Planowanie trasy robota Kurier w środowisku dynamicznym z wykorzystaniem sieci komórkowych

Barbara Siemiątkowska, Rafał Chojecki, Monika Różańska-Walczuk, Maciej Przybylski, Piotr Węclewski, Mateusz Wiśniowski

Wydział Mechatroniki, Instytut Automatyki i Robotyki, Politechnika Warszawska

Streszczenie: Sterowanie robotem mobilnym jest podstawowym zagadnieniem w trakcie tworzenia projektu związanego z platformą mobilną. Najnowsze trendy wskazują, że w chwili obecnej najpowszechniej rozwijane są autonomiczne systemy sterowania ruchem. W artykule przedstawiono projekt naukowy, którego celem jest zbudowanie robota z zaimplementowaną całkowicie autonomiczną nawigacją. Końcowym założeniem projektu jest wdrożenie wyników badań do realnej aplikacji. Istotnym problemem, który dotyczy nawigacji jest planowanie trasy robota. Aby platforma mogła być wykorzystana w realnym środowisku z poruszającymi sie obiektami dynamicznymi, należy je również uwzględnić w trakcie planowania trasy. W artykule dokonano porównania istniejących i proponowanych przez autorów rozwiązań planowania ścieżki. Projektując robota usługowego, który będzie mógł dzielić przestrzeń z ludźmi oraz obiektami dynamicznymi istotny jest także krótki czas reakcji na zmiany otoczenia. Problem ten rozwiązywany jest poprzez urównoleglenie mocy obliczeniowej komputera. Zaproponowaną technologią wykorzystaną w tym celu są obliczenia na procesorach graficznych GPU.

Słowa kluczowe: robotyka mobilna, nawigacja, planowanie trasy, otoczenie dynamiczne

1. Wstęp

Autonomiczne systemy sterowania ruchem są aktualnie najpowszechniejszymi tematami naukowych prac badawczych. Istnieje coraz więcej różnego typu platform mobilnych, przystosowanych do przemieszczania się w warunkach przemysłowych lub laboratoryjnych.

W artykule została przedstawiona koncepcja łącząca projekt naukowy w pełni autonomicznej nawigacji robota mobilnego z przykładem jednej z proponowanych realnych aplikacji wdrożeniowych.

Znaczącym problemem w dużych zakładach pracy, biurach, szpitalach (podobnie jak w warunkach produkcji przemysłowej) jest zagadnienie usprawnienia ogólnie pojętego transportu. W szczególności dotyczy to przenoszenia materiałów (np. dokumentów, elementów, materiałów itp.) na większe odległości.

Zautomatyzowany transport na halach produkcyjnych jest już sprawdzonym rozwiązaniem tego typu problemów. Przeniesienie takiego systemu transportu do warunków biura lub szpitala jest jednak niemożliwe lub zbyt utrudnione. Przystosowanie budynku do poruszania się autonomicznych wózków (np. odpowiednie znaczniki w podłodze, szerokie korytarze itp.) jest zazwyczaj zbyt kosztowne. Konieczny jest więc inny system sterowania robotem.

Najistotniejszym czynnikiem wyróżniającym zastosowanie robota transportującego w warunkach biurowych jest przede wszystkim fakt, że będzie poruszał się wśród ludzi. Pierwszym i najważniejszym warunkiem takiego środowiska pracy jest bezwzględne bezpieczeństwo ludzi.

Robot będzie poruszać się w przestrzeni uczęszczanej również przez pracowników i klientów (korytarze, pokoje itd.), więc środowisko to będzie zmieniać się dynamicznie: niektóre przeszkody mogą być przemieszczane lub mogą się poruszać. Ten fakt musi być uwzględniony w sterowaniu ruchem robota.

W dalszych częściach artykułu, w poszczególnych sekcjach, zostały omówione proponowane przez Autorów rozwiązania problemu.

2. Budowa robota Kurier

Projekt budowy opisywanego robota mobilnego (rys. 1) bazuje na koncepcji modułowości i podatności na modyfikacje. Założono również, aby dostęp do poszczególnych podzespołów był łatwy i nie wymagał długotrwałego demontażu wielu elementów. Konstrukcja nośna robota została wykonana z profili aluminiowych usztywnianych duralowymi panelami. Dzięki takiej budowie rama posiada dużą sztywność przy stosunkowo niskiej masie.

Robot mobilny posiada trójkołowe podwozie. Dwa główne koła napędowe o średnicy 200 mm napędzane są niezależnie przed dwa silniki prądu stałego o napięciu 24 V i mocy 100 W. Oba silniki posiadają motoreduktory. Tylne skrętne koło (wleczone) nie jest napędzane.

Wewnątrz kadłuba umieszczone zostały stopienie mocy, przetwornice DC-DC, elektryczne układy logiczne oraz akumulatory. Ze względu na wygodę obsługi, akumulatory zostały umieszczone w bocznych kieszeniach, do których dostęp możliwy jest po usunięciu bocznych paneli.

Platformę wyposażono w zestaw niskopoziomowych sensorów do wykrywania przeszkód wokół robota. Zestaw składa się z siedmiu sonarów oraz dziesięciu sensorów optycznych umieszczonych dookoła robota. Pod robotem umieszczono sensory optyczne wykrywające uskoki na trasie pojazdu.

W tylnej części pojazdu znajduje się panel, umożliwiający włączanie i wyłączanie pojazdu oraz gniazda ładowania i zasilania zewnętrznego.

