Sterowanie parkowaniem pojazdu z wykorzystaniem logiki rozmytej

Amadeusz Nowak, Roman Regulski

Zakład Urządzeń Mechatronicznych, Politechnika Poznańska

Streszczenie: Artykuł dotyczy problemu automatycznego parkowania pojazdu oraz sterowania wykorzystującego zasady logiki rozmytej. Przedstawiono istotę zagadnienia i zastosowane uproszczenia. Opracowano model symulacyjny w programie MATLAB/Simulink i wizualizację procesu, pozwalające na przeprowadzenie eksperymentów dotyczących parkowania i sterowania rozmytego. Baza reguł sterownika logiki rozmytej została wygenerowana automatycznie na podstawie danych numerycznych w oparciu o algorytm Wang+. Przeprowadzono dodatkowe eksperymenty badające wpływ zmian w strukturze sterownika na jakość parkowania. Na podstawie tego sterownika zaprojektowano stanowisko laboratoryjne, wykorzystując zestaw LEGO Mindstorms.

Słowa kluczowe: logika rozmyta, parkowanie, MATLAB, Simulink, LEGO Mindstorms

1. Problemy automatycznego parkowania

Projektant systemu automatycznego parkowania pojazdu musi brać pod uwagę bardzo dużą liczbę czynników wpływających na przebieg procesu. Aby pojazd mógł poprawnie zaparkować w określonym miejscu w określony sposób, system musi ocenić sytuację (pozycję), w jakiej znajduje się pojazd. W zagadnieniu parkowania automatycznego przyjmuje się następujące założenia:

1) pojazd może poruszać się po określonej przestrzeni (plac manewrowy),
2) miejsce i sposób parkowania są ściśle określone,
3) cel sterowania – pojazd pozostawiony w dowolnym miejscu, w dowolnej orientacji jest w stanie w pełni automatycznie znaleźć miejsce postojowe i poprawnie do niego wjechać tyłem, o ile pozwoli mu na to przestrzeń manewrowa.

Ponadto układ powinien, w celu zapewnienia bezpieczeństwa, odpowiednio reagować na otoczenie i zachodzące w nim zmiany, oraz omijać przeszkody. W rzeczywistych systemach automatycznego parkowania pożądaną funkcją wspomagającą decyzyjność kierowcy jest urządzenie wyszukujące odpowiednie miejsce postojowe. Należy zaznaczyć, że warunki parkowania mogą się bardzo różnić w zależności od miejsca. Przykładowo, na parkingu za punkt odniesienia można obrać oznakowania poziome jezdni, lecz pod warunkiem, że będą w dobrym stanie, nałożone poprawnie, nieprzykryte żadnymi przedmiotami, a pozostałe pojazdy będą poprawnie ustawione itd. Doświadczenie mówi, że taka sytuacja zdarza się rzadko i nie jest to uniwersalna metoda. Jako drugi wariant punktami orientacyjnymi mogą być pojazdy już ustawione na placu. Niestety, każdy pojazd ma inną konstrukcję i system musi autonomicznie rozpoznać gdzie kończy się pojazd. Kolejny problem występuje, gdy plac jest pusty.

Pojazd w zagadnieniu parkowania można uprościć do bryły o określonych wymiarach, o trzech stopniach swobody: współrzędne placu tj. x , y, kąt obrotu względem osi parkowania – β [1]. Dla potrzeb symulacji komputerowej to wystarczający model, jednak w rzeczywistych warunkach trudno jednoznacznie i poprawnie określić punkt centralny układu współrzędnych, czy jakikolwiek punkt odniesienia. Systemy automatycznego parkowania muszą być tak stworzone, aby mogły pobierać informacje z otoczenia i przetwarzać je, co czyni je niezwykle złożonymi.

Z powyższego wynika, że najistotniejszymi problemami automatycznego parkowania są:

1) prawidłowe rozpoznanie pozycji i orientacji pojazdu w różnych, często zmiennych warunkach,
2) prawidłowe określenie przestrzeni postojowej, również jej pozycja i orientacja względem pojazdu,
3) prawidłowe wyznaczenie trasy przejazdu samochodu stosownie do sytuacji.

Zakres niniejszej pracy ograniczy się do przedstawienia wariantu rozwiązania ostatniego z wymienionych aspektów parkowania, oraz skupi się na sposobie sterowania.

1.1. Systemy parkowania w pojazdach

Układy wspomagania parkowania (ang. *park assist*) są obecnie montowane fabrycznie w coraz większej liczbie samochodów osobowych. Stosowane rozwiązania umożliwiają automatyczne wykrycie miejsca postojowego w trakcie przejazdu, oszacowanie czy pojazd zmieści się w określonej przestrzeni, korektę wyboru miejsca przez użytkownika, oraz automatyczne zaparkowanie prostopadłe lub równoległe. Osoba kierująca steruje tylko i wyłącznie prędkością samochodu, kierowanie pozostawiając układowi elektrycznemu pojazdu. Każdy ze sprzedawców systemów parkowania zaznacza, że za prawidłowy przebieg manewru odpowiedzialny jest kierowca. Musi on znajdować się w samochodzie i zareagować w razie zagrożenia kolizji. Niewielki ruch kierownicą powoduje natychmiastowe wyłączenie systemu [8]. Świadczy to o niedoskonałości systemów, pomimo zastosowania wielu różnych czujników pobierających informację z otoczenia.

