksetspeed(ref,R(1),R(2))               ; ustawienie prędkości
end
```

Powyższa procedura umożliwia sterowanie robotem w nieznanym i dynamicznie zmieniającym się otoczeniu. Ponieważ sygnały otrzymane z czujników pomiarowych nigdy nie osiągają wartości ujemnej, konieczne okazało się zastosowanie progowania. Przez

ustawienie wartości progu możemy regulować odległość, na jaką zbliży się robot do przeszkody. W powyższym przykładzie pojawienie się przeszkody było sygnalizowane przez „1", a jej brak przez „-1", jak występowało to w (4), jednak nic nie stoi na przeszkodzie, aby sygnały pojawiające się miały zmieniony tylko znak.

5. Podsumowanie

Prezentowane w pracy zastosowania sieci neuronowych do sterowania robotem zostały zaimplementowane z wykorzystaniem rzeczywistego robota mobilnego Khepera. Cel stawiany robotowi, czyli poruszanie i unikanie kolizji w dynamicznie zmieniającym się otoczeniu, został zrealizowany. Robot wykazywał cechy inteligentnego zachowania się. Jednak aby w pełni mówić o inteligentnym robocie, należy prezentowane algorytmy wykorzystać jako jedno z zadań w strukturze sekwencyjnej lub hybrydowej [4], gdzie poszczególne zadania koordynowane są przez warstwę wyższą posiadającą o wiele bogatszą wiedzę o otoczeniu, w którym znajduje się robot (np. mapa otoczenia [2]). Warstwa wyższa powinna także posiadać informacje o celach stawianych robotowi i sposobach ich osiągnięcia.

Niniejsza praca finansowana była z funduszu BK 200/Rau1/2002/t.1.

LITERATURA

4. Braitenberg V.: Vehicles: Experiments in synthetic psychology, MIT Press, Cambridge 1984.
5. Jaskot K.: Mapa otoczenia jako podstawa nawigacji autonomicznego robota mobilnego, XIII Krajowa Konferencja Automatyzacji Procesów Dyskretnych, Zakopane 2002.
6. Osowski S.: Sieci neuronowe, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 1996.
7. Skrzypczyk K.: Behaviour based approach to a synthesis of a mobile vehicle control system, IV Ogólnopolskie Warsztaty Doktoranckie 2002, Istebna Zaolzie.
8. Tadeusiewicz R.: Sieci neuronowe, Akademicka Oficyna Wydawnicza, Warszawa 1993.
9. Khepera User Manual, Lausanne 1999.

Recenzent: Dr hab.inż. Andrzej KRÓLIKOWSKI
Prof. Politechniki Poznańskiej

Wpłynęło do Redakcji dnia 04 lipca 2002 r.

Abstract

The paper presents an application of neural networks to reactive control of mobile robot. By control in this case we will understand to be able to avoid collisions with obstacles during drive in unknown environment. Described Braitenberg's algorithm - fig.1, and its modification by using ADALINE rules - fig.3. Results of real application are also shown - fig.5. Presents using BAM (Bi-directional Associative Memory) algorithm for modelling intelligent behaviour. Results of computer simulations and real application are also shown. Preliminary results of work on full autonomous mobile robot that can operate in human environment are presented.