Krzysztof JASKOT, Krzysztof KNAPIK

Politechnika Śląska, Instytut Automatyki

Budowa mapy otoczenia z wykorzystaniem grupy robotów mobilnych

Streszczenie. *Artykuł opisuje sposób wykorzystania grupy robotów mobilnych do budowania mapy otoczenia. Przedstawiona została konstrukcja użytych robotów mobilnych oraz opisano zastosowane algorytmy budowania mapy otoczenia. Zaprezentowano wyniki działania algorytmów w rzeczywistym środowisku.*

Abstract. *The article describes how to use a group of mobile robots to build a map of the environment. Presents the design of mobile robots used and describes the algorithms to build the map of the environment. Presented the results of the algorithms in a real environment.* ***(Building the environment map using the group of mobile robots)***

Słowa kluczowe: roboty mobilne, czujniki, algorytmy, mapa otoczenia
Keywords: mobile robots, sensors, algorithms, environment map

doi:10.12915/pe.2014.12.07

Wstęp

Artykuł stanowi próbę przedstawienia rozwiązania dla mapowania otoczenia z wykorzystaniem grupy robotów mobilnych. Związane jest to z chęcią stawienia czoła problemom, które eliminowane są przez wykorzystanie tylko jednego robota. Problemy te dotyczą głównie komunikacji z robotami, wyznaczania ich pozycji oraz zapewnienia bezkolizyjnej pracy na ograniczonej przestrzeni. Przyjęte podejście zaowocowało utworzeniem kompleksowego projektu, w którego skład weszły nie tylko algorytmy przystosowane do pracy z grupą robotów i tworzenia map z ich wykorzystaniem, ale również projekt i konstrukcja samych robotów. W pracy [14] wykorzystane zostały znane już rozwiązania wykorzystywane przez roboty mobilne, ale także autorskie propozycje dotyczące rozwiązania niektórych z podstawowych problemów.

Eksploracja i mapowanie otoczenia stanowi fundamentalne wyzwanie w robotyce mobilnej. Budowa oraz praktyczne wykorzystywanie map tworzonych przez roboty uważane jest za najważniejszy problem, którego rozwiązanie jest kluczem do tworzenie prawdziwie autonomicznych urządzeń. Co prawda, ostatnie dwie dekady przyniosły w tej kwestii bardzo duże postępy, jednak pomimo istnienia już wielu zaawansowanych rozwiązań temat ten wciąż pozostaje w dużej mierze niezbadany [1].

Proces budowy mapy otoczenia jest bardzo złożony i często wiąże się z wieloma problemami. Dotyczą one przede wszystkim ograniczeń jakimi obciążone są roboty biorące udział zadaniu tworzenia mapy związanymi z koniecznością przetwarzania i gromadzenia znacznych ilości informacji. Wiele problemów nastręcza również podstawowe zadanie podczas tworzenia mapy jakim jest zorientowanie robota w przestrzeni [6, 10].

Właściwe wyznaczenie pozycji robota w przestrzeni podczas procesu tworzenia mapy otoczenia jest niezmiernie istotne. Nowe pomiary odległości do przeszkód w przypadku robotów wyposażonych w dalmierze laserowe lub ultradźwiękowe nanoszone są na mapę otoczenia zawsze względem pozycji robota [13, 16]. Niewielki błąd popełniony podczas ustalania pozycji robota, powstały na przykład podczas zliczania ilości impulsów enkoderów, w które często wyposażone są roboty, może skutkować stworzeniem błędnej mapy. Należy także pamiętać, że błędy powstałe podczas ustalania pozycji robotów mają charakter systematyczny i stopniowo narastają uniemożliwiając uzyskanie rzetelnych informacji na temat przestrzeni, w której znajduje się robot. Problem ten jest często rozwiązywany poprzez zastosowanie stacji nadających sygnał umożliwiający robotowi ustalenie swojej pozycji. Najczęściej wykorzystywanym systemem tego typu jest globalny system nawigacji GPS, jednak spotkać można również mniejsze systemy pracujące na ograniczonej przestrzeni [18].

Mapowanie niewielkich obszarów z wykorzystaniem robotów mobilnych nie stanowi problemu dla dzisiejszych komputerów. Jednak wraz ze wzrostem mapy znacznie wzrasta ilość danych niezbędnych do jej opisania. Większość przyjętych obecnie rozwiązań związanych z mapowaniem otoczenia opiera się na mapach dwuwymiarowych pomimo tego, że dostępne na rynku czujniki pozwoliłby na tworzenie zaawansowanych map trójwymiarowych. Spowodowane jest to nie tylko koniecznością przechowywania dużych ilości danych, ale przede wszystkim koniecznością przeprowadzania dodatkowych operacji na tych danych. Czas potrzebny na przetworzenie informacji stanowi wówczas duży problem [8, 17].

Poprawna interpretacja danych uzyskanych w procesie mapowania otoczenia zapisanych w postaci mapy jest zadaniem niezmiernie trudnym. Problem ten pojawia się przy mapowaniu przestrzeni zamkniętych, takich jak połączone korytarze lub przestrzenie, do których robot może uzyskać dostęp korzystając z różnych dróg (rys. 1). Pojawia się wówczas konieczność łącznia obszarów mapy, której wynik możne znacznie różnić się od rzeczywistego stanu [21].

Rys. 1. Przykład błędnie wykonanej mapy otoczenia.

Obecnie na rynku nie ma wielu dostępnych komercyjnych rozwiązań pozwalających na tworzenie map otoczenia z wykorzystaniem robotów mobilnych. Wykorzystywane do tego celu urządzenia powstają zazwyczaj w laboratoriach i ośrodkach naukowych a ich zastosowanie ma charakter czysto badawczy. Jedynymi dostępnymi na szeroką skalę robotami tworzącymi i wykorzystującymi mapę otoczenia są coraz popularniejsze roboty sprzątające. Przykładem takiego urządzenia jest robot odkurzający iRobot Roomba (rys. 2). Robot ten zdolny jest do tworzenia mapy otoczenia i praktycznego jej wykorzystywania podczas odkurzania na przykład mieszkania. Nie udostępnione są jednak żadne informacje dotyczące wykorzystanych w tym procesie algorytmów. Można jednak spotkać w literaturze próby opisu algorytmów dla te-

Rys. 2. iRobot Roomba $www.irobot.pl$

go typu robotów [4].

Konstrukcja robotów testowych

Konstrukcja platformy mobilnej robota musi zapewniać stabilność podczas poruszania się i wykonywania manewrów. Stanowi ona także miejsce montażu układów napędowych, elektronicznych, czujników oraz akumulatorów odpowiedzialnych za zasilanie robota.

Wymiary jak i kształt zastosowanej podstawy robotów przystosowane zostały do wielkości i rodzaju zastosowanych elementów, przede wszystkim silników napędzających. Pod uwagę wzięto również konieczność zachowania wymiarów, które nie będą stanowiły przeszkody przy zadaniach mapowania przeprowadzanych w wąskich przestrzeniach. Naturalnym wyborem podczas rozważań nad typem napędu robotów był napęd różnicowy. Taki sposób napędzania robota zapewnia maksymalną zwrotność i możliwość ograniczenia wymiarów robota. Główna platforma wykonana z płyty aluminiowej o grubości 4mm zapewnia wystarczającą stabilność podczas poruszania się. Dodatkowe elementy konstrukcyjne pozwoliły na odpowiednie zamocowanie niezbędnych czujników.

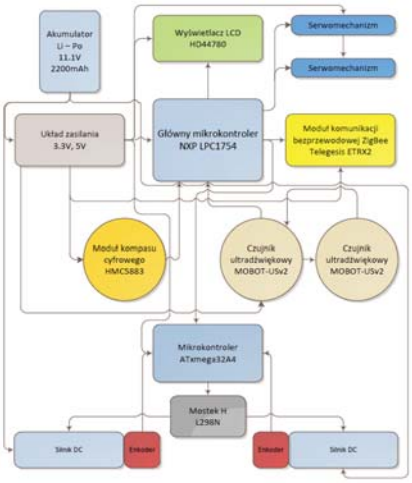
Do napędzania robotów wykorzystano silniki prądu stałego ze zintegrowanymi przekładniami planetarnymi oraz enkoderami inkrementalnymi. Każdy robot wyposażony został w dwa silniki MicroMotors E192 zasilane napięciem 12V. Nominalna maksymalna prędkość obrotowa silnika to 218RPM.