Rys. 1. Toyota – Intelligent Parking Assist System (IPAS)
Fig. 1. Toyota – Intelligent Parking Assist System (IPAS)

2. Logika rozmyta

Logika rozmyta (ang. *fuzzy logic*) jest najczęściej łącznikiem pomiędzy określeniami lingwistycznymi, potocznymi, niedokładnymi, a ścisłymi określeniami matematycznymi, potrzebnymi do opisu danego zagadnienia. Pozwala wykorzystać własne doświadczenia do rozwiązania problemu. Wykorzystywana jest m. in. w technice sterowania, kiedy znana jest reakcja systemu na określone bodźce lub też schemat sterowania, ale nieznany jest model matematyczny procesu (bądź jest skomplikowany). Kilka z zastosowań logiki rozmytej to sterowanie klimatyzatorami, nadzorowanie wentylacji tuneli podziemnych, sterowanie w pralkach [1]. Właśnie dzięki logice rozmytej nieprecyzyjne określenia takie jak „mało", „dużo", „bardzo" znajdują precyzyjne odzwierciedlenie w sygnałach sterujących. Dzięki zastosowaniu logiki rozmytej możliwe jest sterowanie na podstawie jednego, dwóch lub więcej sygnałów, dla których znalezienie odpowiednich funkcji sterujących jest skomplikowane. Sterowanie rozpoczyna się od opisu słownego procesu, dlatego można w stosunkowo prosty sposób sterować procesami nieliniowymi [2].

2.1. Podstawowe definicje

Funkcją przynależności nazywamy funkcję, która określa dla każdego elementu $x \in X$ stopień przynależności do danego zbioru rozmytego

$$\mu_A : X \to [0,1], \tag{1}$$

natomiast zbiorem rozmytym A jest zbiór par

$$A = \{(x, \mu_A(x)); x \in X\}. \tag{2}$$

Jest to uogólnienie klasycznej logiki, gdzie przynależność do danego zbioru występuje, albo jest zerowa. Rozmytość polega na tym, że dana wartość x może w pełni należeć do zbioru A ($\mu_A = 1$), nie być z nim związana ($\mu_A = 0$), ale również przynależeć częściowo do danego zbioru ($0 < \mu_A < 1$). Granica tego zbioru jest rozmyta, nieostra.

Zbiór rozmyty można definiować poprzez wypisanie poszczególnych par (specyficzny zapis symboliczny, a nie matematyczny)

$$A = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \ldots + \frac{\mu_A(x_n)}{x_n} = \sum_{i=1}^{n} \frac{\mu_A(x_i)}{x_i}, \tag{3}$$

gdzie: $x_1, x_2, \ldots x_n \in X$,
oraz poprzez zdefiniowanie funkcji matematycznej

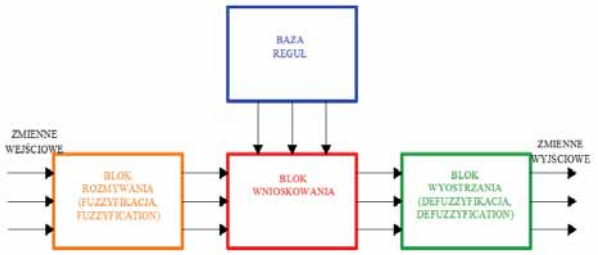
$$\mu_A(x) = \frac{1}{1 + (x-7)^2}, \tag{4}$$

gdzie: $x \in X$.

Zdefiniowano wiele operacji matematycznych związanych z logiką rozmytą. Często są to odpowiedniki działań na wartościach klasycznej logiki, niekiedy nowe pojęcia. Badacze tej dziedziny proponują różne podejścia do tych samych zagadnień [1].

2.2. Budowa sterownika logiki rozmytej

W skład podstawowego sterownika opartego na regułach logiki rozmytej (typu Mamdaniego) wchodzą cztery bloki funkcyjne, pokazane na rysunku 8: rozmywania, baza reguł, wnioskowania, wyostrzania.



Rys. 2. Sterownik logiki rozmytej typu Mamdaniego [1]
Fig. 2. Mamdani type fuzzy logic controller [1]

3. Model symulacyjny parkowania

3.1. Opis modelu matematycznego ruchu pojazdu

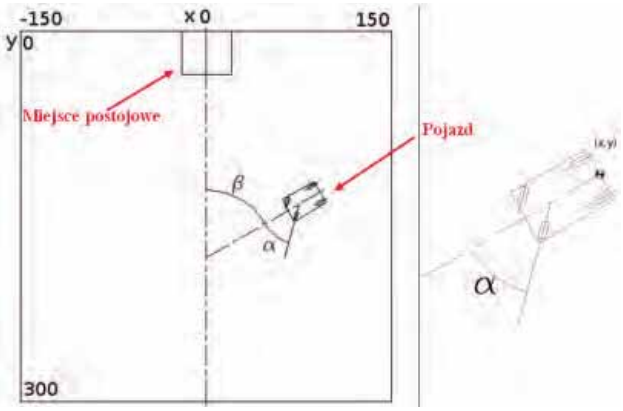
Model iteracyjny pojazdu, o stałym kroku czasowym, wykorzystuje się do sposobu obliczania jego trasy ruchu. Kolejny punkt trajektorii $(x(t+1), y(t+1))$ jest wyznaczany na podstawie aktualnego kąta skręcenia kół $\alpha(t)$ – jest to równocześnie sygnał sterujący, jego wcześniejszej orientacji względem miejsca parkingowego $\beta(t)$) i poprzedniej pozycji na placu manewrowym $[\, x(t),\ y(t)\,]$. Program wymaga podania wartości początkowych i bieżącej zmiany skrętu. Symulacja nie uwzględnia dynamiki pojazdu oraz rzeczywistych zależności towarzyszących skręcaniu samochodu, ze względu na małą prędkość ruchu (założono uproszczenie obliczeń). Pozwalające na wyznaczenie trasy ruchu pojazdu poniższe równania (5)–(7) pokazują, że mimo uproszczeń ułatwień model jest nieliniowy.

$$x(t+1) = x(t) + \sin[\alpha(t) + \beta(t)] - \sin[\alpha(t)]\cos[\beta(t)], \quad (5)$$

$$y(t+1) = y(t) - \cos[\alpha(t) + \beta(t)] - \sin[\alpha(t)]\sin[\beta(t)], \quad (6)$$

$$\beta(t+1) = \beta(t) - \arcsin\ \frac{2\sin[\alpha(t)]}{b}\ , \quad (7)$$

gdzie:
$\alpha(t)$ – aktualna wartość kąta skrętu kół pojazdu, zakres (–45°, +45°),
$\beta(t)$, $\beta(t+1)$ – aktualna i kolejna wartość kąta ustawienia pojazdu β, zakres (–180°, 180°),
$x(t)$, $x(t+1)$ – aktualna i kolejna wartość pozycji x (odcięta) pojazdu, zakres (–150, 150),
$y(t)$, $y(t+1)$ – aktualna i kolejna wartość pozycji y (rzędna) pojazdu, zakres (0, 300),
b – długość między osiami kół: 20.