Rys. 1. Robot Kurier
Fig. 1. Kurier Robot

Moduł sterujący przymocowany jest do platformy mobilnej na specjalnych szynach, dzięki czemu możliwy jest jego łatwy i szybki demontaż. W module został umieszczony komputer nadrzędny, ekran dotykowy oraz sensory wysokiego poziomu (systemy wizyjne, skaner laserowy, sensor inercyjny).

3. Środowisko dynamiczne

Otoczenie, w którym będzie przemieszczał się robot Kurier to głównie korytarze, pokoje, laboratoria itp. Są to obszary, które robot współdzieli z pracownikami i nie powinien utrudniać im pracy, poruszając się w sposób blokujący codzienny ruch.

Planowanie trasy robota korytarzami, często uczęszczanymi przez ludzi, jest znacznie bardziej złożone niż tylko odnajdowanie optymalnej ścieżki na mapie stworzonej na podstawie planów budynku. Sposób, w jaki robot postrzega otoczenie, w którym się porusza, jest bardzo istotny dla zadania planowania trasy.

W większości przypadków przeszkody są nieruchome (np. ściany, słupy, barierki itp.). Wśród przeszkód statycznych znajdują się również obiekty, które zmieniają swoje położenie w czasie – przeszkody dynamiczne. Takie obiekty muszą być klasyfikowane jako dynamiczne, gdyż dla algorytmu planowania trasy (opisany dalej w rozdziale 5) jest to zupełnie inna informacja niż np. ściana.

Przeszkody dynamiczne można podzielić ze względu na ciągłość ruchu na dwie grupy ruchu obiektów.

Pierwszy przypadek (tzw. quasi-statyczny lub semi-dynamiczny) to obiekt, który może być przestawiany co pewien czas w inny stan (pozycję) [4]. Przykładem mogą być krzesła, kosz na śmieci (przestawiane nawet nieznacznie) oraz drzwi, które mogą znajdować się w pozycji otwartej lub zamkniętej.

Drugim przypadkiem (dynamicznym) jest obiekt ruchomy, czyli wykonujący ruch w chwili obserwacji [4]. Każdy przemieszczający się człowiek czy otwierające się skrzydło drzwi będzie rozpoznane jako obiekt dynamiczny.

Rozróżnienie obiektów dynamicznych względem statycznych może odbywać się za pomocą kilku metod. Jedną z najprostszych jest wykrywanie różnic pomiędzy poszczególnymi pomiarami otoczenia, dokonywanymi z użyciem skanera laserowego SICK PLS3000.

Metoda ta w przypadku stacjonarnego pomiaru jest bardzo prosta w implementacji – wynikiem różnicy pomiarów są jedynie te obiekty, które przemieściły się w czasie Δt. Wykonując różnicowanie cyklicznie wykonywanych pomiarów, umożliwia uzyskanie kolejnych pozycji przeszkód, czyli przesunięć Δs. W rezultacie można obliczyć wynikowy wektor prędkości każdego obiektu dynamicznego.

W przypadku, gdy pomiar nie jest dokonywany stacjonarnie (np. robot przemieszcza się w trakcie wykonywania pomiaru), powyższa metoda musi być uzupełniona o blok odejmujący wektor prędkości robota od wszystkich obliczonych prędkości obiektów ruchomych. W ten sposób obiekty statyczne będą miały nadal prędkość zerową, a wektory obiektów dynamicznych będą odniesione względem układu globalnego, a nie samego robota.

Przy zastosowaniu powyższych metod możliwe jest określenie jedynie obiektów ruchomych – quasi-statyczne przeszkody będą nadal klasyfikowane jako statyczne. Istnieje możliwość sprawdzenia, który obiekt został przestawiony w inne miejsce, za pomocą porównania aktualnego otoczenia statycznego oraz mapy bazującej na poprzednich przejazdach (czyli pewnego typu historii otoczenia).

W przypadku rozważań robota Kurier – otoczenie quasi-statyczne nie będzie klasyfikowane – system planowania trasy będzie traktował te obiekty jako statyczne, dopóki nie będą znajdowały się w ruchu.

4. Planowanie trasy w środowisku dynamicznym

Zdolność omijania przeszkód jest podstawowym zadaniem, jakie powinien wykonywać robot mobilny [1, 2]. Planowanie trasy możemy zdefiniować jako określenie ciągłej krzywej, która łączy cel z bieżącym położeniem robota [3]. Dodatkowo przyjmuje się założenie, że trasa musi być pokonana przez robota, należy więc uwzględnić wymiary robota oraz ograniczenia holonomiczne i nieholonomiczne.

Rys. 2. Elementy systemu nawigacyjnego robota mobilnego
Fig. 2. Parts of navigation system of a mobile robot

W przypadku robotów autonomicznych proces planowania trasy jest ściśle związanym z innymi elementami systemu nawigacyjnego (rys. 2): tworzeniem reprezentacji otoczenia, lokalizacją i określaniem dopuszczalnych prędkości. W wielu sytuacjach wiedza o otoczeniu może być określona przed rozpoczęciem procesu planowania trasy. Dotyczy to głównie uporządkowanych środowisk statycznych. W statycznym znanym środowisku można więc zastosować

globalne algorytmy planowania trasy [10]: metody dyfuzyjne, potencjałowe, algorytmy genetyczne, sieci neuronowe i wiele innych.

Metody globalne są niewystarczające w sytuacjach przemieszczania się robota w środowisku, którego mapa nie jest całkowicie znana lub występowania przeszkód dynamicznie zmieniających swoje położenie np. ludzi. W takich warunkach zwykle oprócz metod globalnych stosowane są lokalne algorytmy planowania trasy [5–7]: systemy rozmyte, pola wektorowe, algorytm D*. W przypadku, gdy w otoczeniu przemieszcza się grupa pojazdów w procesie planowania trasy można stosować algorytmy sterowania zdarzeniowego [8, 9] lub sieci komórkowe [16].