Do zasilania robotów wykorzystano akumulatory litowo – polimerowe o pojemności 2200mAh i napięciu 11,1V. Czas pracy robota na w pełni naładowanych akumulatorach wynosi w przybliżeniu 150 minut.

Projekt części elektronicznej robotów oparty został o prototypowy układ uniwersalnego sterownika robota mobilnego. Do zadań jednostki centralnej należy sterownie kierunkiem poruszania się robota, odczytywanie informacji z czujników oraz zapewnianie komunikacji bezprzewodowej z komputerem PC. Główne elementy sterownika robota przedstawione zostały na schemacie blokowym przedstawionym na rysunku 3.

Głównymi elementami układów sterowania napędami robotów są mikrokontroler ATxmega32A4 oraz mostek H L298N. Wydzielenie układu sterowania pracą silników oraz odczytu informacji z enkoderów inkrementalnych jest efektem przeprowadzonych badań. Pogodzenie potrzeby przetwarzania znacznych ilości informacji pochodzących z czujników oraz zapewnienie płynnej komunikacji bezprzewodowej w połączeniu z koniecznością ciągłego zliczania impulsów z enkoderów inkrementalnych oraz regulacji pracy silników jest zadaniem skomplikowanym, wnoszącym wiele ograniczeń. Zastosowanie układu wieloprocesorowego eliminuje udział głównego mikrokontrolera w bezpośrednim sterowaniu napędami, pozwala także, na znaczne rozbudowanie układu sterowania.

Jako główną jednostkę sterującą wybrano mikrokontroler NXP LPC1754 z rdzeniem ARM – CortexM3 [20]. Wybór ten podyktowany był dużymi możliwościami układu, ale także



Rys. 3. Schemat blokowy przedstawiający główne elementy sterownika robota mobilnego.

rosnącą popularnością mikrokontrolerów z rdzeniami ARM.

Zapewnienie stałej bezprzewodowej komunikacji z robotami było kluczową kwestią na etapie projektowania układów elektronicznych. Spośród wielu dostępnych standardów bezprzewodowej transmisji danych zdecydowano o wyborze standardu ZigBee (Telegesis). Standard ten jest powszechnie wykorzystywany w sieciach sensorowych. Pomimo niewielkiej przepustowości wynoszącej do 250 kbit/s wybrano to rozwiązanie ze względu na dużą dostępność modułów oraz łatwość implementacji.

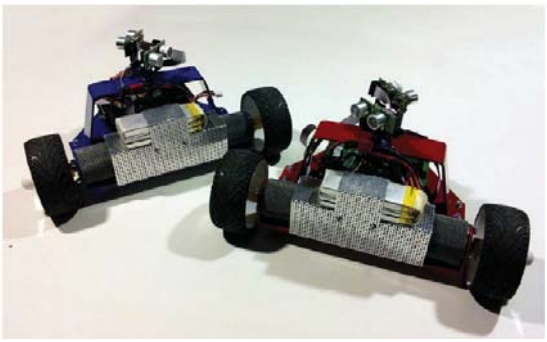
Czujniki należą do podstawowych elementów wykorzystywanych do budowy robotów mobilnych. Zadanie mapowania otoczenia wymaga od robota nie tylko umiejętności poznania przestrzeni znajdującej się wokół niego, ale także możliwości określenia swojej pozycji w tej przestrzeni. Przez pozycję rozumie się tutaj poza samym położeniem w dwuwymiarowej lub trójwymiarowej przestrzeni, także znajomość kierunku, w którym skierowany jest robot i jego odniesie do poprzednich stanów [3, 10, 12].

Roboty wyposażone zostały w ultradźwiękowe czujniki odległości (MOBOT-USv2), których zadaniem było zidentyfikowanie przeszkody oraz odległości jaka dzieli ją od robota są to podstawowe zadania przy tworzeniu mapy otoczenia. Aby uzyskać możliwie jak najwięcej informacji o otoczeniu każdy robot musiałby zostać wyposażony w co najmniej kilka czujników ultradźwiękowych. W celu obniżenia kosztów, które wiązałyby się z takim rozwiązaniem, postanowiono wykorzystać serwomechanizmy do obracania czujników ultradźwiękowych [13].

Dodatkowo każdy z robotów został wyposażony w kompas cyfrowy. Równie istotną cechą robota konieczną do prawidłowego wykonania zadania mapowania otoczenia jest umiejętność właściwego określenia swojej pozycji. Funkcjonalność ta zapewniona jest w dużej mierze przez enkodery inkrementalne zamontowane na wałach silników. Jednak w każdym przypadku wyznaczanie pozycji i kierunku robota wyłącznie na podstawie pomiarów z czujników odometrycznych, w tym przypadku enkoderów, powoduje powstawanie i propagację błędów systematycznych, które skutecznie uniemożliwiają wyznaczenie prawidłowego położenia robota. W celu wyeliminowania części problemów związanych z umiejscowieniem robotów w przestrzeni przy wykorzystaniu odometrii zastosowano kompasy cyfrowe. Rozwiązanie to pozwoliło na korygowanie pozycji robotów podczas poruszania się, a także podczas wykonywania skrętów. W konstrukcji robotów wykorzystano moduły kompasów cyfrowych

HMC5883L działające w oparciu o trójosiowe magnetometry [9].

Wygląd zbudowanych robotów przedstawiony został na rysunku 4.



Rys. 4. Testowe roboty mobilne

Algorytmy tworzenia mapy otoczenia

Problemem podczas pracy z grupą robotów przy tworzeniu map otoczenia jest ich wzajemne położenie. W przypadku małych przestrzeni nieuniknione stają się sytuacje, w których pomiar odległości wykonany zostanie w kierunku innego znajdującego się blisko robota. Wówczas interpretacja pomiaru jako wskazującego przeszkodę jest błędem. Problem ten dotyczy przede wszystkim realizacji zadania mapowania otocznia z wykorzystaniem w pełni autonomicznych robotów. Brak koordynatora nadzorującego pracę wymusza na robotach stałe wymienianie informacji na temat swojego położenia i interpretowanie wyników pomiarów z uwzględnieniem tych danych.

Występowania głównego koordynatora w stworzonym projekcie nadzorującego wykonywanie wszystkich operacji częściowo eliminuje problemy wzajemnego położenia robotów. Pomiary bez większych trudności mogą zostać zestawione z pozycjami wszystkich robotów. Pod uwagę może zostać wzięta nie tylko aktualna pozycja wszystkich urządzeń, ale również ich położenia w przeszłości. Pozwala to podczas interpretowania mapy na odszukanie przestrzeni, w której mogły pojawić się błędne odczyty związane na przykład położeniem robota względem jakiejś przeszkody. Wówczas gdy pozycja innego robota w dowolnej chwili czasu pokryje się z pozycją przeszkody możliwe staje się całkowite usunięcie niepewnego obszaru i ponowne przeprowadzenie mapowania. Każda taka operacja powoduje oczywiście wydłużenie czasu potrzebnego na wykonanie całego zadania, ale w znacznym stopniu zwiększa dokładność utworzonej mapy.

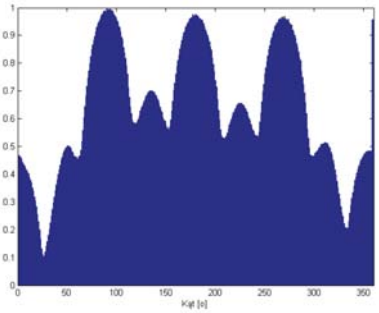
Utworzona aplikacja w środowisku MRDS (Microsoft Robotic Developer Studio) [15] na bieżąco aktualizuje listę wszystkich dostępnych robotów biorących udział w aktualnie wykonywanym zadaniu. Realizowane jest to dzięki informacji uzyskiwanym z modułu ZigBee podłączonego do portu COM komputera. Odpowiednie ustawienie modułu gwarantuje otrzymanie informacji o każdym nowym dołączającym się do sieci urządzeniu. Uzyskana informacja zostaje zinterpretowana i nowy robot zostaje włączony do sieci. Kolejne operacje inicjalizacyjne pozwalają na zebranie niezbędnych danych dotyczących aktualnej pozycji i kierunku poruszania się robota. Po uzyskaniu niezbędnych informacji nowe urządzenie może przystąpić do pracy z pozostałymi bez konieczności rozpoczynania zadania od początku.