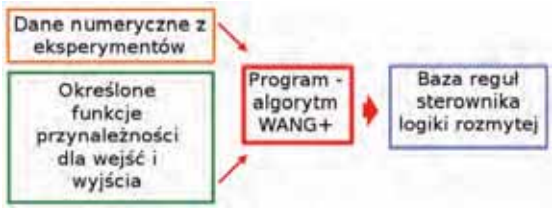


Rys. 3. Przyjęte oznaczenia w modelu komputerowym
Fig. 3. Parameters of computer model

Rysunek numer 3 przedstawia przestrzeń testową wraz z wykorzystywanymi parametrami. Zastosowane równania były już wcześniej wykorzystywane do modelowania ruchu ciężarówki podczas parkowania. Przykładem są prace [1, 2, 5].

3.2. Opis algorytmu Wang+

Wang+ zaproponowany przez M. Pilińskiego to zmodyfikowana wersja algorytmu Wang, stworzonego przez Li-Xin Wanga i J. M. Mendla, wykorzystana do opracowania systemu sterowania parkowaniem pojazdu. Tworząc sterownik logiki rozmytej projektant opiera się na doświadczeniu wiedzy dotyczącej sposobu sterowania procesem, nie znając dokładnie jego właściwości, oraz modelu matematycznego. Jeżeli sposób sterowania jest intuicyjny, liczba zmiennych wejściowych i ich funkcji przynależności jest mała, albo opis lingwistyczny jest prosty (implikacje są oczywiste), wtedy osoba tworząca system może w łatwy sposób stworzyć sterownik o wymaganych parametrach. Problem pojawia się, gdy zagadnienie jest bardziej złożone, do jego rozwiązania potrzebna jest większa liczba bieżących informacji lub proces wnioskowania nie jest oczywisty. Algorytm Wang+ pozwala na stworzenie bazy reguł na podstawie danych numerycznych, dla sterownika typu Mamdaniego z określonymi blokami rozmywania i wyostrzania. Projektant tworzy sterownik logiki rozmytej bez określania bazy reguł, lecz posiadając sygnały wejściowe sterownika i odpowiadające im sygnały wyjściowe. Można je uzyskać przeprowadzając kilka prób sterowania z udziałem operatora i rejestrując przebiegi czasowe. Powinny to być eksperymenty reprezentujące charakterystyczne schematy sterowania. Następnie program oparty na algorytmie Wang+ przetwarza odpowiednio przygotowane dane i generuje bazę reguł dla danego sterownika. Określa się, że jest to uczenie sterownika logiki rozmytej na podstawie danych numerycznych. Co więcej, metoda ta cechuje się zdolnością uogólniania, dlatego wystarczy podać tylko kilka przebiegów sterowania, charakterystycznych dla danego procesu. Algorytm wykorzystano do napisania skryptu w środowisku MATLAB. Poniżej, na rysunku 4 przedstawiono schemat działania podprogramu, Na podstawie znanej pozycji i odpowiadającemu jej kątowi skręcenia kół określa się stopnie przynależności do wszystkich zbiorów rozmytych. Następnie tworzy się wszystkie możliwe reguły typu AND i przypisuje do każdej stopień prawdziwości. Reguły są wpisywane do bazy, jeżeli są ze sobą sprzeczne to wybierana jest ta o wyższym stopniu prawdziwości (jest bardziej odpowiednia do danej sytuacji). Obliczenia wykonujemy dla kolejnych danych uczących [1, 2, 5].

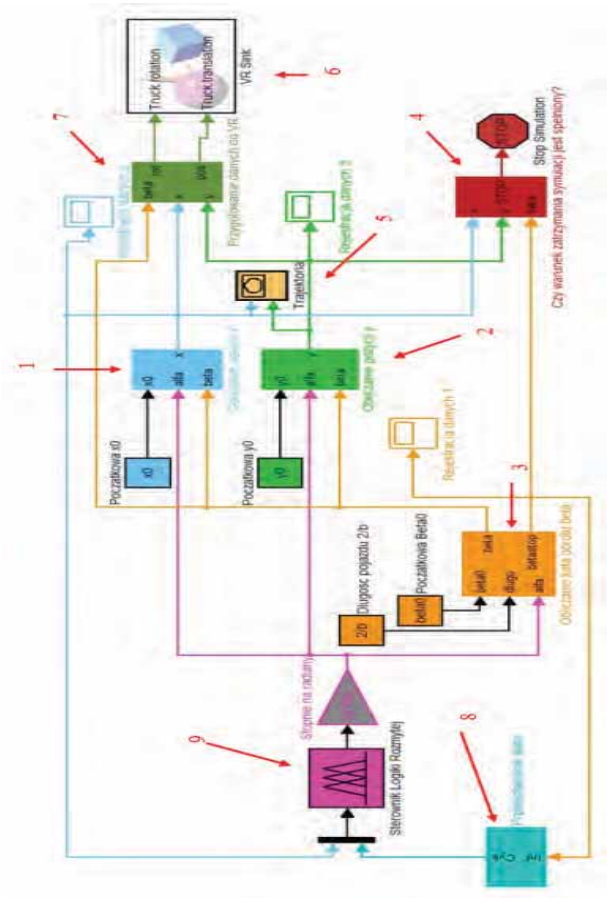


Rys. 4. Schemat działania algorytmu Wang+
Fig. 4. Functional diagram of Wang+ algorithm