Koncepcję neuronowych sieci komórkowych (ang. *Cellular Neural Network*, CNN) wprowadzili w 1988 roku Leon O. Chua i L. Young [11–15]. Autorzy zaproponowali, wzorując się na strukturze siatkówki ludzkiego oka, stworzenie tablicy neuronów (nazwanych komórkami), które są połączone lokalnie i przetwarzają informację w sposób równoległy. Chua wykazał, że stworzone przez niego układy są efektywnym narzędziem do wykonywania czasochłonnych takich zadań, jak przetwarzanie obrazów i rozwiązywania równań różniczkowych. Sieci komórkowe możemy traktować jako układy rozproszonego i równoległego przetwarzania informacji.

W klasycznych sieciach komórkowych przyjmuje się, że neurony są rozłożone w formie regularnej płaskiej lub przestrzennej siatki. Podstawowym elementem sieci jest komórka (neuron), który możemy traktować jako elementarny procesor przetwarzający informacje wejściowe. Zwykle neurony rozłożone są w N wierszach i M kolumnach. Położenie komórki znajdującej się w i-tym wierszu i j-tej kolumnie określamy symbolem c_{ij}. Dla każdego neuronu c_{ij} możemy określić wartość pobudzenia x_{ij}, zbiór wartości wejściowych oraz sygnał wyjściowy y_{ij}. Neurony połączone są ze sobą w obrębie sąsiedztwa. Komórka c_{kl} należy do sąsiedztwa komórki c_{ij}, jeśli dla pewnego parametru r, który jest nazywany promieniem sąsiedztwa spełniony jest warunek:

$$max(|i-k|, |j-l|) \leqslant r \quad \wedge \quad r \geqslant 0. \tag{1}$$

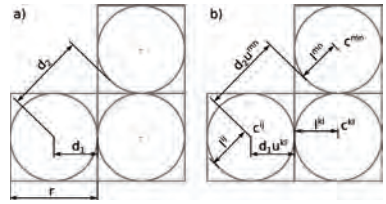
Schemat połączeń synaptycznych jest taki sam dla każdego neuronu, z wyjątkiem komórek brzegowych, dla których wprowadza się sztuczne pobudzenia o wartościach dobranych zależnie od konkretnego zastosowania. Wszystkie komórki przetwarzają sygnał w identyczny sposób. Sygnałami mającymi wpływ na stan komórki x_{ij} oraz generowany przez nią sygnał wyjściowy y_{ij} są:

– sygnały wejściowe $u_{kl} \in R$, gdzie kl jest indeksem komórki należącej do r-sąsiedztwa c_{ij},

– sygnały wyjściowe $y_{kl} \in R$ komórek należących do r-sąsiedztwa,

– sygnał $I \in R$ zwany polaryzacją.

Współczynniki wagowe określają stopień sterowania przez poszczególne sygnały. Wagi połączeń określające oddziaływanie sygnałów wyjściowych komórek sąsiedztwa noszą nazwę współczynników sprzężenia i są oznaczane symbolami a^{kl}_{ij}, ij jest indeksem komórki, która jest sterowana, a kl jest indeksem źródła sterowania. Wagi określające stopień sterowania sygnałami wejściowymi oznaczamy symbolami b^{kl}_{ij}.



Rys. 3. Odległość między komórkami a) w sensie przestrzeni (r – rozdzielczość komórki), b) w sensie czasu

Fig. 3. Cell distance a) in the sense of the space (r – resolution cell), b) in the sense of the time

W klasycznych sieciach komórkowych zakłada się, że wartości współczynników wagowych i połączeń międzykomórkowych są identyczne dla wszystkich komórek sieci. Własność ta pozwala na zapisanie współczynników w postaci macierzy A i B o wymiarach $(2r+1) \times (2r+1)$, gdzie r jest promieniem sąsiedztwa.

Wynikiem działania sieci jest zbiór wartości wyjściowych wszystkich komórek sieci po osiągnięciu stanu równowagi.

4.1. Planowanie ścieżki w środowisku statycznym

W typowych implementacjach sieci komórkowych, wagi połączeń, jak i inne parametry, nie posiadają, żadnej fizycznej interpretacji. W celu ułatwienia ilościowego porównania różnych ścieżek zaproponowano taką interpretację:

– stan x, oraz wartość wyjściowa y, jest estymowanym czasem przejazdu do celu $[s]$,

– wagi połączeń a^{ij}_{kl} są bezwymiarowe i przyjmują wartość z przedziału $\{0, 1\}$,

– wartość wag połączeń b^{ij}_{kl} są równe odległości między sąsiednimi komórkami $\{d_1, d_2\}$ (rys.3) $[m]$,

– sygnał wejściowy $u = \frac{1}{V}$ jest odwrotnością prędkości robota w danej komórce $[s/m]$,

– wartość obciążenia $I = \frac{r/2}{V}$ jest czasem przejazdu przez daną komórkę $[s]$.