Algorytmy tworzenia i jednoczesnego poruszania się robotów SLAM (ang. Simultaneous Localization and Mapping) po mapie otoczenia stanowią bardzo szeroki dział robotyki. Powstało bardzo wiele publikacji opisujących ten problem [5, 22]. Cechy wspólne wszystkich algorytmów SLAM pozostają jednak niezmienne. Zadaniem robota lub grupy robotów jest wykrywanie przeszkód, ich unikanie oraz ciągłe podążanie w kierunku obszaru nieznanego. Wszystko to oczywiście jest możliwe pod warunkiem posiadania informacji o swojej pozycji w chwili obecnej, jak również w przeszłości.

W artykule postanowiono podjeść do problemu mapowania otoczenia jako zbioru mniej złożonych problemów, które zestawione razem utworzą jednak kompleksowy algorytm. W skład proponowanego rozwiązania wchodzi modyfikacja popularnego przy projektowaniu robotów algorytmu VHF+ (Vector Field Histogram Plus) [23, 24] oraz modyfikacja algorytmu Self Biddings [7].

Zmodyfikowany algorytm VFH+

Algorytm Vector Field Histogram Plus (VFH+) [23, 19] jest modyfikacją algorytmu VFH zaprezentowanego przez Borenstein'a oraz Korem'a [2]. Ideą tego algorytmu jest możliwość zapewnienia robotom umiejętności omijania przeszkód w czasie rzeczywistym przy jednoczesnym zachowaniu kierunku ruchu. Metoda VFH+ wykorzystuje dwuwymiarowy histogram polowy umiejscowiony w kartezjańskim układzie współrzędnych jako model świata wokół robota. Model ten jest nieustannie aktualizowany wykorzystując dane napływające z czujników. W algorytmie VFH+ możemy wyróżnić trzy główne etapy postępowania. Pierwszym jest budowa dwuwymiarowego histogramu polowego w kartezjańskim układzie współrzędnych. Histogram ten powstaje na podstawie pomiarów odległości dokonanych przez robota. Budowa histogramu biegunowego rozpoczyna się od wyzna-



Rys. 5. Przykładowy znormalizowany histogram biegunowy utworzony dla 360 segmentów.

czenia wielkości obszaru aktywnego oraz jego podzielenia na równe segmenty. Każdy segment opisuje wycinek obszaru 360°wokół robota (rys. 5). Wielkość obszaru aktywnego oraz ilość segmentów jest dowolna. Następnie dla każdej komórki w obszarze aktywnym obliczana jest wartość kąta wyznaczającego kierunek, w którym znajduje się komórka mapy względem środka obszaru aktywnego reprezentowanego przez położenie robota. Kąt ten obliczany jest zgodnie ze wzorem:

$$\beta_{i,j} = tan^{-1}(\frac{y_0 - y_j}{x_i - x_0}), \tag{1}$$

gdzie: x_i oraz y_j to położenie aktualnie branej pod uwagę komórki mapy, x_0 i y_0 to położenie robota w przestrzeni, a β to wartość kąta wyznaczająca kierunek, w którym znajduje się aktualna komórka. Kolejnym krokiem jest wyznaczenie wartość elementu wektora reprezentującego jeden z segmentów obszaru aktywnego, wybranego na podstawie wyznaczonego kąta. Element wektora obliczany jest według wzoru:

$$m_{i,j} = c_{i,j}^2(a - bd_{i,j}^2), \tag{2}$$

gdzie i oraz j oznaczają aktualnie braną pod uwagę komórkę mapy, c to wartość w komórce, a i b to pewne stałe, a d to

odległość aktualnej komórki od punktu centralnego robota. Wartości stałych a oraz b dobiera się według wzoru:

$$a - b(\frac{w_s - 1}{2})^2 = 1, \tag{3}$$

Gdzie w_s oznacza długość krawędzi obszaru aktywnego. Po wykonaniu powyższych operacji dla całego obszaru aktywnego wartość elementów w komórkach wyznaczonych wektorów są sumowane. Na podstawie tak wyznaczonych danych budowany jest histogram polowy. Oś x histogramu opisuje kąt w stopniach natomiast oś y wartości sumy elementów wektora reprezentującego segment przestrzeni. Należy również wspomnieć, że w algorytmie VFH+ istnieje możliwość uwzględnienia przybliżonych wymiarów robota w celu zapewnienia bezkolizyjnego poruszania się. Odbywa się to przez poszerzanie komórek mapy zawierających informację o wykrytych przeszkodach o promień okręgu, którym może zostać opisany robot. Promień tego okręgu może zostać powiększony o dowolny dodatkowy obszar gwarantujący bezkolizyjną jazdę. Wartość kąta w przestrzeni o jaki poszerzana jest przeszkoda w każdym kierunku obliczana jest zgodnie ze wzorem:

$$\gamma_{i,j} = \arcsin \frac{r_{r+s}}{d_{i,j}} \tag{4}$$

gdzie r_{r+s} to promień okręgu, którym może zostać opisany robot powiększony o dowolny obszar, $d_{i,j}$ to odległość przeszkody od robota, a $\gamma_{i,j}$ to wyznaczony kąt. Drugim głównym etapem algorytmu VFH+ jest wyznaczenie binarnego histogramu polowego. Odbywa się to poprzez ustalenie wartości progu, poniżej którego wartość słupka histogramu polowego przepisywana jest jako zerowa do histogramu binarnego, a powyżej którego jako wartość równa jedności. Etap trzeci zakłada budowę maskowanego histogramu polowego, uwzględniającego kinematykę robota. Rozwiązania to pozwala na wybieranie najbardziej optymalnej drogi miedzy przeszkodami bez konieczności wykonywania bardzo ostrych skrętów, co w przypadku niektórych robotów mobilnych nie jest możliwe. Etap ten jednak nie został uwzględniony w utworzonym projekcie i nie zostanie dokładnie opisany. Modyfikacja ta podyktowana była niekorzystnym wpływem maskowanego histogramu na wyznaczenie drogi w nieznanym otoczeniu oraz zastosowaniem napędu różnicowego w konstrukcji robotów, który zapewnia możliwość wykonania ruchu w dowolnym kierunku.

Ostatnim etapem algorytmu jest wyszukanie dolin w binarnym histogramie i wartość odpowiadających im kątów opisujących kierunek. Poprzez doliny rozumie się tutaj słupki histogramu binarnego, których wartość jest zerowa. Każda dolina zostaje zinterpretowana na podstawie ustalonego przy starcie algorytmu parametru S_{max}. Dolina, której szerokość jest większa niż S_{max} uznawana jest za szeroką, a każda mniejsza za wąską. W przypadku wąskich dolin kierunek ruchu robota wyznacza się zgodnie ze wzorem:

$$c_n = \frac{k_r + k_l}{2} \tag{5}$$

gdzie k_r to wartość kąta, w którym rozpoczyna się dolina, k_l to wartość kąta, w którym kończy się dolina, a c_n to wyznaczony kierunek. Dla dolin szerokich występują dwa możliwe kierunki ruchu, jeden bliżej lewej krawędzi doliny c_l, a drugi bliżej prawej c_r. Kierunek wskazujący bezpośrednio cel c_t również może zostać wzięty pod uwagę, gdy znajduje się wewnątrz szerokiej doliny.