3.3. Opis modelu Simulink i jego działania

Model symulacyjny automatycznego parkowania pojazdu stworzono przy pomocy programu MATLAB z dodatkiem Simulink. Jest to rozbudowane środowisko programistyczne, symulacyjne, eksperymentalne. Jego funkcjonalność można zwiększyć poprzez instalację dodatkowych pakietów – bibliotek tzw. „Toolbox". Dzięki edytorowi można stworzyć skrypty w osobnych plikach o rozszerzeniu „*.m", działające jak wewnętrzne programy obliczeniowe. Zaletą środowiska MATLAB jest dostęp do dużej liczby gotowych, często skomplikowanych, funkcji ułatwiających programowanie, oraz oprócz podstawowych operacji arytmetycznych występują odpowiedniki macierzowych. Simulink natomiast pozwala na modelowanie procesów lub obliczeń przy pomocy bloków funkcyjnych. Wygląd takiego systemu to schematy blokowe takie jak stosowane w automatyce. Wykorzystując następujące rozszerzenia programu MATLAB: Simulink, Fuzzy Logic Toolbox i Virtual Reality Toolbox, przygotowano symulację par-

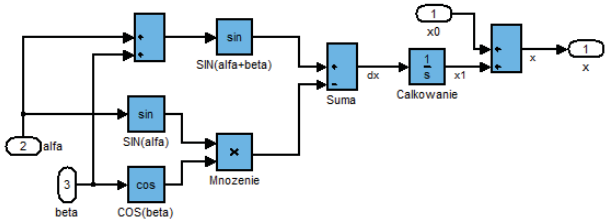
kowania pojazdu sterowanego logiką rozmytą. Na rys. 5 przedstawiono ogólny widok modelu MATLAB/Simulink. Bloki funkcyjne zostały pogrupowane w tzw. „Subsystemy", aby uporządkować wizualnie model. Bloki numer (1), (2), (3) są bezpośrednio odpowiedzialne za obliczenie trajektorii pojazdu zgodnie z równaniami (5)–(7). Grupa (4) określa czy spełnione są warunki przerwania symulacji np. czy pojazd wyjechał poza obszar ruchu. Obiekt (5) ma na celu wykreślenie trasy pojazdu w przyjętym układzie, jako możliwość weryfikacji przebiegu parkowania. Blok oznaczony numerem (6) pobrany z biblioteki Virtual Reality, jest powiązany z modelem trójwymiarowym pojazdu i dzięki wprowadzeniu odpowiednich zmiennych umożliwia wizualizację procesu parkowania (rys. 9). Elementy numer (7) odpowiadają za przygotowanie danych o pozycji i orientacji pojazdu do przesłania dalej. Subsystem numer (8) przeprowadza konieczną operację przeliczania kąta obrotu do wartości z przedziału (–180°, 180°). Sterownik logiki rozmytej, z możliwością podglądu wnioskowania, zamodelowano w bloku (9).



Rys. 5. Model parkowania pojazdu wykonany w środowisku MATLAB/Simulink

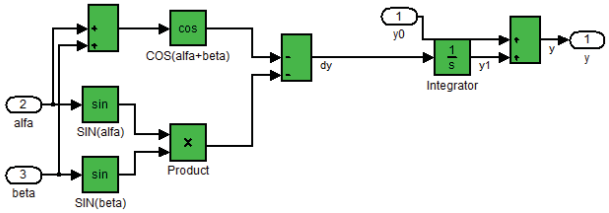
Fig. 5. MATLAB/Simulink model of vehicle parking

Poniżej przedstawiono rozwinięcia najważniejszych subsystemów, grupujących poszczególne operacje. Pierwszy, pokazany na rysunku 6, odpowiada za obliczenie pozycji x pojazdu. Dane wejściowe to aktualny kąt obrotu pojazdu („beta"), aktualny kąt skrętu kół („alfa"), oraz początkowa orientacja modelu („x0"). Odpowiada on równaniu (5). Identyczną strukturę ma subsystem numer 2, zilustrowany na rysunku 7. Jego zadaniem jest obliczenie pozycji y zgodnie z równaniem (6).



Rys. 6. Subsystem odpowiedzialny za obliczenie następnej pozycji x – blok (1)

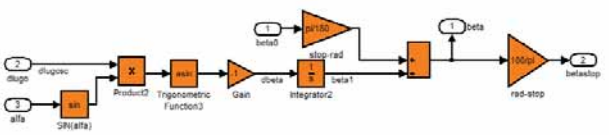
Fig. 6. Subsystem, which calculate the next position x – block (1)



Rys. 7. Subsystem odpowiedzialny za obliczenie następnej pozycji y – blok (2)

Fig. 7. Subsystem, which calculate the next position y – block (2)

Trzeci z kolei subsystem odpowiada równaniu (6), czyli wyznacza na podstawie aktualnego kąta skrętu kolejną wartość kąta obrotu pojazdu. Kąt jest obliczany w mierze łukowej (rad), ale także przeliczany na stopnie (°) – wyjście „betastop". Parametry wejściowe to początkowy kąt obrotu i długość pojazdu (rys. 8).