Obliczenie nowego stanu komórki opisane jest wzorem (2,3), gdzie czas jazdy do celu z danej komórki jest sumą czasu przejazdu między sąsiednimi komórkami $b^{ij}_{kl} \cdot u_{kl}$, całkowitego czasu jazdy do celu z komórki sąsiedniej $a^{ij}_{kl} \cdot y_{kl}$, oraz czasu jazdy przez aktualną komórkę I^{ij}.

$$x^{ij} = \min_{kl \in N_{ij}} (a^{ij}_{kl} \cdot y_{kl} + b^{ij}_{kl} \cdot u_{kl} + I^{ij}) \tag{2}$$

$$y^{ij} = x^{ij} \tag{3}$$

Każda komórka próbuje zminimalizować czas jazdy poprzez aktywację połączenia z sąsiednią komórką, które da najmniejszą wartość w komórce aktualnej. Ponieważ robot może przemieścić się tylko do jednej z sąsiednich komórek, założone jest, że tylko jedno połączenie może być aktywowane. Wagi a^{ij}_{kl} and b^{ij}_{kl} są ze sobą sprzężone. Jeżeli połączenie jest aktywne, waga a^{ij}_{kl} przyjmuje wartość 1 a b^{ij}_{kl} przyjmuje jedną z wartości odległości między komórkami d_1 lub d_2 (rys. 3). W przeciwnym wypadku obie wagi mają wartość 0. Wybór połączenia między komórkami implikuje kierunek ruchu robota. W związku z tym oraz faktem, że w komórce przechowywana jest wartość odwrotności

	y[s]	x[s]	u[s/m]	I[s]	$\lvert N_{ij}\rvert$
Wolna przestrzeń	L	L	$\frac{1}{V}$	$\frac{r}{V}$	$8(n+1)$
Przeszkoda statyczna	L	L	L	L	0
Cel (Robot w przypadku dynamicznym)	0	0	$\frac{1}{V}$	0	0

Tab. 1. Wartości początkowe parametrów komórek, gdzie L jest bardzo dużą liczbą (nieskończoność), V jest prędkością robota, r jest rozdzielczością komórki, n jest ilością przeszkód dynamicznych ($n = 0$ w przypadku planowania w środowisku statycznym)

Tab. 1. Initial cells values

prędkości robota, możliwe jest łatwe obliczenie pożądanego wektora prędkości.

Wartości początkowe parametrów komórek w konkretnych sytuacjach zostały pokazane w tabeli 1. Komórka jest oznaczona jako *wolna*, jeżeli jakakolwiek z pozycji robota jest możliwa do osiągnięcia, bez wejścia w kolizję z otoczeniem. Komórki odpowiadające przeszkodom statycznym nie mają żadnych połączeń komórkami otaczającymi, co oznacza, że te komórki nie biorą udziału w procesie optymalizacji. Aby uniknąć poruszania się robota zbyt blisko ścian, wystarczy zmniejszyć maksymalną wartość prędkości w komórkach znajdujących się blisko przeszkód statycznych.

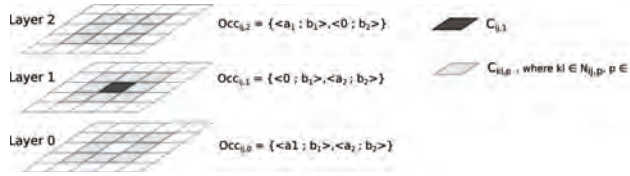
Przedstawiony algorytm oblicza ścieżkę do celu z każdego punktu bez konieczności znajomości pozycji robota. Aby dotrzeć do celu, robot podąża za połączeniami między komórkami. Wiele robotów tego samego typu (o tych samych rozmiarach) może korzystać z tej samej mapy. Ponadto mogą zostać uwzględnione obszary o dodatkowych ograniczeniach na prędkość maksymalną (np. nierówna powierzchnia) czy kierunek ruchu (np. ruch jednokierunkowy).

4.2. Planowanie ścieżki w środowisku dynamicznym

W większości systemy nawigacyjne robotów traktują obiekty napotkane na swojej drodze jako przeszkody statyczne i próbują przeliczyć ścieżkę, nawet jeżeli dany obiekt porusza się w tym samym kierunku. W tym rozdziale przedstawiamy algorytm, który rozwiązuje ten problem przy założeniu, że wiedza o dynamicznych obiektach pochodzi z bieżącej obserwacji. W związku z tym odświeżanie odczytów oraz ewaluacja sieci wykonywane są na przemian.

Występowanie przeszkód dynamicznych oznacza, że komórki mogą być niedostępne w pewnych przedziałach czasowych. Dlatego, dodatkowo, każda komórka przechowuje listę przedziałów czasowych, w których komórka jest zajęta. Aby uniknąć przeszkód dynamicznych, chwila pojawienia się robota w danej komórce musi być znana. Oznacza to, że w przeciwieństwie do planowania w środowisku statycznym, wymagana jest wiedza na temat początkowej pozycji robota.

Równania (2, 3) opisujące obliczenia wykonywane w komórce pozostają bez zmian. Zmienia się jedynie interpretacja. W tej wersji każda komórka próbuje zminimalizować czas przejazdu robota z pozycji początkowej do tej komórki. Poprzez aktywację połączenia, komórka wybiera



Rys. 4. Architektura sieci komórkowej z dwiema dodatkowymi warstwami: $c_{ij,1}$ – rozważana komórka, $c_{kl,p}$ – komórka sąsiednia, gdzie kl – indeksy komórki, p – numer warstwy L; $Occ_{ij,p}$ – zbiór przedziałów czasowych, w których komórka ij z warstwy p jest zajęta przez przeszkodę dynamiczną

Fig. 4. Dcnn architecture with two additional layers: $c_{j,1}$ – considered cell, $c_{kl,p}$ – the neighbouring cell, where kl – cell indices, p – layer number L; $Occ_{j,p}$ – the set of time intervals in which the cell ij of p layer is occupied by a dynamic obstacle

kierunek, z którego robot przybędzie, a nie kierunek następnego kroku, jak miało to miejsce w algorytmie planowania w środowisku statycznym.