$$c_r = k_r + \frac{S_{max}}{2} \tag{6}$$

$$c_l = k_l + \frac{S_{max}}{2} \tag{7}$$

$$c_t = k_t \text{ jeżeli } k_t \in [c_r, c_l] \tag{8}$$

Wyznaczenie bezpośredniego celu robota może zostać przeprowadzone w dowolny sposób i zostanie opisane w dalszej części podrozdziału. W przypadku gdy znaleziono więcej niż jedną dolinę wybór właściwej definiowany jest na podstawie minimalnej wartości uzyskanej przy obliczeniu funkcji kosztów:

$$g(c) = \mu_1 \Delta(c, k_t) + \mu_2 \Delta(c, \frac{\Theta_i}{\alpha}) + \mu_3 \Delta(c, k_{n,j-1}) \tag{9}$$

gdzie μ_1, μ_2, μ_3 to wagi, k_t to kierunek wyznaczający aktualny cel, $\frac{\Theta_i}{\alpha}$ to aktualny kierunek robota, $k_{n,j-1}$ to poprzedni kierunek robota, c to aktualnie brana pod uwagę dolina, $g(c)$ obliczona wartość funkcji kosztów. Dobór wag wykorzystywanych w funkcji kosztów jest dowolny pod warunkiem spełnienia zależności:

$$\mu_1 \geq \mu_2 + \mu_3 \tag{10}$$

Poprzez właściwe dobranie wartości wag można wpływać na sposób poruszania się robota. Pierwszy człon funkcji kosztów i waga μ_1 reprezentują koszt związany z różnicą pomiędzy aktualnie braną pod uwagę doliną, a zadanym kierunkiem ruchu. Człon drugi i waga μ_2 reprezentują koszt związany z różnicą między aktualną orientacją robota, a kierunkiem wyznaczonym przez aktualnie braną pod uwagę dolinę. Trzeci człon oraz waga μ_3 reprezentują koszt związany z różnicą między poprzednim wybranym kierunkiem, a aktualnie branym pod uwagę. Im większa wartość wagi μ_1 tym bardziej sposób poruszania się robota jest zorientowany na osiągnięcie wyznaczonego celu. Większe wartości wagi μ_2 powodują wybieranie kierunku ruchu jak najbardziej zbliżonego do aktualnej orientacji robota. Duże wartości wagi μ_3 wpływają natomiast na wygładzenie trajektorii poruszania się przez wybór kierunku zbliżonego do poprzedniego.

Wyznaczenie celu robota przy wykorzystywaniu algorytmu VFH+ jest niezmiernie istotne ponieważ definiuje sposób omijania przeszkód. W omawianej pracy zastosowano podejście zaproponowane w [23] dla robotów mających zwiedzać otoczenie. Polega ono na przyjęciu za cel robota kierunku losowego w pierwszej iteracji algorytmu, a następnie kierunku poprzedniego. Takie rozwiązanie sprawia, że sposób poruszania się robota jest w pewnym stopniu losowy, ale nie chaotyczny. Poprzez dobranie odpowiednich wag wykorzystywanych w funkcji kosztów cel tak dobrany okazuje się idealny przy wykonywaniu zadania mapowania otoczenia.

Algorytm Self Biddings

Algorytm Self Biddings [7] powstał specjalnie z myślą o wykonywaniu zadania mapowania otoczenia z wykorzystaniem grupy robotów. Idea tego rozwiązania opiera się na przeprowadzaniu przetargów pomiędzy robotami dotyczących odległości do wybranego obszaru nieodkrytego. Algorytm ten zakłada zdolność robotów do samodzielnej wymiany informacji dotyczących budowanej mapy otoczenia. Jego przebieg można podzielić na kilka podstawowych kroków. W pierwszym kroku robot, który aktualnie nie posiada żadnego

celu, lub właśnie zakończył jego realizację wybiera najbliższy nieodkryty obszar na mapie. W tym kroku następuje również wymiana informacji dotyczących całego obszaru mapy. W drugim kroku następuje wysłanie informacji o wybranym obszarze do wszystkich robotów biorących udział w zadaniu mapowania. Każdy robot oblicza swoją odległość do nowego obszaru nieznanego, uwzględniając również odległość jaka dzieli go od celu aktualnego zadania. W kroku trzecim informacje zawierające obliczone odległości przesyłane są do wszystkich robotów. Następuje wówczas obliczenie wskaźnika zwanego w algorytmie wartością przetargu:

$$(11)\quad bid_k = p_k + \alpha_k - \sum_{i=0, i\neq k}^{n} (p_i + \alpha_i) \text{ dla } k = 0, ..., n-1$$

gdzie i oraz k reprezentują numery robotów, n ilość robotów, p odległość do końca aktualnego zadania, α odległość do aktualnie branego pod uwagę obszaru nieznanego. W ostatnim kroku następuje wymiana informacji dotycząca wartości obliczonego wskaźnika. Robot, którego wartość wskaźnika jest najniższa przyjmuje nowy obszar za swój cel. Należy także wspomnieć, iż każdy robot może prócz aktualnego zadania posiadać tylko jedno dodatkowe.

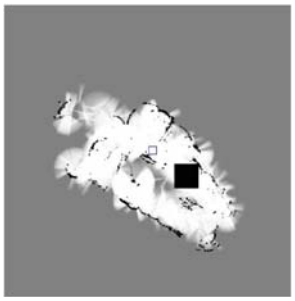
Algorytm ten posiada jedną podstawową wadę, którą jest brak części odpowiedzialnej za ustalanie drogi do wyznaczonego obszaru nieodkrytego. Jego poprawne działanie zaproponowane przez autorów w [7] jest więc niemożliwe. Rozwiązaniem tego problemu jest połączenie algorytmu Self Biddings z algorytmem VFH. Niezbędne okazało się również wprowadzenie zmian w algorytmie określających sposób wymiany informacji pomiędzy robotami. Zaproponowane przez autorów rozwiązanie zakładające samodzielną pracę robotów zostało wzbogacone o jednostkę koordynującą ich pracę. Pozwala to w znacznym stopniu ograniczyć ilość danych wysyłanych za pośrednictwem sieci bezprzewodowej, co w przypadku dużej liczby robotów mogłoby być utrudnione. Proces wyznaczania i przydzielania nowego obszaru, w którym musi zostać przeprowadzony proces mapowania odbywa się w całości w pamięci komputera, a następnie jego wynik zostaje przesłany do konkretnego robota.

Połączenie algorytmów

Połączenie działania algorytmów VFH+ oraz Self Biddings polega na przyjęciu za aktualny cel robota w algorytmie VFH+ kierunku wskazującego obszar przypisany przez algorytm Self Biddings. Rozpoczęcie procesu mapowania otoczenia wymusza na wszystkich połączonych robotach wykonanie pełnego skanowania swojego otoczenia w promieniu 360°. Po otrzymaniu uzyskanych danych za pośrednictwem sieci bezprzewodowej następuje wyznaczenie nowego najbliższego nieokrytego obszaru wokół robota, którego wiadomość jest przetwarzana w pierwszej kolejności. Wyszukiwanie nieznanego obszaru odbywa się przez iteracyjne poszerzanie obszaru zajmowanego przez robota i sprawdzanie wartości na jego krawędziach. W momencie napotkania komórki mapy, której wartość odpowiada obszarowi nieznanemu następuje sprawdzenie wymiarów tego obszaru. Jeżeli powierzchnia zajmowana przez obszar jest większa od minimalnej wartości ustawionej w programie następuje obliczenie wartości przetargu zgodnie z algorytmem Self Biddings oraz przypisanie obszaru do odpowiedniego robota. Poinformowanie robota o nowym kierunku ruchu odbywa się wraz z otrzymaniem wiadomości o jego aktualny położeniu, która wysyłana jest co pewien określony czas. Wiadomość ta wymusza uruchomienie algorytmu VFH+, który na podstawie informacji z mapy otoczenia i położenia nowego zadanego obszaru wyznacza kierunek ruchu robota. Wyznaczenie kierunku wskazującego zadany obszar jest wykonywane w każdym kolejnym kroku algorytmu na nowo ze względu na zmieniającą się pozycję i orientację robota. Podczas wyznaczania nowego kierunku następuje również sprawdzenie czy możliwe jest osiągnięcie zadanego obszaru. Jeżeli droga do obszaru zagrodzona jest przez przeszkodę i nie uda się jej ominąć w kolejnych dziesięciu krokach algorytmu obszar ten zostaje usunięty.

Unikanie kolizji

Wykorzystanie grupy robotów podczas procesu tworzenia map otoczenia pociąga za sobą konieczność utworzenia zabezpieczenia zapobiegającego możliwym kolizją robotów. Zabezpieczenie to może sprowadzać się do właściwego interpretowania informacji z czujników robota i przerwania pracy w przypadku gdy możliwa kolizja zostanie wykryta. To rozwiązanie jednak nie gwarantuje, że zderzenie dwóch robotów poruszających się na niewielkiej przestrzeni nigdy nie nastąpi. W opisywanej pracy zaproponowano rozwiązanie nazywane dalej mapą kolizji. Mapa kolizji to mapa zawierająca wszystkie uzyskane na temat otoczenia informacje wzbogacone o aktualną pozycję robotów biorących udział w procesie mapowania. Tworzenie takiej mapy następuje w każdym kroku algorytmu wyznaczającego bezpieczną ścieżkę poruszania się robota i dzieli się na dwa etapy. W pierwszym następuje utworzenie listy robotów i usunięcie z niej robota, dla którego wyznaczana jest aktualnie bezpieczna trasa. W drugim etapie na podstawie najaktualniejszej znanej pozycji każdego robota z listy na mapę reprezentującą otoczenie nanoszone są obszary o bardzo wysokim prawdopodobieństwie wystąpienia przeszkody. Wielkość każdego obszaru jest znacznie większa niż rzeczywista powierzchnia zajmowana przez pojedynczego robota (rys. 6), co ma na celu zapewnienie utrzymania bezpiecznej odległości pomiędzy poruszającymi się urządzeniami. Na podstawie tak utworzonej mapy odbywa się wyznaczanie bezpiecznego kierunku ruchu robota.