Rys. 8. Subsystem odpowiedzialny za obliczenie następnego kąta obrotu beta – blok (3)

Fig. 8. Subsystem, which calculate the next angle beta – block (3)

4. Wyniki symulacji

4.1. Badanie sterownika dla różnych pozycji

Wykonano model ruchu parkującego pojazdu, oraz model sterownika parkowana wykorzystujący logikę rozmytą. Sterownik w sposób nierównomierny dzieli przestrzenie zmiennych x, α na odpowiednio: 5, 7 i 7 regionów rozmytych. Baza reguł rozmytych została utworzona automatycznie z wykorzystaniem algorytmu Wang+. W celu weryfikacji poprawności sterowania parkowaniem wykonano próby: pojazd ustawiono w kilku różnych miejscach przestrzeni manewrowej, kolejno uruchomiano symulację i rejestrowano trasę przejazdu, oraz czas trwania symulacji. Sterownik logiki rozmytej ma pewne ograniczenia

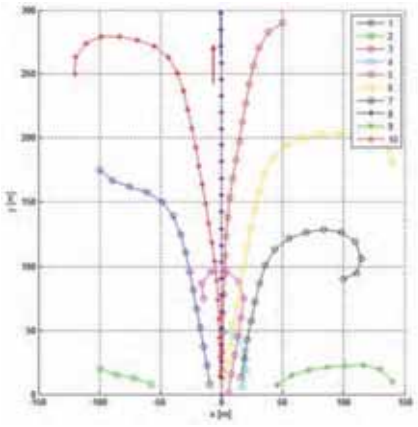
w zdolności parkowania. Układ nie jest w stanie poprawnie zaparkować z każdego punktu placu. Przykład przedstawiono na rys. 9. Problem występuje, kiedy pojazd w pozycji początkowej znajduje się zbyt blisko miejsca parkingowego, albo na granicy placu. System nie ma odpowiednio dużej przestrzeni do wykonania manewru ze względu na ograniczony skręt, niedopracowaną strategię sterowania i brak uwzględnienia ograniczeń wynikających z założeń. Istnieją następujące rozwiązania:

1) manualne zmiany reguł wnioskowania, kształtu funkcji przynależności na podstawie obserwacji,
2) wprowadzenie dodatkowych tras uczących uwzględniających kłopotliwe sytuacje,
3) wprowadzenie dodatkowej procedury wyprowadzenia pojazdu ze strefy, z której nie można przeprowadzić parkowania przy pomocy sterownika rozmytego.

Ostatnie z proponowanych rozwiązań jest wadliwe, ponieważ w tym przypadku proces sterowania nie jest już w pełni rozmyty. Możliwe jest także zwiększenie jakości sterowania ze względu na następujące parametry przykładowo: skrócenie czasu parkowania, skrócenie długości trasy parkowania, uproszczenie drogi parkowania. Wprowadzając modyfikację należy jednak pamiętać o dwóch aspektach:

1) aby móc porównywać sterowniki o różnych parametrach trzeba przeprowadzać symulację przy tych samych danych początkowych,
2) poprawa jakości sterowania dla określonych warunków początkowych może wiązać się z pogorszeniem parkowania dla innych przypadków, wręcz pojazd może mieć trudności z zaparkowaniem.

Problem parkowania nie jest zagadnieniem trywialnym, ponieważ mimo uproszczeń w sterowaniu (dwie zmienne wejściowe, uproszczona kinematyka), równania ruchu są nieliniowe i na wynik parkowania mają wpływ aż trzy zmienne: x, y, β. Dodatkowym ograniczeniem prezentowanego sterownika rozmytego jest jego podejście deskryptywne. Jak opisano w pozycji [2], pozwala ono na opracowanie schematu sterowania procesem o nieznanym, albo złożonym modelu matematycznym, w oparciu o doświadczenie operatora i dane numeryczne. Wadą tego rozwiązania jest fakt, że tą metodą można ustalić bardzo dobrą strategię sterowania, lecz nie można znaleźć rozwiązania optymalnego [2]. Na rysunku 9 przedstawiono dodatkowe przebiegi parkowania dla dziesięciu pozycji początkowych. Na ich podstawie zaobserwowano, że dany sterownik nie jest w stanie zaparkować, jeżeli jego pozycja y jest mniejsza niż 50. Kolejna uwaga: parkowanie jest w pełni poprawnie wykonane, gdy pojazd początkowo znajduje się w dalekiej odległości (powyżej 175), natomiast jeżeli jest ona umiarkowana (50–175), to pojazd parkuje niedokładnie. Przy dalszym ustawieniu pojazd manewruje dłużej i może precyzyjnie zaparkować. Zwrócono uwagę na przebieg numer 8 – jest to sytuacja wyjątkowa, ponieważ pojazd początkowo znajduje się w poprawnej pozycji, lecz jest obrócony o 180° względem docelowego kierunku. Należałoby najpierw wyjechać z miejsca parkingowego, następnie obrócić pojazd i zaparkować, lecz sterownik kieruje po linii prostej (w przybliżeniu) w stronę przeciwną do parkingu. Prawdopodobną przyczyną jest brak odpowiedniej trajektorii w zbiorze danych uczących, która uwzględniałaby taką sytuację, oraz nieodpowiedni podział przestrzeni wejściowych na zbiory rozmyte. Na rysunku zaznaczono kierunek ruchu pojazdu dla tej trasy.



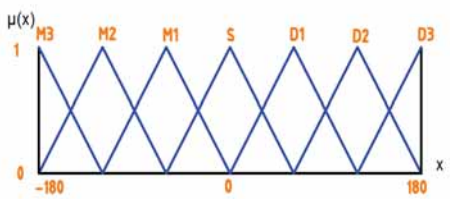
Rys. 9. Trajektorie parkowania dla dziesięciu różnych pozycji początkowych
Fig. 9. Parking routes for ten different initial positions

Tab. 1. Parametry początkowe dla przebiegów
Tab. 1. Initial parameters for routes

Nr przebiegu	1	2	3	4	5	6	7	8	9	10
Początkowa x	-100	-100	50	0	-15	140	100	0	140	-120
Początkowa y	175	20	290	50	75	180	90	0	10	250
Początkowa β	30	60	-75	90	-150	180	90	180	-150	-160

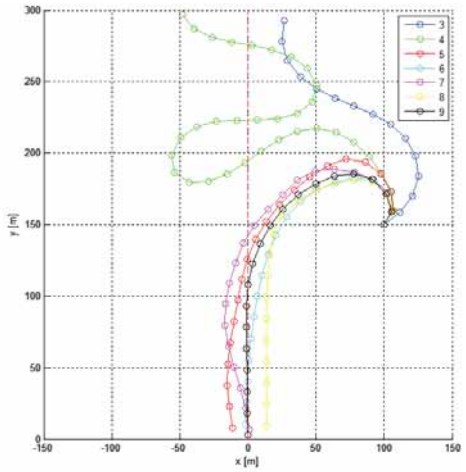
4.2. Badania sterowników o podobnej strukturze

