Planowanie przejazdu przez zbiór punktów dla zadania zrobotyzowanej inspekcji

Conference Paper · July 2014

DOI: 10.13140/2.1.1224.2565

2 authors:

Tomasz Gawron
Poznan University of Technology
9 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Maciej Marcin Michałek
Poznan University of Technology
52 PUBLICATIONS **198** CITATIONS
SEE PROFILE

Planowanie przejazdu przez zbiór punktów dla zadania zrobotyzowanej inspekcji*

Tomasz Gawron[1], *Maciej Marcin Michałek*[1]

Streszczenie

Artykuł poświęcony jest dwuetapowemu, globalnemu (tj. planującemu całe zadanie ruchu przy pomocy dostępnych informacji nt. całego środowiska ruchu) algorytmowi planowania bezkolizyjnego przejazdu przez zbiór punktów referencyjnych dla monocykla w środowisku ustrukturyzowanym. Opracowana metoda generuje plany dla sterownika VFO (z ang. *Vector Field Orientation*). Pierwszy etap planowania polega na wyszukaniu zbioru punktów w dwuwymiarowej (pomimo trójwymiarowej przestrzeni problemu) siatce zajętości za pomocą zmodyfikowanego algorytmu A*. Wprowadzone modyfikacje polegają na zastosowaniu: nowej definicji funkcji kosztu, niedopuszczalnej funkcji heurystycznej oraz redukcji sąsiedztwa w grafie. Na podstawie wyszukanego zbioru punktów wyznaczane są pozycje punktów referencyjnych. Drugi etap stanowi planowanie orientacji referencyjnych dla otrzymanych punktów referencyjnych poprzez wykorzystanie własności sterownika VFO. Proponowany algorytm planowania dopuszcza zmiany strategii ruchu (przód/tył). W pracy pokazano jak dobór odpowiednich parametrów metody wpływa na właściwości generowanych planów.

1. Wprowadzenie

Planowanie ruchu dla nieholonomicznych robotów mobilnych w środowisku ustrukturyzowanym jest zagadnieniem, któremu poświęcono w ostatnich latach bardzo dużo uwagi. Systemy opracowane w ostatnich latach planują trajektorie, ścieżki lub zbiory punktów referencyjnych. Istnieje wiele technik globalnego planowania trajektorii lub ścieżek dla robotów mobilnych, które podczas jednego z etapów przetwarzania generują zbiór pozycji punktów referencyjnych. Dużą część z nich stanowią algorytmy wyszukiwania kombinatorycznego, których przykłady znajdują się w [1]. Algorytmy te są w większości rozszerzonymi wersjami algorytmu A* zaprezentowanego w [4], które wzbogacają go o możliwość adaptacji ścieżki do zmian środowiska, możliwość przyspieszonego przeszukiwania map hierarchicznych, etc. Ciekawe wydają się algorytmy oparte o jednoczesne przeszukiwanie przestrzeni pozycji i orientacji lub przeszukiwanie skończonego zbioru prymitywów ruchu (np. [6]). Rozwiązania te charakteryzują się jednak koniecznością wykonania dodatkowego przetwarzania wstępnego. Istnieją również liczne metody planowania oparte o algorytmy ewolucyjne, czy też próbkowanie probabilistyczne (np. [2]). Spotyka się też rozwiązania dedykowane dla zbioru punktów referencyjnych takie jak to prezentowane w [5], czy też metoda opracowana dla środowiska bez ograniczeń podana w [9]. Szczegółowy przegląd literatury

*Praca finansowana przez NCBiR w ramach Programu Badań Stosowanych z grantu PBS1/A3/8/2012, nr projektu 2224K.

[1]Katedra Sterowania i Inżynierii Systemów, Politechnika Poznańska, ul. Piotrowo 3A, 60-965 Poznań
e-mail: tomasz.gawron@doctorate.put.poznan.pl

nt. planowania ruchu można znaleźć w [7].