Unikanie kolizji z przeszkodami dynamicznymi wymaga dostosowania prędkości robota. Prędkość może zostać zmieniona w komórce aktualnej, jak i poprzedzającej. Algorytm doboru prędkości działa następująco:

1) Jeżeli prędkość robota w komórce aktualnej jest mniejsza niż maksymalna, prędkość jest zwiększana w celu szybszego opuszczenia komórki, zanim pojawi się w niej przeszkoda dynamiczna.

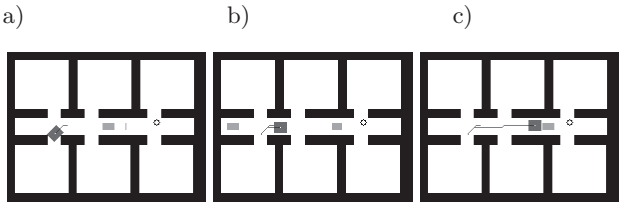
2) Jeżeli prędkość robota w komórce poprzedzającej jest mniejsza od maksymalnej, prędkość może zostać zwiększona, co sprawia, że robot wjedzie do aktualnej komórki i opuści ją wcześniej, tak aby uniknąć przeszkody dynamicznej.

3) W przeciwnym wypadku, prędkość robota jest zmniejszana w komórce poprzedzającej w celu opóźnienia momentu wjazdu robota do komórki aktualnej, tak aby robot wjechał do komórki aktualnej po tym, jak przeszkoda dynamiczna ją opuści.

Istotne jest, aby przedstawiona procedura dostosowania prędkości była wykonywana tylko wówczas, gdy przedział czasowy obecności robota w komórce pokrywa się z obecnością przeszkody dynamicznej.

Inicjalizacja komórek sieci zaprezentowana w tabeli 1 jest aktualna również dla algorytmu planowania w środowisku dynamicznym, z tą różnicą, że robot staje się w tym przypadku celem.

W pewnych sytuacjach robot musi się cofnąć, aby zrobić miejsce dla przeszkody dynamicznej, i kontynuować jazdę, kiedy droga jest wolna. Ponieważ każda komórka reprezentuje obecność robota w danym miejscu w określonej chwili, konieczne jest stworzenie dodatkowej warstwy pokrywającej ten sam obszar, ale dostępnej w chwilach po przejeździe przeszkody dynamicznej. W prostym przypadku (przy założeniu, że przeszkody nie poruszają się ruchem oscylacyjnym) liczba warstw sieci równa jest liczbie przeszkód dynamicznych +1 (rys. 4). Każda komórka w wielowarstwowej sieci ma osiem połączeń z komórkami z tej samej warstwy oraz osiem połączeń z analogicznymi komórkami z każdej z dodatkowych warstw. Warstwa odpowiadająca konkretnej przeszkodzie dynamicznej staje się dostępna dopiero po przejeździe przeszkody dynamicznej



Rys. 5. Wyznaczona ścieżka robota w trzech kolejnych chwilach: a) robot wycofuje się na chwilę, aby zrobić miejsce dla pierwszej przeszkody dynamicznej poruszającej się w przeciwnym kierunku; b) kiedy korytarz jest wolny robot kontynuuje jazdę do celu; c) robot podąża za wolniejszą przeszkodą
Fig. 5. Robot path in 3 next time steps

(nie wcześniej), podczas gdy przedziały czasowe obecności pozostałych przeszkód w tej warstwie pozostają bez zmian. W *warstwie 0* wszystkie przedziały czasowe pozostają takie same.

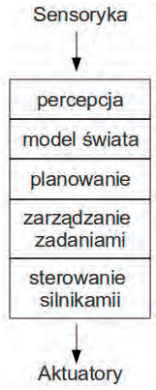
Zaprezentowany algorytm pozwala na określenie, co jest bardziej opłacalne – podążenie za przeszkodą z mniejszą prędkością czy jazda inną drogą (jeżeli taka istnieje). Jednak w przeciwieństwie do rozwiązania zaprezentowanego w poprzednim rozdziale, pozycja początkowa robota musi być znana. Dlatego tylko jeden robot może korzystać z sieci komórkowej. Aby dotrzeć do celu, robot podąża wzdłuż odwrotnej ścieżki od celu do jego pozycji początkowej. Algorytm wymaga dodatkowo znajomości wektora prędkości ruchu przeszkody dynamicznej w celu poprawnego zainicjowania przedziałów czasowych w komórkach.

Algorytm planowania ścieżki w środowisku dynamicznym został przetestowany symulacyjnie. W symulowanym scenariuszu zdarzeń robot porusza się wzdłuż wąskiego korytarza, w którym pojawiają się dwie dynamiczne przeszkody. Pierwsza z nich porusza się w kierunku przeciwnym do kierunku ruchu robota, podczas gdy druga porusza się w tym samym kierunku, ale ze znacznie mniejszą prędkością. W takiej sytuacji typowe podejście, które sprawdza się w środowisku statycznym, nie jest w stanie znaleźć ścieżki do celu. Zaprezentowany algorytm jest w stanie znaleźć najkrótszą ścieżkę w sensie czasu, nawet jeżeli musi wycofać się na chwilę.

Każdy piksel na grafice 5 odpowiada jednej komórce zerowej warstwy sieci komórkowej. Czarna linia pokazuje drogę pokonaną przez robota w kolejnych chwilach. Czarne obszary przedstawiają przeszkody statyczne, natomiast obszary jasnoszare reprezentują przeszkody dynamiczne. Aktualna pozycja robota przedstawiona jest za pomocą ciemnoszarego prostokąta. Czarny okrągły znacznik wskazuje pozycję celu.