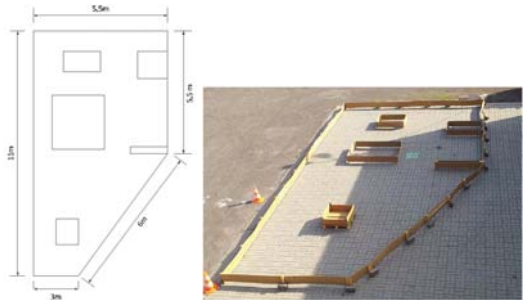


Rys. 6. Przykładowa mapa kolizji utworzona dla robota niebieskiego. Duży ciemny kwadrat na mapie oznacza obszar, w którym aktualnie znajduje się drugi robot.

Przeprowadzone testy

Przeprowadzone testy dotyczą poprawności w odwzorowaniu otoczenia, wpływu czynników zewnętrznych na jakość uzyskiwanych danych, a także poprawności w wyznaczaniu pozycji robotów. Opisano również napotkane problemy. W celu zapewnienia robotom możliwie dużego obszaru, na którym mógłby być przeprowadzony proces mapowania zdecydowano o budowie toru testowego, który został przedstawiony na rysunku 7. Pozwoliło to na różnorodne konfigurowanie obszaru pracy robotów, a także na uniknięcie przeszkód, do wykrywania których roboty nie zostały przystosowane. Mowa tu przede wszystkim o schodach, progach i przeszkodach, których kształt i materiał uniemożliwia wykrycie przez

ultradźwiękowe czujniki odległości. Tor testowy zbudowany został na zewnątrz budynku, co podyktowane było przede wszystkich jego rozmiarami, ale także problemami dotyczącymi poprawnej pracy kompasów cyfrowych w otoczeniu metalu znajdującego się w zbrojeniach budynków.

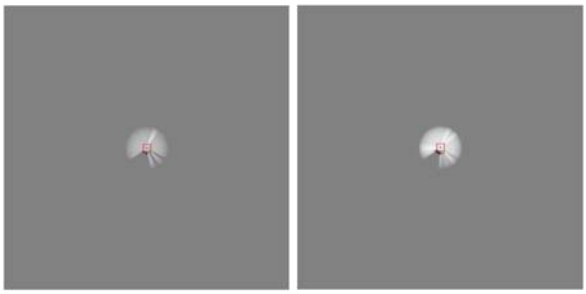


Rys. 7. Widok stanowiska badawczego. Kolorem zielonym oznaczono pole startowe.

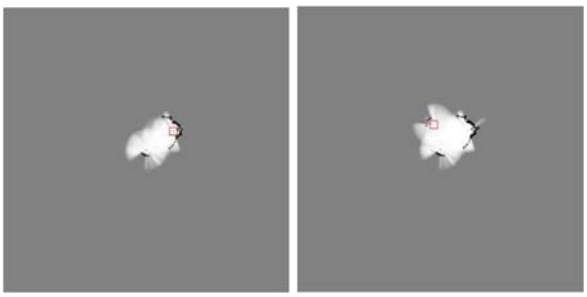
W celu możliwie jak najdokładniejszego porównania działania dwóch zastosowanych algorytmów ich testy przeprowadzone zostały z wykorzystaniem takiej samej konfiguracji toru testowego. Wyznaczono także miejsce startowe, z którego każdy robot rozpoczynał pracę. Takie postępowanie jest konieczne w opisywanym projekcie ze względu na brak systemu wyznaczania pozycji początkowej robotów. Zastosowanie systemu, który pozwoliłby na określenie początkowego położenia robotów w przestrzeni byłoby skomplikowane i znacznie wykraczałoby poza temat pracy.

Test zmodyfikowanego algorytmu VFH+

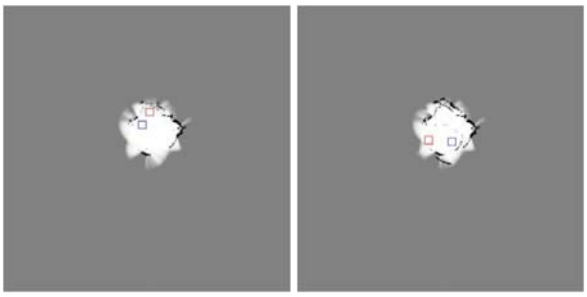
Testy algorytmu VFH+ przeprowadzony został na przygotowanym torze testowym. Zadaniem robotów było wykonanie pełnej mapy całego dostępnego obszaru. W czasie pracy robotów nie dokonywano żadnych zmian w obszarze roboczym. Proces mapowania rozpoczął się z wykorzystaniem jednego robota ustawionego w punkcie startowym. Po wyznaczeniu kierunku ruchu i opuszczeniu przez pierwszego robota pola startowego do procesu mapowania otoczenia wprowadzono robota drugiego. Wyniki mapowania w postaci obrazów zapisywano co 60s i przedstawiono poniżej.



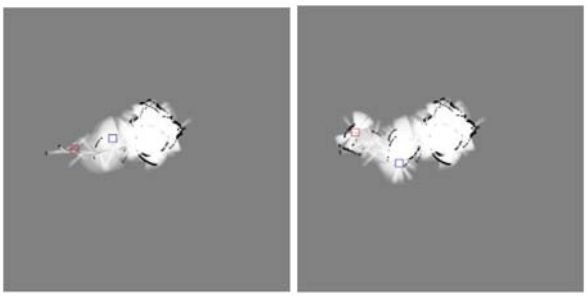
Rys. 8. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 1 min).



Rys. 9. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 3 min).

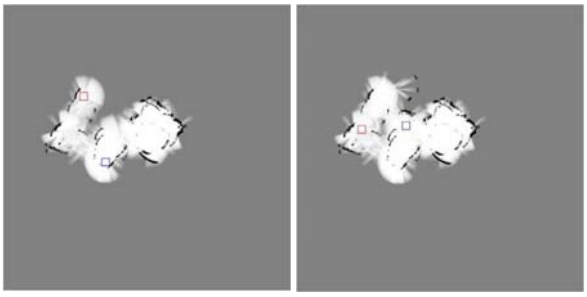


Rys. 10. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 5 min).

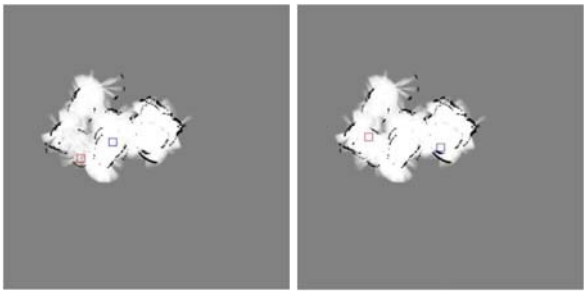


Rys. 11. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 7 min).