W [8] udowodniono, że dla problemów o małej wymiarowości techniki systematycznego wyszukiwania są konkurencyjne w stosunku do metod opartych o próbkowanie probabilistyczne. Zasadne wydaje się więc zaprojektowanie dwuetapowego algorytmu planowania korzystającego w pierwszym etapie ze specjalizowanego wyszukiwania kombinatorycznego do wyznaczenia zbioru punktów, które można połączyć łamaną przebiegającą przez wolną przestrzeń w siatce zajętości reprezentującej środowisko. Zbiór wierzchołków tej łamanej uzupełniony o punkty pośrednie, które dodawane są w zależności od odległości między wierzchołkami łamanej, stanowi zbiór pozycji punktów referencyjnych. W odróżnieniu od znanych dotychczas algorytmów wyszukiwania, w prezentowanej metodzie odległość punktów od granic środowiska wpływa na decyzję o ich wyborze. W przeciwieństwie do metod prezentowanych w [1], wpływ ten ma charakter globalny i jest kluczowy dla uzyskania bezpiecznych planów ruchu. Drugi etap planowania polega na analitycznym wyznaczeniu orientacji referencyjnych w poszczególnych pozycjach referencyjnych oraz parametrów sterownika VFO (z ang. *Vector Field Orientation*) tak, aby uzyskać bezkolizyjną wynikową ścieżkę robota przy spełnieniu więzów nieholonomicznych. Wynikiem drugiego etapu jest zbiór pozycji referencyjnych wraz z orientacjami referencyjnymi i parametrami sterownika VFO. Ze względu na drugi etap planowania, proponowane rozwiązanie jest specjalizowane dla systemów ze sterownikiem VFO. Rozdzielenie procesu planowania na dwa etapy ma różne zalety. Pozwoliło ono na redukcję przestrzeni przeszukiwanej kombinatorycznie z trójwymiarowej (przestrzeń pozycji i orientacji) do przestrzeni dwuwymiarowej (przestrzeń pozycji). Proponowane rozwiązanie jest intuicyjne, gdyż działa w sposób podobny do rozumowania ludzkiego. Podobnie jak w pierwszym etapie proponowanego algorytmu, ludzie wyznaczają w pamięci pozycje punktów referencyjnych wybierając korytarze środowiska, którymi będą się poruszać i identyfikując wąskie przejścia. Dopiero po wyznaczeniu pozycji tych punktów, ludzie zastanawiają się jak optymalnie przez nie przejść. Podobną funkcję pełni drugi etap prezentowanego algorytmu, w którym orientacje referencyjne są dobierane tak, aby dopasować się do łamanej wyznaczanej przez zbiór punktów wygenerowanych w poprzednim etapie przy zachowaniu możliwie niskiej krzywizny ruchu robota. Poza tym, wspomniany podział czyni prezentowane rozwiązanie modularnym. Proponowana metoda planowania nie wymaga przetwarzania wstępnego tak jak metody oparte o dekompozycję przestrzeni problemu. W przeciwieństwie do rozwiązań opartych o próbkowanie probabilistyczne, proponowany algorytm działa zawsze w skończonym czasie i jest deterministyczny (takie same wejścia dają takie same wyniki). Proponowane rozwiązanie nie ogranicza się do kompozycji planów ze skończonego zbioru prymitywów ruchu, tak jak ma to miejsce w [6].

2. Postawienie problemu

Przyjmujemy następujące założenia:

Z1. Informacje o środowisku ruchu zawarte są w mapie będącej ograniczoną, zbinaryzowaną, dwuwymiarową siatką zajętości. Mapa jest stała i znana a priori. To założenie wynika ze specyfiki projektu RobREx, którego celem jest budowa platformy mobilnej służącej m.in. do eksploracji budynków po kata-

strofach. Plany budynków są znane, łatwo dostępne i aktualne w przypadku wystąpienia pożaru, skażenia chemicznego lub zagrożenia bombowego.

Z2. Robot, dla którego planowany jest ruch ma kinematykę monocykla:

$$\dot{\boldsymbol{q}}(t) = \begin{bmatrix} 1 & 0 \\ 0 & \cos\theta(t) \\ 0 & \sin\theta(t) \end{bmatrix} \boldsymbol{u}(t), \qquad \boldsymbol{u}(t) \triangleq \begin{bmatrix} \omega(t) \\ v(t) \end{bmatrix}, \tag{1}$$

gdzie t stanowi czas, $\dot{\boldsymbol{q}}(t)$ jest wektorem prędkości konfiguracyjej robota, $\boldsymbol{q}(t) \triangleq [\theta(t) \quad x(t) \quad y(t)]^\top = \left[\theta \quad \boldsymbol{q}^*(t)^\top\right]^\top$ stanowi współrzędne konfiguracyjne robota, $\theta(t)$ jest orientacją robota, $x(t)$ i $y(t)$ są współrzędnymi punktu prowadzenia platformy robota (patrz rys. 1a), $\boldsymbol{u}(t)$ jest wektorem sygnałów sterujących, przy czym $\omega(t)$ i $v(t)$ są prędkościami kątową i postępową platformy robota.