5. Architektura współbieżna

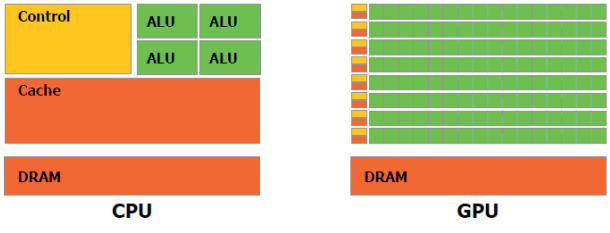
Roboty usługowe, do których zalicza się omawiany system, podczas wykonywania zadań dzielą przestrzeń z ludźmi i innymi przeszkodami dynamicznymi. Założenie to nakłada na system wymóg pracy z minimalnym czasem reakcji na zmiany otoczenia. Jednocześnie robot powinien realizować swoje cele bez wcześniejszego specjalnego przygotowania środowiska pracy. Wynika to zarówno ze względów ekonomicznych, jak i praktycznych – nie chcemy ograniczać siebie po to, żeby robot mógł skorzystać z naszej przestrzeni. To ostatnie założenie powoduje znaczne skomplikowanie architektury sterowania robotem, a co za tym idzie zwiększenie zapotrzebowania na moc obliczeniową. W przypadku bardziej złożonych systemów, wymagających architektur deliberatywnych, standardowe przetwarzanie sekwencyjne nie pozwala zachować interakcji robota ze środowiskiem w czasie niezauważalnym dla człowieka (tzw. czasie rzeczywistym). Ważna staje się re-implementacja znanych algorytmów, ich wektoryzacja i dostosowanie do koncepcji pozwalających na przetwarzanie programu w sposób równoległy i niezależny. Pierwszym podejściem jest skorzystanie z wielordzeniowości dzisiejszych procesorów. Ta metoda może być wydajna w przypadku urównoleglenia pracy warstw sterowania deliberatywnego. Rysunek 6 pokazuje powszechnie implementowaną postać szeregową tej architektury. Nadal jednak w obrębie pojedynczego rdzenia wymagane jest sekwencyjne przetwarzanie dużej ilości danych np. sensorycznych.



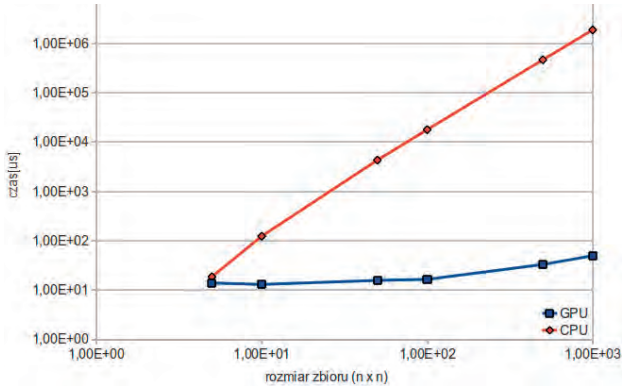
Rys. 6. Szeregowe sterowanie deliberatywne
Fig. 6. Sequential deliberative control

Jak można zaobserwować, większość danych, z jakimi spotykamy się w architekturze sterowania robotem to dane o małej ziarnistości, tj. proste typy (np. liczbowe) występujące w dużych skupiskach (dane pomiarowe, mapy, współczynniki sieci przetwarzające informacje itp.). Dodatkowo dane w tych strukturach są zazwyczaj iteracyjnie przetwarzane za pomocą identycznego lub bardzo podobnego algorytmu. Doskonałym rozwiązaniem dla takiego przetwarzania są architektury GPU (ang. *Graphics Processing Unit*). Porównanie ze strukturą zwykłego procesora CPU przedstawiono na rysunku 7. Procesory GPU składają się z setek arytmometrów mogących pracować równolegle. Rozwiązania te doskonale sprawdzają się podczas przetwarzania grafiki i fizyki na potrzeby gier.

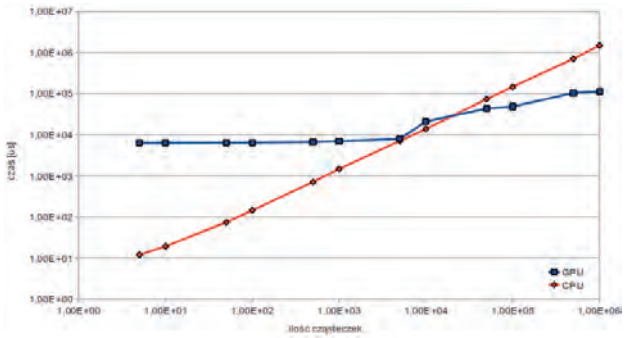
Oczywistym jest, że architektura współbieżna powinna łączyć ze sobą niezależne wykonanie warstw sterujących jak i przetwarzanie równoległe w obrębie każdej warstwy.