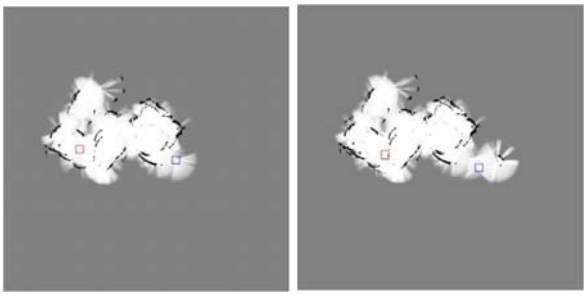
Wynik końcowy procesu tworzenia mapy otoczenia z wykorzystaniem zmodyfikowanego algorytmu VFH+ chociaż nie we wszystkich obszarach zbliżony do rzeczywistej konfiguracji toru testowego można uznać za zadowalający. Charakterystyczne punkty przestrzeni jak i jej ogólny kształt zostały właściwie przedstawione na mapie (rys. 22). Działanie tego algorytmu opiera się jednak na wyznaczaniu nowego kierunku bez uwzględniania położenia obszarów nieznanych, więc proces mapowania dużego otoczenia jest bardzo czasochłonny. Należy też zauważyć, że pomimo czasu mapowania wynoszącego 28 minut pozycja robotów nadal była wyznaczana poprawnie, a pomiary odległość do przeszkód wykonane w momencie rozpoczynania zadania pokrywały się z tymi wykonywanymi pod koniec procesu mapowania. Obszarem najgorzej odwzorowanym na mapie otoczenia jest wąski korytarz pomiędzy dużą przeszkodą, a ścianą zewnętrzną toru testowego. Spowodowane jest to ustawieniami algorytmu VFH+ odpowiadającymi za bezkolizyjne poruszanie się między przeszkodami. Algorytm powiększa przeszkody o obszar niezbędny robotowi do bezpiecznego przejazdu, co w przypadku wąskich przestrzeni może prowadzić do ich całkowitego zamaskowania w trakcie procesu wyszukiwania nowego kierunku.



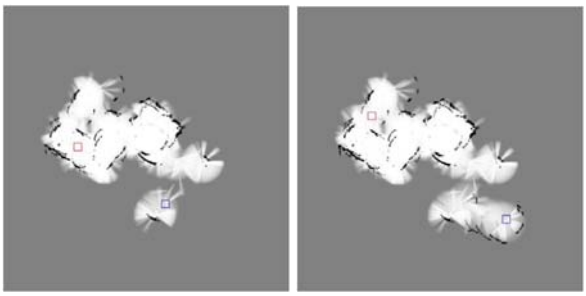
Rys. 12. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 9 min).



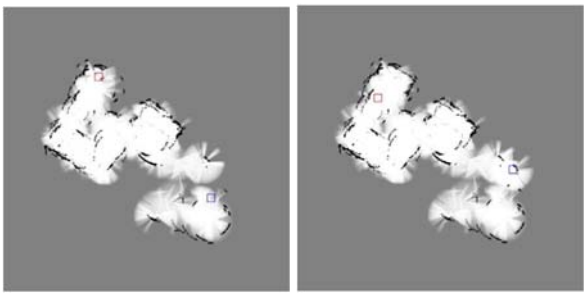
Rys. 13. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 11 min).



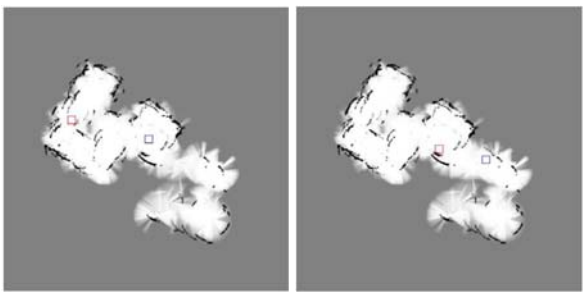
Rys. 14. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 13 min).



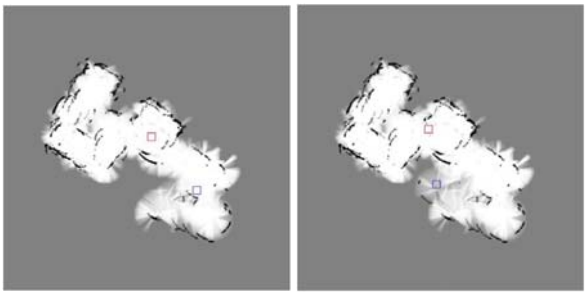
Rys. 15. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 15 min).



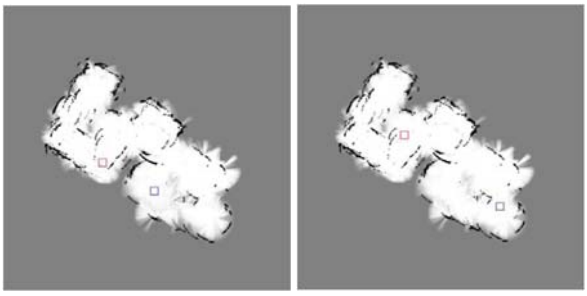
Rys. 16. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 17 min).



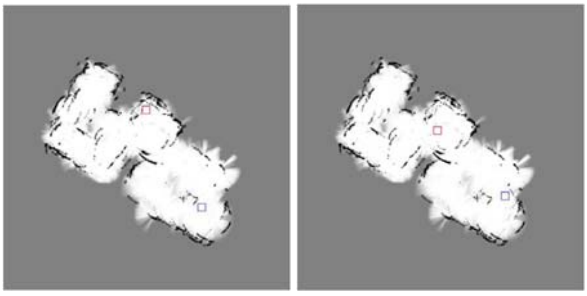
Rys. 17. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 19 min).



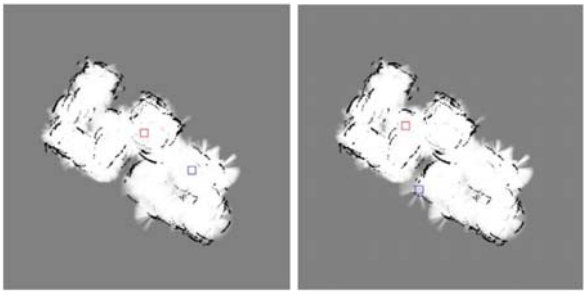
Rys. 18. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 21 min).



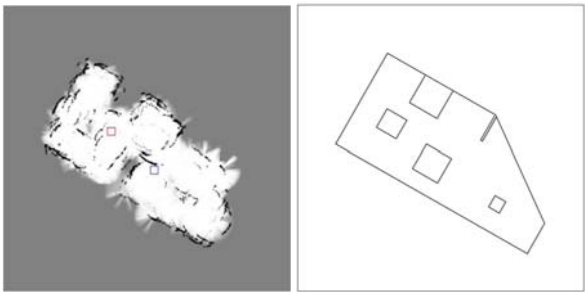
Rys. 19. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 23 min).



Rys. 20. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 25 min).



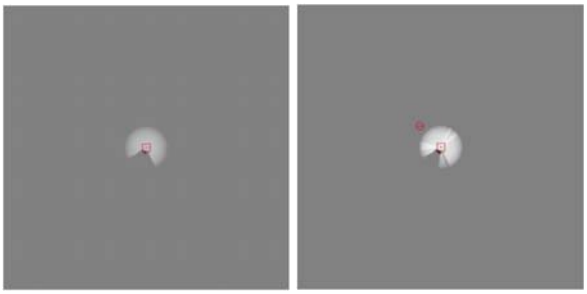
Rys. 21. Proces tworzenia mapy otoczenia z wykorzystaniem algorytmu VFH+ (czas pracy 27 min).



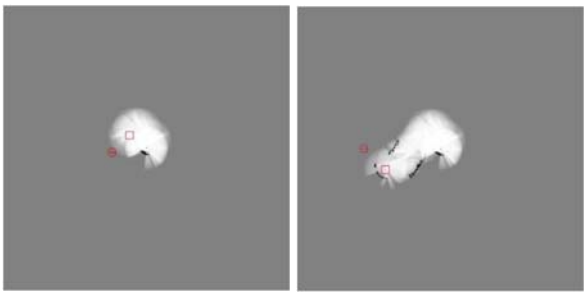
Rys. 22. Porównanie mapy utworzonej z wykorzystaniem zmodyfikowanego algorytmu VFH+ z rzeczywistym kształtem i rozmieszczeniem przeszkód.

Test połączonych algorytmów VFH+ oraz SELF BINDINGS

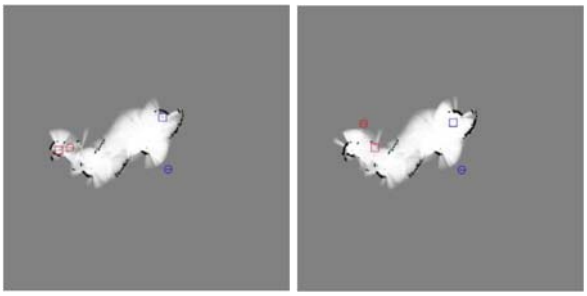
Test połączonych algorytmów VFH+ oraz Self Biddings przeprowadzony został w taki samych warunkach jak test zmodyfikowanego algorytmu VFH+. Zadaniem robotów było stworzenie mapy otoczenia całego dostępnego obszaru. Uzyskane wyniki podobnie jak w poprzednim teście zapisywano w postaci obrazu mapy co 60 sekund.