Z3. Zadanie ruchu realizowane będzie przez sterownik VFO w wersji z [9].

Z4. Punkt końcowy $\boldsymbol{q}_{dN}$ jest znany a priori. Powierzchnia zajmowana przez robota jest aproksymowana prostokątem o szerokości a_r i długości b_r (patrz rys. 1a). Rozpatrywany robot może mieć dowolny kształt niewykraczający poza ten prostokąt.

Zadanie przejazdu przez zbiór punktów referencyjnych jest traktowane przez sterownik VFO jako N podzadań sterowania do punktu. Istotą jego działania jest wykorzystanie wektorowego pola zbieżności $\boldsymbol{h}(t)$. Pole to ma charakter prędkości i jest projektowane w taki sposób, że śledzenie tego pola przez prędkość konfiguracyjną robota gwarantuje zbieżność do i-tej pozycji referencyjnej z dokładnością ε_i w chwili t_i (zgodnie z [9]). Pole zbieżności $\boldsymbol{h}_i(t)$ dla i-tego punktu referencyjnego ma postać:

$$\boldsymbol{h_i}(t) = \begin{bmatrix} h_{ai}(t) \\ \boldsymbol{h}_i^*(t) \end{bmatrix} \triangleq \begin{bmatrix} k_a e_{ai}(t) + \dot{\theta}_{ai}(t) \\ h_{xi}(t) \\ h_{yi}(t) \end{bmatrix}, \qquad e_{ai}(t) \triangleq \theta_{ai}(t) - \theta(t), \tag{2}$$

$$\theta_{ai}(t) \triangleq \mathrm{Atan2c}(\sigma_i h_{yi}(t), \sigma_i h_{xi}(t)), \qquad \boldsymbol{h}_i^*(t) \triangleq k_p \boldsymbol{e}_i^*(t) + \boldsymbol{v}_i^*(t), \tag{3}$$

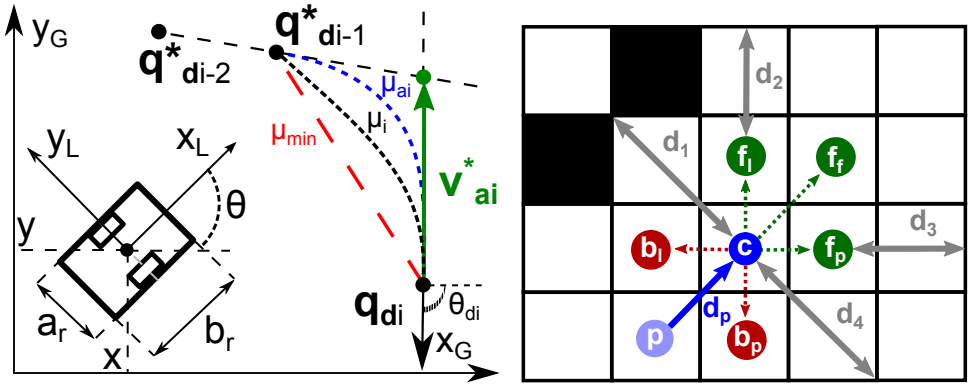
$$\boldsymbol{v}_i^*(t) \triangleq -k_p \mu_i \sigma_i ||\boldsymbol{e}_i^*(t)||, \qquad \boldsymbol{e}_i^*(t) \triangleq \boldsymbol{q}_{di}^* - \boldsymbol{q}^*(t) \tag{4}$$

$$\mu_i = \frac{\eta_i}{k_p}, \tag{5}$$

gdzie $h_{ai}(t)$ ma charakter prędkości kątowej, $k_a > 0$ i $k_p > 0$ są strojonymi przez użytkownika wzmocnieniami, $e_{ai}(t)$ jest pomocniczym uchybem orientacji, θ_{ai} jest orientacją pomocniczą, $\boldsymbol{h}_i^*$ jest polem zbieżności pozycji, $\boldsymbol{e}_i^*(t)$ jest uchybem pozycji, $\boldsymbol{v}_i^*(t)$ jest prędkością wirtualną, $\eta_i \in (0, k_p)$ jest współczynnikiem naprowadzania, $\mu_i \in (0,1)$ jest względnym współczynnikiem naprowadzania, a $\sigma_i \in \{1; -1\}$ jest zmienną decyzyjną określającą strategię ruchu (jazda przodem/tyłem). Prędkość wirtualna $\boldsymbol{v}_i^*(t)$ powoduje efekt naprowadzania, którego intensywność jest zależna od wartości μ_i. Postać sterowania $\boldsymbol{u}_i(t)$ gwarantującą zbieżność do pola $\boldsymbol{h}_i(t)$ pokazano w [9]. Rozpatrywany problem polega na wybraniu uporządkowanych zbiorów:

$$Q \triangleq \{\boldsymbol{q}_{d1}; \boldsymbol{q}_{d2}; \ldots \boldsymbol{q}_{dN}\}, \qquad \Sigma \triangleq \{\sigma_1; \sigma_2; \ldots \sigma_N\}, \qquad M \triangleq \{\mu_1; \mu_2; \ldots \mu_N\}, \tag{6}$$

gdzie $\boldsymbol{q}_{di} = [\theta_{di} \quad x_{di} \quad y_{di}]^\top$ jest wektorem zadanych współrzędnych konfiguracyjnych robota dla i-tego punktu referencyjnego. Zbiory punktów referencyjnych Q, zmiennych decyzyjnych Σ i względnych współczynników naprowadzania M muszą być takie, że przy założeniach Z1-Z4 możliwa jest bezkolizyjna realizacja przejazdu przez zbiór punktów referencyjnych Q poprzez realizację wszystkich punktów z tego zbioru w kolejności, w której w nim występują. Należy zwrócić uwagę na to, że w [9] założono brak ograniczeń środowiska (brak możliwości kolizji), a proponowana metoda zakłada ograniczone środowisko ruchu zgodnie z założeniem Z1. Ścieżkę robota podczas realizacji punktu referencyjnego $\boldsymbol{q}_{di}$ nazywać będziemy i-tym segmentem ruchu. Przez orientację punktu $\boldsymbol{q}_{di}$ rozumieć będziemy orientację zadaną θ_{di} dla robota w tym punkcie.



Rys. 1: a– Model robota oraz geometryczna interpretacja algorytmu planowania współczynnika μ_i, b – Sąsiedztwo w przeszukiwanym grafie oraz kierunki wyznaczania odległości od granic środowiska

3. Strategia planowania

Prezentowane rozwiązanie jest realizacją koncepcji dwuetapowego planowania ruchu. W pierwszym etapie mapa środowiska jest przeszukiwana zmodyfikowanym algorytmem A* z określoną dokładnością. W przypadku znalezienia zbioru punktów, których połączenie tworzy łamaną prowadzącą do punktu końcowego, jej wierzchołki przyjmowane są jako pozycje punktów referencyjnych. Taki wybór jest uzasadniony ze względu na modyfikacje algorytmu A* opisane w podrozdziale 3.1, które powodują że wierzchołki łamanej są oddalone od granic środowiska, a ich pozycje zależą od topologii mapy. Jeśli wierzchołki łamanej są oddalone od siebie o więcej niż wartość parametru projektowego l_s, dodawany jest pomiędzy nimi dodatkowy punkt referencyjny leżący na wyszukanej łamanej. Wynikiem pierwszego etapu jest zbiór pozycji punktów referencyjnych wraz ze zmiennymi decyzyjnymi σ_i odpowiadającymi za strategię ruchu (przód/tył). Zmienne σ_i są znane, gdyż wyszukiwanie odbywa się z uwzględnieniem zmiany strategii ruchu przy wyborze niektórych sąsiadów w grafie (patrz podrozdział 3.1). W drugim etapie planowane są orientacje θ_{di} i względne współczynniki naprowadzania μ_i punktów referencyjnych. Proces ten rozpoczyna

się od od punktu $\boldsymbol{q}^*_{dN}$ i kończy na punkcie $\boldsymbol{q}^*_0$. Po zakończeniu drugiego etapu plan zawiera pozycje punktów referencyjnych, orientacje referencyjne w tych punktach, względne współczynniki naprowadzania μ_i oraz zmienne decyzyjne σ_i. Proces planowania jest globalny, tzn. osiągany jest pełny i dokładny plan wykonywanego zadania, który powstaje w oparciu o informacje na temat całego środowiska ruchu. Poszczególne etapy planowania wyjaśnione są szczegółowo w kolejnych podrozdziałach.