Rys. 7. Porównanie wewnętrznej budowy procesora CPU i GPU
Fig. 7. Comparison of the internal structure of the CPU and GPU



Rys. 8. Czas wykonania algorytmu liczenia wektorów normalnych w funkcji wielkości chmury punktów
Fig. 8. Execution time counting algorithm as a function of the normal vectors of the size of point clouds



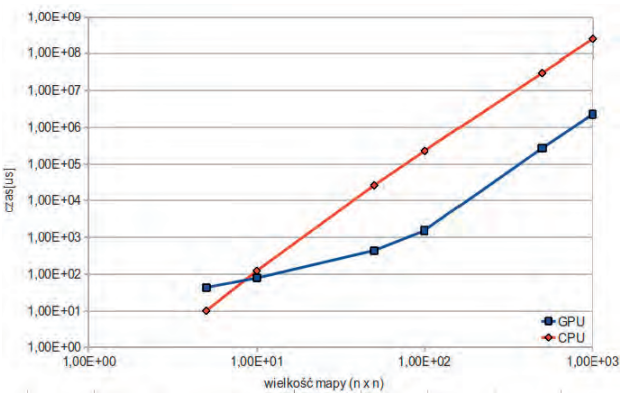
Rys. 9. Czas wykonania algorytmu filtru cząsteczkowego w funkcji wielkości zbioru cząstek
Fig. 9. Execution time filter algorithm as a function of particle size of the set

Przetwarzanie na procesorach GPU zapewnia odseparowanie obliczeń od procesora CPU co poprawia wydajność i odporność architektury. Przy niezależnym wykonaniu poszczególnych zadań można utworzyć system, który będzie odporny na przeciążania i zawieszenia pracy wynikające z obciążenia procesora. W ramach dotychczasowych badań zaimplementowano kilka elementów architektury sterowania robotem w celu potwierdzenia powyższych założeń. Implementacje objęły:

– przetwarzanie chmury punktów do zbioru wektorów normalnych,
– filtr cząsteczkowy do estymacji orientacji robota,
– sieć komórkową w zadaniu globalnego planowania ścieżki.

Każde z zadań pochodzi z innej warstwy architektury sterowania. Algorytmy opracowano na platformie NVIDIA CUDA. Każdy przetestowano na przykładowym zbiorze danych i porównano czasy wykonania (w funkcji rozmiaru danych wejściowych) z implementacjami sekwencyjnymi. Rysunki 8, 9, 10 przedstawiają uzyskane wyniki.

Jak prezentują wykresy, dla każdego z algorytmów otrzymano poprawę czasu wykonania. Mimo to część wyników nie jest optymalna i dalsze prace będą prowadzone głównie w celu dopasowania metod synchronizacji obliczeń w celu podniesienia sprawności implementacji. Wyniki pokazują, że obliczenia na procesorach GPU mogą być doskonałym narzędziem w przyspieszaniu obliczeń w robotyce mobilnej.



Rys. 10. Czas wykonania algorytmu planowania ścieżki w funkcji wielkości mapy
Fig. 10. Execution time path planning algorithm as a function of the size of the map

6. Wnioski

Część pierwszego etapu projektu dotycząca zaprojektowania robota, złożenia i przetestowania bazy mobilnej zakończona została poprzez przeprowadzenie testów i prób na robocie mobilnym. Na podstawie pierwszych testów wykazano, iż przyjęte rozwiązania są poprawne.

W artykule przedstawiono i porównano algorytmy planowania ścieżki w środowisku statycznym oraz dynamicznym. Na podstawie symulacji, która została przeprowadzona dla algorytmu planowania ścieżki w środowisku dynamicznym potwierdzono, że jest on w stanie znaleźć najkrótszą drogę przejazdu, biorąc pod uwagę czas przemieszczania się robota, uwzględniając także konieczność wycofania robota w razie pojawienia się obiektu dynamicznego.

Na potrzeby projektu przeprowadzono także testy na platformie NVIDIA CUDA. Dzięki algorytmowi, który został opracowany na daną platformę przykładowy zbiór danych porównano w funkcji rozmiaru danych wejściowych względem czasu. Symulacje potwierdziły zwiększenie wydajności czasowej wykonywania procesu. Jednak ze względu na niesatysfakcjonującą we wszystkich wynikach optymalizację procesu, prowadzone będą dalsze badania. Z pewnością należy prowadzić kolejne testy na procesorach GPU, gdyż są dobrym narzędziem, które może być wykorzystane do zwiększenia mocy obliczeniowej.

Obecnie projekt jest w trakcie realizacji części pierwszego etapu dotyczącej opracowania i testowania algorytmu filtracji oraz segmentacji danych.

Kolejne etapy pracy będą dotyczyć opracowania algorytmów klasyfikacji obiektów statycznych oraz dynamicznych, algorytmu semantycznej lokalizacji robota, a także algorytmu określania stanu obiektów dynamicznych.

Następnie planowane jest m.in. tworzenie mapy poznawczej oraz opracowanie algorytmu nawigacji semantycznej.

Projekt finansowany jest przez Narodowe Centrum Nauki w ramach grantu nr 2011/01/B/ST6/07385.

Bibliografia

1. Susnea I., Viorel Minzu, Grigore Vasiliu (2009): *Simple, real-time obstacle avoidance algorithm for mobile robots*, 8th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS'09).