Rys. 23. Proces tworzenia mapy otoczenia z wykorzystaniem połączonych algorytmów VFH+ i Self Biddings (czas pracy 1 min)

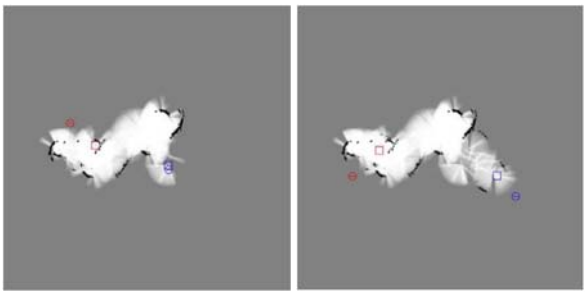


Rys. 24. Proces tworzenia mapy otoczenia z wykorzystaniem połączonych algorytmów VFH+ i Self Biddings (czas pracy 3 min)

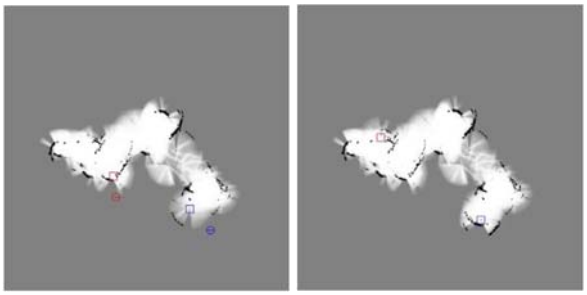


Rys. 25. Proces tworzenia mapy otoczenia z wykorzystaniem połączonych algorytmów VFH+ i Self Biddings (czas pracy 5 min)

W przypadku zastosowania połączonych algorytmów w procesie mapowania otoczenia znacznie skrócił się czas całego procesu. W porównaniu do poprzedniego testu czas ten wyniósł 18 minut, czyli był o około 30% krótszy. Można mieć natomiast kilka zastrzeżeń co do jakości uzyskanej mapy (rys. 32), której dokładność jest niższa niż w teście zmodyfikowanego algorytmu VFH+ (rys. 22). Spowodowane jest znacznie mniejszą ilością pomiarów odległości wykonanych podczas działania połączonych algorytmów oraz podążaniem robota cały czas w kierunku nowego nieznanego obszaru. Proces ten można bardzo dobrze zaobserwować na rysunkach 26 i 27. W przeciwieństwie do poprzedniego testu udało się oddać przybliżony kształt wąskiego korytarza pomiędzy dużą przeszkoda w centralnej części obszaru a jego zewnętrzną ścianą. Robot został niejako zmuszony przez algorytm Self Biddings do podążania w kierunku korytarza przez czas wystarczający do zebrania wystarczającej ilości informacji o jego kształcie.



Rys. 26. Proces tworzenia mapy otoczenia z wykorzystaniem połączonych algorytmów VFH+ i Self Biddings (czas pracy 7 min)



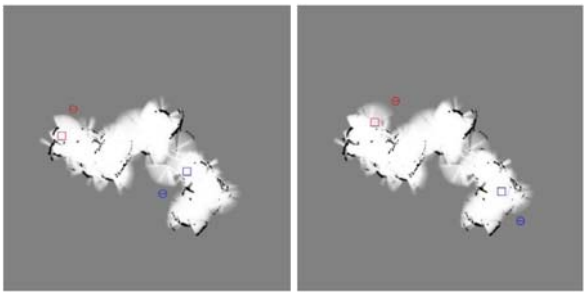
Rys. 27. Proces tworzenia mapy otoczenia z wykorzystaniem połączonych algorytmów VFH+ i Self Biddings (czas pracy 9 min)

Wpływ zasięgu pomiarów na dokładność mapy

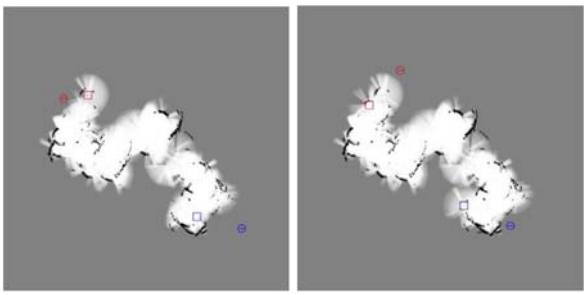
Wykorzystane w projekcie ultradźwiękowe czujniki odległość charakteryzują się maksymalnym zasięgiem wynoszącym 3,5m. Wartość ta może zostać łatwo zmieniona jednak w początkowej fazie testów algorytmów otoczenia wykorzystano domyślne ustawienia czujników. Po kilkukrotnym wykonaniu mapowania obszaru testowego stwierdzono, że bardzo niewielki procent przeszkód znajdujących się w dużej lecz będącej w zasięgu czujnika odległości zostaje wykrytych poprawnie (rys. 33). Częstym zjawiskiem okazało się błędne wprowadzanie na mapę otoczenia obszarów, w których znajdowała się przeszkoda jako obszarów wolnych. Badania oraz liczne testy wykazały, że w zależności od mierzonej odległości wykrywanie przeszkód, których powierzchnia nie jest idealnie prostopadła do czujnika robota jest bardzo utrudnione. Minimalny kąt pod jakim mogła znajdować się płaszczyzna przeszkody w bliskiej odległości od czujnika oszacowano na około 45°. Zwiększanie odległości od czujnika powodowało zwiększanie wartości tego kąta do około 80°przy maksymalnym zasięgu czujnika. Aby uniknąć problemów związanych z niewłaściwym pomiarem odległości do oddalonych przeszkód zdecydowano o zmniejszeniu maksymalnego zasięgu czujników do wartości 1,2m.

Wnioski

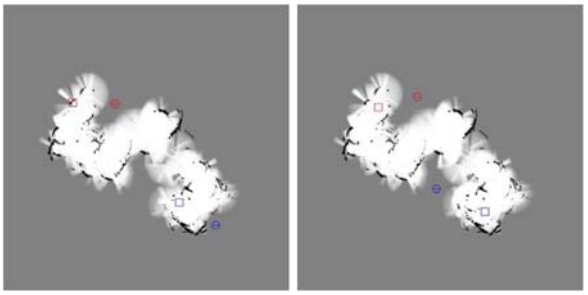
Celem pracy było przetestowanie algorytmów budowy map otoczenia z wykorzystaniem rzeczywistych robotów mobilnych. W tym celu stworzono kompleksowy projekt, w któ-



Rys. 28. Proces tworzenia mapy otoczenia z wykorzystaniem połączonych algorytmów VFH+ i Self Biddings (czas pracy 11 min)

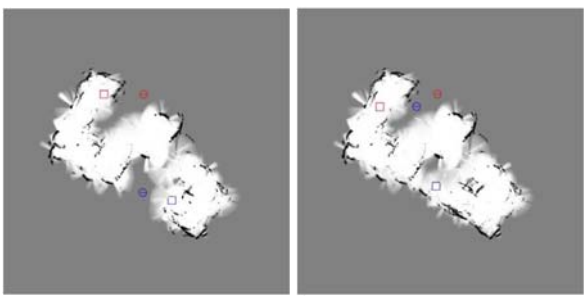


Rys. 29. Proces tworzenia mapy otoczenia z wykorzystaniem połączonych algorytmów VFH+ i Self Biddings (czas pracy 13 min)

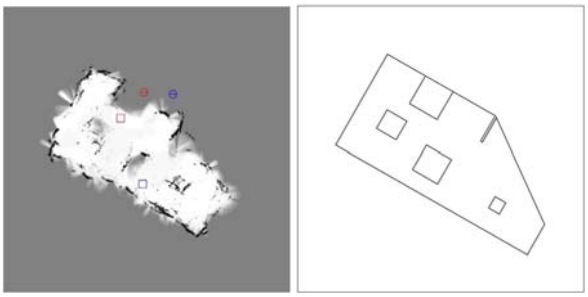


Rys. 30. Proces tworzenia mapy otoczenia z wykorzystaniem połączonych algorytmów VFH+ i Self Biddings (czas pracy 15 min)