3.1. Etap I: wyszukiwanie kombinatoryczne

Mapa przetwarzana jest jako siatka zajętości (a dokładniej siatka Sukhareva, patrz [8]) z ziarnem ϕ, które zależy od topologii mapy i jest parametrem projektowym. Proces przeszukiwania siatki zajętości odbywa się według schematu postępowania charakterystycznego dla algorytmu A*, w którym komórki mapy należące do wynikowego zbioru punktów wybierane są tak, aby minimalizować funkcję całkowitego kosztu ruchu $f \triangleq g + h$. Modyfikacje algorytmu polegają na wprowadzeniu nowej definicji funkcji kosztu ruchu g oraz funkcji heurystycznej h. Zmodyfikowano również sposób wyznaczania sąsiadów w grafie dla danej komórki mapy. Na rys. 1b przedstawione jest sąsiedztwo dla aktualnie przetwarzanej komórki c przy założeniu, że komórka p jest aktualnym poprzednikiem dla komórki c. Przyjmijmy, że $\boldsymbol{\lambda}(c)$ oznacza wektor pozycji środka komórki w przestrzeni zadania. Sąsiedztwo komórki c zależy od kierunku wektora $\boldsymbol{d}_p \triangleq \boldsymbol{\lambda}(c) - \boldsymbol{\lambda}(p)$, w którym musi odbyć się ruch, aby z komórki p przejść do komórki c. Kierunek ten jest potencjalnym kierunkiem wynikowej ścieżki w grafie pomiędzy punktami p i c. Na potrzeby dalszych rozważań przyjmujemy, że nawrotem w c nazywana będzie zmiana strategii ruchu (jazda przodem/tyłem) przez robota z zatrzymaniem w punkcie odpowiadającym pozycji komórki c. Sąsiedztwo $S \triangleq F \cup B$ składa się ze zbioru $F \triangleq \{f_l; f_f; f_p\}$ najbliższych komórek osiągalnych przez robota bez nawrotu i zbioru $B \triangleq \{b_l; b_p\}$ najbliższych komórek osiągalnych poprzez nawrót w c. Przyjmuje się, że do zbioru F należą komórki osiągalne ruchem w kierunku zgodnym z wektorem $\boldsymbol{d}_p$ lub wektorami powstałymi z obrotu $\boldsymbol{d}_p$ o $\pm\frac{\pi}{4}$. Do zbioru B należą tylko komórki osiągalne ruchem w kierunkach zgodnych z wektorami powstałymi z obrotu $-\boldsymbol{d}_p$ o $\pm\frac{\pi}{4}$. W sąsiedztwie pomija się komórkę p, aby zapobiec cyklicznym nawrotom i powtarzaniu się segmentów łamanej powstałej z połączenia wyszukanych punktów. Pomija się również sąsiadów osiągalnych ruchem w kierunku zgodnym z wektorem $\boldsymbol{d}_p$ obróconym o $\pm\frac{\pi}{2}$. Zgodnie z analizą przeprowadzoną w [3] sąsiedzi leżący na tych kierunkach występują w optymalnych rozwiązaniach tylko wtedy, gdy jeden z najbliższych sąsiadów komórki będącej częścią rozwiązania jest zajęty. Przy prawidłowym doborze ziarna ϕ i zastosowanych definicjach funkcji f, g, h taka sytuacja nie może mieć miejsca. Co więcej, brak możliwości zmiany kierunku o $\frac{\pi}{2}$ powoduje zagęszczenie wierzchołków wyszukanej łamanej w okolicach ciasnych skrętów wynikowej ścieżki robota. Jest to zjawisko pożądane ze względu na krytyczność tych fragmentów ścieżki i przyjęty sterownik VFO realizujący ruch. Pominięcie niektórych sąsiadów przyczynia się również do skrócenia czasu wyszukiwania.

Aby uwzględnić możliwość zaplanowania zmian strategii ruchu (przód/tył) robota, podczas przetwarzania mapy etykietuje się przetwarzane komórki znajdujące się w wyszukanym zbiorze punktów podczas jazdy przodem i jazdy tyłem. Podczas porównywania komórek brana jest pod uwagę ich zmienna decyzyjna $\sigma \in \{1; -1\}$

oznaczająca strategię ruchu (-1 oznacza jazdę tyłem, 1 jazdę przodem). Zmienne σ wierzchołków wyszukanej łamanej stają się wprost zmiennymi σ_i odpowiednich punktów referencyjnych.