Posiadając obszerną bazę danych numerycznych pochodzących z doświadczeń, można przeprowadzić eksperymenty polegające na badaniu wpływu zmian w strukturze sterownika rozmytego na sposób parkowania pojazdu. W pierwszej kolejności zmieniono liczbę zbiorów rozmytych w przestrzeni wejściowej i wyjściowej. Założono jednakową ilość regionów dla każdej zmiennej (wejściowych i wyjściowej). Przeprowadzono kolejno eksperymenty dla 3, 4, 5, 6, 7, 8, 9 zbiorów (liczone dla pojedynczej zmiennej). Zbiory zostały rozmieszczone automatycznie, na całej rozciągłości przestrzeni poszczególnych zmiennych. Funkcje przynależności są jednakowe, typu trójkątnego, a ich szerokość i pozycja zależą wyłącznie od liczby zastosowanych regionów. Każda zmienna może należeć maksymalnie do 2 zbiorów rozmytych. Sytuację obrazuje przykładowy rys. 10. Wyniki przedstawiono w formie graficznego wykreślenia trajektorii pojedynczego przejazdu, dla pozycji początkowych: (100, 150, 120°), (50, 150,–90°), (–50, 40, –90°). Do uczenia wszystkich sterowników wykorzystano ten sam pakiet danych uczących, oraz podprogram oparty na algorytmie Wang+. Symulacje miały na celu wykazanie, że przy uczeniu na podstawie określonych danych numerycznych, pochodzących od rzeczywistego procesu, można uzyskać różne rezultaty w zależności od struktury sterownika. Wyniki pokazują, że sterownik musi mieć odpowiednią strukturę do nauki strategii sterowania, nie może ona być dowolna.



Rys. 10. Przykładowe funkcje przynależności dla zmiennej *x* – fragment bloku rozmywania [1]

Fig. 10. Sample membership functions for variable *x* – part of fuzzyfication block [1]



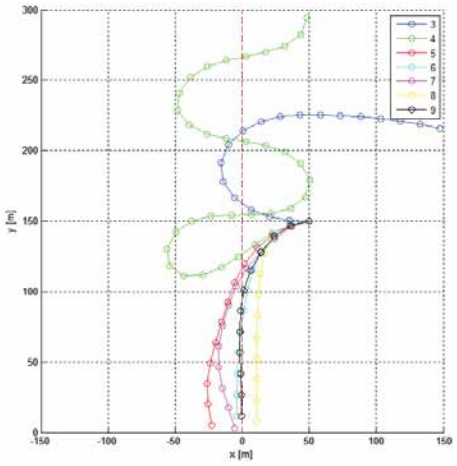
Rys. 11. Symulacje parkowania dla różnych sterowników, pozycja początkowa (100, 150, 120°), numer przy trasie oznacza liczbę zastosowanych zbiorów

Fig. 11. Simulations of parking for different controllers, initial condition (100, 150, 120°), number by the route means the number of used sets

W zależności od wymogów dokładności parkowania, można przyjąć, że sterownik musi dzielić każdą przestrzeń wejściową na przynajmniej 5 regionów rozmytych. Dla 3 zbiorów pojazd nawet nie zbliża się do parkingu, natomiast dla 4 zaczyna oscylować wokół osi parkowania, jednak oddala się od punktu (0,0) (rys. 11). Prawdopodobnie przyczyną jest brak określenia zbioru rozmytego określającego pozycję środkową (wierzchołek funkcji przynależności w punkcie 0), przez co sterownik nie może określić, czy cel sterowania został osiągnięty. Sterownik numer 5 ustawia pojazd pod odpowiednim kątem (przy zwiększonej tolerancji błędu), lecz w złej pozycji. Kolejne dwa (numer 6 i 7) parkują w pełni poprawnie.

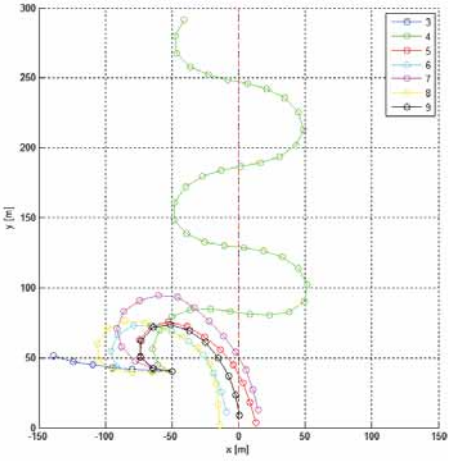
Kolejne przebiegi mogą zostać opisane w podobny sposób. Również minimalna liczba zbiorów rozmytych dla zagadnienia parkowania wynosi 5, natomiast najlepszy wynik uzyskano dla dziewięciu (rys. 12).

Ostatnia sytuacja stwarza najwięcej błędów. Wersje sterownika z liczbami regionów od 5 do 8 ustawiają pojazd pod odpowiednim kątem (przy zwiększonej tolerancji błędu), lecz w złej pozycji, obok miejsca postojowego. W pełni poprawne parkowanie przeprowadzono dla podziału przestrzeni na 9 zbiorów rozmytych (oznaczona czarnym kolorem). Trasa przejazdu jest nieskomplikowana, zmiana promienia skrętu jest płynna, cel parkowania został osiągnięty (rys. 13).