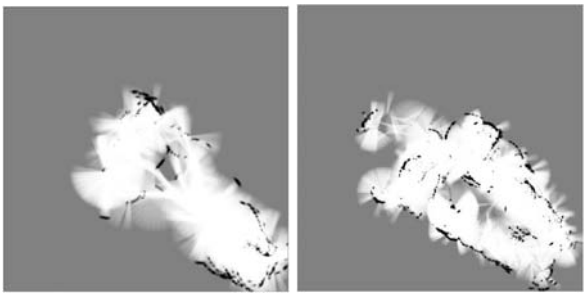
rego skład weszły nie tylko wspomniane algorytmy, ale również projekt, konstrukcja i oprogramowanie robotów mobilnych. Budowa robotów mobilnych nie była niezbędną częścią pracy, a proces tworzenia i testowania algorytmów z powodzeniem mógł być realizowany w środowisku symulacyjnym. Postanowiono jednak podjąć to wyzwaniem kierując się chęcią zweryfikowania proponowanych algorytmów w rzeczywistym środowisku. Część elektroniczna odpowiedzialna za właściwe funkcjonowanie robotów powstała w drugiej kolejności. Podczas jej projektowania zdecydowano między innymi o wykorzystaniu systemu wieloprocesorowego łączącego zalety mikrokontrolerów z rodziny ARM oraz AVR, wybrano również protokół ZigBee jako medium łączące roboty oraz koordynujący ich pracę komputer PC. Zaprojektowano oraz wykonano płytki drukowane umożliwiające zamontowanie układów elektronicznych na platformach robotów. Zdecydowano także o wyborze czujników ultradźwiękowych do wykrywania przeszkód na oraz modułów kompasu cyfrowego w wyznaczania kierunku. Po zmontowaniu robotów przystąpiono do tworzenia oprogramowania sterującego ich pracą. Napisano oprogramowanie na mikrokontroler ATxmega32A4 sterujące pracą napędów robotów, a także na mikrokontroler NXP LPC1754 zarządzające wszystkimi funkcjami robota. Jako ostatni rozpoczęto etap związany z utworzeniem oraz przetestowaniem algorytmów budowy map otoczenia z wykorzystaniem grupy robotów. W tym celu napisano oprogramowanie na komputer PC zdolne do komunikacji z robotami



Rys. 31. Proces tworzenia mapy otoczenia z wykorzystaniem połączonych algorytmów VFH+ i Self Biddings (czas pracy 17 min)



Rys. 32. Porównanie mapy utworzonej z wykorzystaniem połączonych algorytmów VFH+ oraz Self Biddings z rzeczywistym kształtem i rozmieszczeniem przeszkód.



Rys. 33. Wyniki testów przeprowadzonych dla różnych wartości maksymalnego zasięgu czujnika ultradźwiękowego. Po lewej stronie fragment mapy otoczenia utworzony przy zasięgu czujnika wynoszącym 2m, po stronie prawej przy zasięgu 1.2m.

oraz prezentacji wyników mapowania otoczenia.

Problematyka mapowania otoczenia z wykorzystaniem robotów mobilnych jest bardzo rozległa i przedstawiona praca realizuje tylko niewielki fragment tego zagadnienia, wnosząc jednak kilka nie występujących w podobnych projektach modyfikacji dotyczących sposobu przetwarzania i wymiany informacji między robotami, a także funkcjonowaniem algorytmów poza układami robotów. Zarówno projekt robotów jak i algorytmów może podlegać dalszym usprawnieniom w celu przyspieszenia procesu mapowania otoczenia oraz uzyskania dokładniejszej mapy. Do usprawnień, które mogłyby pomóc w uzyskaniu wyników lepszych od zaprezentowanych zaliczyć można przede wszystkim skorzystanie z laserowych czujników zamiast ultradźwiękowych oraz zapewnienie szybszej komunikacji bezprzewodowej pozwalającej na wymianę znacznie większej ilości danych.

LITERATURA

[1] Borenstein J., Koren Y.: Histogramic in-motion mapping for mobile robot obstacle avoidance, IEEE Transaction on Robotics and Automation, vol. 7, no. 4, pp. 535–539, 1991.

[2] Borenstein, J., Koren Y.: The vector field histogram-fast obstacle avoidance for mobile robots, IEEE Transactions on Robotics and Automation, vol. 7, no. 3, pp. 278–288, 1991.

[3] Choset H. and all, Principles of Robot Motion: theory, algorihms and implementations, Englewood Cliffs and New Jersey: MIT Press, 2005.

[4] De Carvalho R.N., Vidal, H.A., Vieira, P., Ribeiro M.I.: Complete coverage path planning and guidance for cleaning robots, Proceedings of the IEEE International Symposium Industrial Electronics ISIE '97, col. 2, pp. 677–682, 1997.

[5] Durrant-Whyte, H., Bailey T.: Simultaneous localization and mapping: part I, IEEE Robotics & Automation Magazine, Vol. 13 no. 2, pp. 99–110, 2006.

[6] Elfes A.: Sonar-based real-world mapping and navigation, IEEE Journal of Robotics and Automation, vol. RA-3, no. 3, pp. 249–265, 1987.

[7] Elizondo-Leal J. C., Ramírez-Torres G., Toscano P. G.: Multi-robot exploration and mapping using self biddings and stop signals, Lecture Notes in Computer Science Springer, MICAI, pp. 615–625, 2008.

[8] Project Tango, [web page] https://www.google.com/atap/projecttango/

[9] 3-Axis Digital Compass IC HMC5883L, Honeywell, [web page] http://honeywell.com
[10] Howard A.: Multi-robot Simultaneous Localization and Mapping using Particle Filters, International Journal of Robotics Research, Vol 25 No 12, pp. 1243–1256, 2006.
[11] Howard A., Parker L. E., Sukhatme G. S.: Experiments with a Large Heterogeneous Mobile Robot Team: Exploration, Mapping, Deployment and Detection International Journal of Robotics Research, Vol , pages , 2006.
[12] Howard A., Mataric´ M. J., Sukhatme G. S.: An Incremental Self-Deployment Algorithm for Mobile Sensor Networks, Autonomous Robots, Special Issue on Intelligent Embedded Systems, Vol 13 No 2, pp. 113–126, 2002.
[13] Kamon I., Rimon E., Rivlin E.: A range-sensor based navigation algorithm, International Journal of Robotics Research, vol. 17, no. 9, pp. 934-–953, 1991.
[14] Knapik K.: Algorytmy budowy map otoczenia z wykorzystaniem grupy robotów mobilnych, Praca Magisterska, Gliwice, 2012.
[15] Microsoft Robotic Developer Studio, [web page] \http://www.microsoft.com/robotics/
[16] Moravec H.,Elfes A.: High resolution maps from wide angle sonar, Proceedings of the IEEE International Conference on Robotics and Automation, pp. 116–121, 1985.
[17] Nowakowski J., Mencwal A., Kośla P.: Rekonstrukcja otoczenia robota mobilnego przy wykorzystaniu systemu stereowizyjnego, Automatyka, Tom 11 z.7, pp. 127–140, 2007.
[18] Ocana M., Bergasa LM., Sotelo MA., Nuevo J., Flores R.: Indoor robot localization system using WiFi signal measure and minimizing calibration effort, IEEE ISIE,vol. 20, pp. 1545–1550, 2005.
[19] Ribeiro M. I.: Obstacle Avoidance, 2005
[20] Sadasivan S.: An Introduction to the ARM Cortex-M3 Processor, [web page] http://www.arm.com/files/pdf/IntroToCortex-M3.pdf, ARM, 2006.
[21] Thrun S.: Robotic Mapping: A Survey, Exploring Artificial Intelligence in the New Millenium, 2002.
[22] Thrun S., Leonard J. J.: Simultaneous Localization and Mapping, Springer Handbook of Robotics, pp. 871–889, 2006.
[23] Ulrich I., Borenstein J.: Vhf+: Reliable obstacle avoidance for fast mobile robots, Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 1572–1577, 1998.
[24] Ulrich I., Borenstein J.: Vhf+: Local obstacle avoidance with look-ahead verification, Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 2505–2511, 2000.

***Autorzy**: dr inż. Krzysztof Jaskot, Politechnika Śląska, Instytut Automatyki, ul. Akademicka 16, 44-100 Gliwice, Email:krzysztof.jaskot@polsl.pl, mgr inż. Krzysztof Knapik, absolwent Instytutu Automatyki Politechniki Śląskiej*