2. Sariff N., Buniyamin N. (2006): *An Overview of Autonomous Mobile Robot Path Planning Algorithms*, 4th Student Conference on Research and Development (SCORED 2006).
3. J. Wan Ngah, B.N., Mohamad Z. (2010): *Point to Point Sensor Based Path Planning Algorithm for Mobile Robots*, 9th WSEAS International Conference on System Science and Simulation in Engineering Iwate, Japan.
4. Rahul Biswas, Benson Limketkai, Scott Sanner, Sebastian Thrun (2002): *Towards Object Mapping in Non-Stationary Environments With Mobile Robots*, [w:] *Intelligent Robots and Systems*, 2002 IEEE/RSJ, CA, USA.
5. Azarm K., Schmidt G. (1996): *A decentralized approach for the conflict-free motion of multiple mobile robots*, [w:] *Proc. of the IEEE International Conference on Intelligent Robots and Systems* (IROS), 1667–1674.
6. Barraquand B., Langois J.C, Latombe J. (1992): *Potential field techniques for robot path planning*, "IEEE Transactions on Robotics and Automation, Man and Cybernetics", 22(2), 224–241.
7. Bennewitz M., Burgard W., Thrun S. (2000): *Optimizing schedules for prioritized path planning of multi-robot systems*, [w:] *Proc. of the IEEE International Conference on Robotics & Automation* (ICRA).
8. Roszkowska E., Pawłowski J., Źródlak Ł. (2006): *System koordynacji i symulacji ruchu pojazdów*, Postępy Robotyki, str. 117–126.
9. E. Roszkowska, B. Kreczmer (2006): *System sterowania i symulacji ruchu pojazdów transportowych w sieci ścieżek*, "Postępy Robotyki", 107–116.
10. Latombe J.C. (1992): *Robot Motion Planning*, Kluwer Academic Publishers, MA Boston.
11. Chua L., Roska T. (1998): *Cellular Neural Networks*, "IEEE Transaction on Circuit System", vol. 2, 985–988.
12. Chua L., Roska T. (1998): *Cellular Neural Network: Theory*, "IEEE Transaction on Circuit System", vol. 35, 1257–1272.
13. Chua L., Roska T. (1988): *Cellular Neural Network*, "IEEE Transaction on Circuit System", vol. 35, 1271–1290.
14. Chua L., Roska T. (1993): *The CNN paradigm*, "IEEE Transaction on Circuit Systems", vol. 40, 147–156.
15. Chua L., Hasler M., Moschytz G. S., Neirynck J. (1995) *Autonomous Cellular Neural Networks: A unified paradigm for pattern formation and active wave propagation*, "IEEE Transaction on Circuit System", vol. 42, 559–577.
16. Siemiątkowska B. (2007): *Coordinating the Motion of Mobile Robots Using CNN*, ECMR 2005, 32–37.

Path planning in a dynamic environment based on CNN

Abstract: The control of mobile robot is a fundamental and basic task. Nowadays, the most actual trends focus on autonomous control systems. This paper describes a scientific project, which main goal is fully autonomous navigation system, designed for new construction of mobile robot. Final stage of the project is real application. The significant problem of robot navigation is path planning, especially when mobile platform is predestinated to be used in a real environment enclosed with dynamic obstacles. Dynamic objects should be considered in the path planning algorithm. The navigation system of mobile robot, moving among people, should response with short reaction time for fast environment changes. In this paper authors present parallel computing implementation, in this case – with use of graphic processors (GPU).

Keywords: mobile robotics, navigation, path planning, dynamic environment

dr hab. Barbara Siemiątkowska

Pracuje w Instytucie Automatyki i Robotyki Politechniki Warszawskiej. Jest absolwentką Wydziału Matematyki, Mechaniki i Informatyki Uniwersytetu Warszawskiego. Pracę doktorską z dziedziny zastosowań sztucznej inteligencji w robotyce obroniła w Instytucie Podstawowych Problemów Techniki PAN. Główną dziedziną zainteresowań naukowych autorki jest zastosowanie sieci komórkowych w nawigacji robotów mobilnych.
e-mail: b.siemiatkowska@mchtr.pw.edu.pl

mgr inż. Rafał Chojecki

Absolwent Wydziału Mechatroniki Politechniki Warszawskiej. Asystent w Zakładzie Urządzeń Wykonawczych Instytutu Automatyki i Robotyki. W pracy naukowo-badawczej zajmuje się zagadnieniami budowy robotów i systemów nawigacyjnych robotów mobilnych. Jest autorem kilkunastu prototypowych robotów mobilnych oraz opiekunem koła naukowego Cyborg++.
e-mail: r.chojecki@mchtr.pw.edu.pl

mgr inż. Monika Różańska-Walczuk

Absolwentka specjalności robotyka na wydziale Mechatroniki PW. W roku 2011 obroniła pracę dyplomową magisterską dotyczącą opracowania i analizy modelu robota równoległego typu Tripod. Uczestniczka studiów doktoranckich w Instytucie Automatyki i Robotyki PW. W pracy naukowej zajmuje się zagadnieniem otoczenia dynamicznego robotów mobilnych, a także systemów multiagentowych.
e-mail: m.rozanska@mchtr.pw.edu.pl

mgr inż. Maciej Przybylski

Absolwent specjalności robotyka na wydziale Mechatroniki PW. W roku 2007 obronił pracę dyplomową magisterską dotyczącą tworzenia trójwymiarowych map otoczenia przez roboty mobilne. Student studiów doktoranckich w Instytucie Automatyki i Robotyki PW. W pracy naukowej zajmuje się zagadnieniem wysokopozio-

mowego planowania akcji przez roboty mobilne z wykorzystaniem informacji jakościowej.
e-mail: maciej.przybylski@mchtr.pw.edu.pl

mgr inż. Piotr Węclewski

Jest studentem robotyki wydziału Mechatroniki PW. W roku 2011 obronił pracę inżynierską w temacie obliczeń współbieżnych dla robotyki mobilnej. Główne zainteresowania to inteligentne systemy sterowania robotów mobilnych oraz zagadnienia inżynierii kosmicznej.
e-mail: p.weclewski@mchtr.pw.edu.pl

mgr inż. Mateusz Wiśniowski

Absolwent specjalności robotyka na wydziale Mechatroniki PW. W roku 2009 obronił pracę dyplomową magisterską dotyczącą autonomicznej nawigacji odruchowej robota mobilnego. Uczestnik studiów doktoranckich w Instytucie Automatyki i Robotyki PW. W pracy naukowej zajmuje się zagadnieniem adaptacji systemu sterowania do zmiennych warunków terenowych.
e-mail: m.wisniowski@mchtr.pw.edu.pl