Rys. 12. Symulacje parkowania dla różnych sterowników, pozycja początkowa (50, 150, –90°), numer przy trasie oznacza liczbę zastosowanych zbiorów

Fig. 12. Simulations of parking for different controllers, initial condition (50, 150, –90°), number by the route means the number of used sets

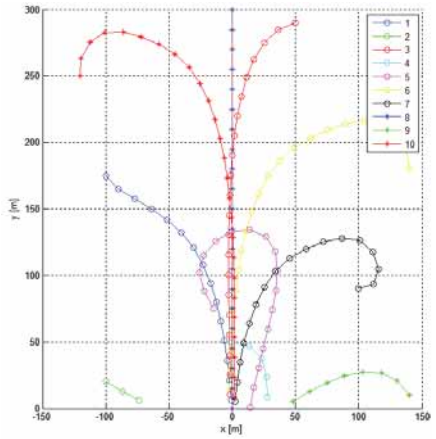


Rys. 13. Symulacje parkowania dla różnych sterowników, pozycja początkowa (–50, 40, –90°), numer przy trasie oznacza liczbę zastosowanych zbiorów

Fig. 13. Simulations of parking for different controllers, initial condition (–50, 40, –90°), number by the route means the number of used sets

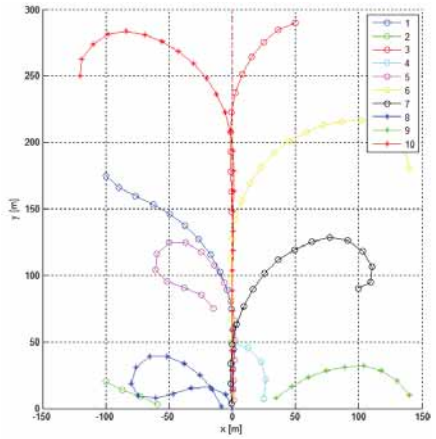
Podsumowując wyniki doświadczeń można stwierdzić, że dla określonych warunków, najlepszą strategię sterowania obiera sterownik, który dzieli przestrzenie wyjściowe na maksymalną możliwą liczbę zbiorów rozmytych, czyli 9. Dysponuje on największą bazą reguł, przez co może trafniej określić większą liczbę sytuacji, także dzieli zakresy zmiennych wejściowych i wyjściowych z większą rozdzielczością. Jeżeli sterownik ma zdefiniowaną zbyt małą liczbę regionów, wtedy niemożliwe jest jego poprawne wyszkolenie, nawet przy bardzo dobrze określonej bazie danych numerycznych. Należy jednak pamiętać, że proces parkowania może zostać także udoskonalony przez zmianę kształtu i rozmieszczenia funkcji przynależności w przestrzeni zmiennej, czyli niepowodzenie, bądź sukces parko-

wania należy rozpatrywać tylko w świetle przyjętych założeń. Nie można wykluczyć, że przy inaczej zdefiniowanych funkcjach przynależności rezultaty będą całkowicie odmienne. Dowodem tego jest porównanie sterownika numer 5, oraz sterownika opisanego w podrozdziale 4.1. Różnią się one liczbą funkcji przynależności dla zmiennej x, oraz parametrami funkcji trójkątnych, efektem czego sterownik generuje zupełnie inne trajektorie. Wadą stosowania rozbudowanych sterowników rozmytych (duża liczba zbiorów) jest długi czas i złożoność obliczeń, oraz korzystanie z większej ilości pamięci operacyjnej.



Rys. 14. Przykładowe trajektorie sterownika o podziale przestrzeni na 6 zbiorów, dla 10 punktów początkowych

Fig. 14. Sample routes for controller with 6 sets space division, for 10 initial positions



Rys. 15. Przykładowe trajektorie sterownika o podziale przestrzeni na 9 zbiorów, dla 10 punktów początkowych

Fig. 15. Sample routes for controller with 9 sets space division, for 10 initial positions

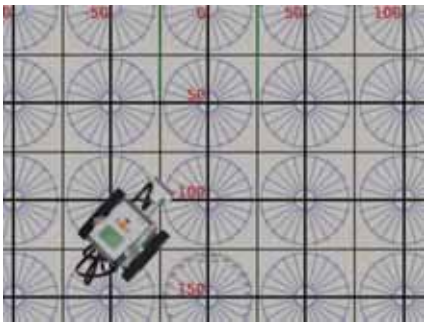
W celu lepszego porównania wybranych sterowników przeprowadzono eksperymenty dla tych samych pozycji początkowych jak w przypadku sterownika z rys. 9. Jak widać, sterownik o 6 zbiorach jest bardziej skuteczny od wcześniej przedstawionego sterownika, jednak nie jest w stanie zaparkować z pozycji: 2, 4, 8 i 9 (rys. 15). Sterownik z podziałem na 9 zbiorów, oprócz poprawy większości trajektorii, cechuje się niemal prawidłowym przebiegiem parkowania dla najtrudniejszej pozycji początkowej, czyli 8. Niemożliwe wciąż było zaparkowanie z pozycji 2, 4 i 9, stąd wniosek, że baza danych uczących powinna zostać rozszerzona o przebiegi dla tych sytuacji (rys. 16).

Oba sterowniki wykonują dłuższy przejazd przy parkowaniu z punktu numer 5 niż sterownik z rozdziału 4.1.

5. Wyniki doświadczeń

Jednym z celów pracy było stworzenie stanowiska doświadczalnego umożliwiającego badanie zachowania pojazdu podczas sterowania z wykorzystaniem logiki rozmytej. Eksperymenty przeprowadzone na tym stanowisku pozwalałyby na sprawdzenie wpływu zmian kształtu, rozmieszczenia funkcji przynależności, zmian w bazie reguł, oraz zmian w danych numerycznych, na jakość sterowania, porównanie trajektorii symulacyjnych i rzeczywistego przebiegu. Dodatkowo zestaw może pełnić funkcje edukacyjne dla studentów, jako ćwiczenie laboratoryjne. W jego skład wchodzą: robot mobilny gąsienicowy zbudowany z części LEGO Mindstorms oraz płyta wyznaczająca przestrzeń parkowania pojazdu. Komplet LEGO Mindstorms wykorzystywany jest do nauki podstaw programowania, automatyki, robotyki. Zawiera on klocki – elementy konstrukcyjne, ale również 3 serwomotory, 4 czujniki i co najważniejsze programowalny sterownik NXT 2.0. Dostępne są darmowe biblioteki umożliwiające sterowanie robotem z programu MATLAB utworzone przez studentów i pracowników RWTH Aachen University.

Podobnie jak w rzeczywistym przypadku parkowania pojazdu musi istnieć ograniczona przestrzeń ruchu. Dla potrzeb eksperymentu wykonano podstawę imitującą plac manewrowy, po której może poruszać się robot LEGO Mindstorms. Podziałkę przedstawiono w skali modelu komputerowego, czyli w zakresie (–150, 150) dla współrzędnej x i (0, 300) dla współrzędnej y.



Rys. 16. Stanowisko doświadczalne

Fig. 16. Experimental place

Ruch robota LEGO Mindstorms odbywa się na podstawie obliczonej trajektorii ruchu pojazdu parkującego, a dokładniej jego aktualnego kąta obrotu β. Przemieszczanie się jest krokowe: w każdym następuje obrót robota i przesunięcie do przodu o określony, stały odcinek. Zmiana kierunku odbywa się w miejscu, czyli gąsienice poruszają się w przeciwnych kierunkach. Kąt obrotu, jako różnica między kolejną orientacją, a poprzednią, przeliczany jest na odpowiedni kąt obrotu kół, zgodnie z wzorem (4).

Stała wyznaczona została empirycznie. Odcinki, po których pojazd się przemieszcza, zostały tak dobrane, aby trasa przejazdu odpowiadała przeskalowanej trajektorii z symulacji. Skala została obliczona również na podstawie doświadczeń.

6. Podsumowanie

Automatyczne parkowanie pojazdów jest bardzo złożonym problemem, dlatego do sterowania tym procesem wykorzystuje się złożone systemy. Nawet te najbardziej rozbudowane wymagają kontroli człowieka w celu zapewnienia bezpieczeństwa. Projektant nie jest w stanie przewidzieć wszystkich sytuacji, które mogą zaistnieć w ruchu drogowym, a urządzenie zbierające sygnały z otoczenia może zawsze mylnie zinterpretować dane. W takiej sytuacji pożądane są cechy układów sztucznej inteligencji, takie jak sterowniki logiki rozmytej, które posiadają cechy uogólniania wiedzy, przez to mogą analizować sytuacje nieprzewidziane na etapie uczenia. Algorytmy metod sztucznej inteligencji same dopasowują parametry sterownika w zależności od podanych danych wejściowych, oraz odpowiadających im sygnałów sterujących, co upraszcza projektowanie, jednak ograniczeniem jest konieczność dopasowania struktury sterownika do danego procesu. Dalszy rozwój tej dziedziny pozwoli na wzbogacenie technik sterowania i rozwiązanie wielu problemów. Kolejne badania dotyczące wykonanego modelu parkowania pojazdu powinny bazować na doskonaleniu sterownika logiki rozmytej przy pomocy symulacji komputerowej, oraz dodaniu możliwości odczytu bieżącej pozycji pojazdu LEGO Mindstorms tak, aby trasa przejazdu mogła być korygowana na bieżąco.

korygowana na bieżąco.

Bibliografia

1. Rutkowska D., Piliński M., Rutkowski L., *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte,* Wydawnictwo Naukowe PWN, Warszawa Łódź 1999.
2. Kacprzyk J., *Wieloetapowe sterowanie rozmyte*, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.
3. Piliński M., *Metodologia automatycznego uczenia sterowników rozmytych z wykorzystaniem sieci neuronowych*, praca doktorska, Częstochowa 1997.
4. Takagi T., Sugeno M., *Fuzzy identifcation of systems and its applications to modeling and control*, IEEE transactions on systems, man and cybernetics, Vol. 15, No. 1, 1985, 116–132.
5. Wang Li-Xin, Mendel J.M., *Generating Fuzzy rules By Learning from examples*, "IEEE Transactions on systems, man and cybernetics", Vol. 22, No. 6, 1992, 1414–1427.
6. [www.mathworks.com/help/pdf_doc/fuzzy/fuzzy.pdf] – *Fuzzy Logic Toolbox User's Guide*, Mathworks 2012 (11.06.2012).
7. [www.mathworks.com/help/releases/R13sp2/pdf_doc/vr/vr.pdf] – Virtual Reality Toolbox User's Guide, Mathworks (11.06.2012).
8. [http://en.wikipedia.org/wiki/Intelligent_Parking_Assist_System] – IPAS System (10.06.2012).
9. [http://www.mindstorms.rwth-aachen.de] – RWTH – Mindstorms NXT Toolbox (10.06.2012).

Vehicle parking control with fuzzy logic

Abstract: The article concerns the automatic parking of a vehicle problem and control that uses fuzzy logic rules. It presents the essence of the issue and used simplifications. The simulation in MATLAB/Simulink program and visualization of the process were elaborated, allowing to execute experiments concerning parking and fuzzy control. Rule base of the fuzzy logic controller was automatically generated with numerical data, based on the Wang+ algorithm. Additional experiments were made to check influence of changing controller structure on parking quality. It is designed educational exercise, based on the controller and LEGO Mindstorms set.

Keywords: fuzzy logic, parking, MATLAB, Simulink, LEGO Mindstorms

mgr inż. Amadeusz Nowak

Doktorant w Zakładzie Urządzeń Mechatronicznych, Wydział Budowy Maszyn i Zarządzania Politechniki Poznańskiej.
Zainteresowania naukowe: mechatronika, metody sztucznej inteligencji w sterowaniu.
e-mail:
amadeusz.nowak@doctorate.put.poznan.pl

mgr inż. Roman Regulski

Doktorant w Zakładzie Urządzeń Mechatronicznych, Wydział Budowy Maszyn i Zarządzania Politechniki Poznańskiej.
Zainteresowania naukowe: mechatronika, sterowniki mikroprocesorowe, automatyczne systemy rozpoznawania mowy, sterowanie głosowe.
e-mail: roman.regulski@doctorate.put.poznan.pl