W celu generowania planów respektujących narzucone orientacje θ_0 i θ_{dN} wprowadzono następujące rozwiązania. Po pierwsze, dla pozycji początkowej przyjmuje się wektor $\boldsymbol{d}_p$ zgodny z początkowym kierunkiem jazdy robota, co zmienia wybór zbioru sąsiadów S. Po drugie, każdy sąsiad $s \in S$ komórki najbliższej pozycji końcowej $\boldsymbol{q}^*_{dN}$, który nie spełnia warunku zgodności $|\text{Atan2}(s_y, s_x) - \theta_{dN}| \leq \frac{pi}{4}$, gdzie s_x i s_y są współrzędnymi pozycji komórki, traktowany jest jak komórka zajęta. Powoduje to utworzenie sztucznej przeszkody o kształcie stożka w otoczeniu komórki końcowej i skutkuje odrzucaniem rozwiązań prowadzących do komórki końcowej w kierunkach istotnie różniących się od kierunku orientacji zadanej. Badania symulacyjne pokazują, że technika ta jest wydajniejsza niż heurystyczne modyfikacje funkcji f, g, h.

Klasyczny algorytm A* wyszukuje ścieżki w grafie optymalne pod względem długości, które przebiegają często niebezpiecznie blisko granic środowiska. Spotykanym rozwiązaniem tego problemu jest lokalna modyfikacja kosztu ruchu g taka, jak w [1]. Niestety globalne zastosowanie takiej modyfikacji kosztu g skutkuje gwałtownym wzrostem ilości komórek przetwarzanych przez algorytm A*. W prezentowanym algorytmie zastosowano specyficzną modyfikację zarówno funkcji kosztu g, jak i funkcji heurystycznej h, dzięki której wspomniany wyżej niekorzystny efekt modyfikacji funkcji kosztu ruchu został zniwelowany. Zmodyfikowane, przyjęte heurystycznie funkcje dla komórki c dane są zależnościami:

$$h(c) \triangleq g(c)\,[s_f ||\boldsymbol{q}^*_{dN} - \boldsymbol{\lambda}(c)|| - 1], \quad g(c) \triangleq m_c s_c\,[g(p) + ||\boldsymbol{\lambda}(c) - \boldsymbol{\lambda}(p)||], \tag{7}$$

$$s_f \triangleq \sqrt{\frac{\hat{D}}{\min(D)}}, \qquad s_c \triangleq 1 + \frac{k_s}{\phi \exp(\min(D))}, \tag{8}$$

przy czym

$$m_c \triangleq \begin{cases} 0.9 & \text{jeśli wybór c nie zmieni kierunku ruchu,} \\ 1.1 & \text{jeśli wybór c zmieni strategię ruchu,} \\ 1 & \text{w pozostałych przypadkach,} \end{cases} \tag{9}$$

gdzie g jest kosztem ruchu z $\boldsymbol{q}^*_0$ do $\boldsymbol{\lambda}(c)$, a h jest funkcją heurystyczną estymującą koszt ruchu z $\boldsymbol{\lambda}(c)$ do $\boldsymbol{q}^*_{dN}$. Funkcja s_c jest miarą bezpieczeństwa komórki c zależną nieliniowo od minimalnej odległości ze zbioru $D \triangleq \{d_1; d_2; d_3; d_4\}$ odległości robota od granic środowiska. Odległości brane pod uwagę w zbiorze D zależą od kierunku wektora $\boldsymbol{d}_p$, co widać na rys. 1b. $\hat{D}$ oznacza średnią arytmetyczną elementów zbioru D. Współczynnik kierunkowy m_c nieznacznie zwiększa koszt ruchu w przypadku nawrotów i zmniejsza go w przypadku ruchu bez zmiany kierunku. Powoduje to redukcję ilości planowanych zmian strategii ruchu i zapobiega nadmiarowych wierzchołków łamanej. Funkcja g rośnie gwałtownie dla komórek położonych blisko granic środowiska ze względu na funkcję s_c. Wpływ tej funkcji na funkcję g można regulować parametrem projektowym $k_s \geq 1$. Wybór większego k_s skutkuje wyszukiwaniem łamanych bardziej oddalonych od granic środowiska, jednak przyjęcie zbyt dużej wartości tego parametru powoduje zdominowanie kosztu f przez

współczynnik s_c i pogorszenie jakości rozwiązań. Funkcja skalująca s_f jest zależna od wartości średniej $\hat{D}$ odległości robota od granic środowiska. Wzmacnia ona działanie funkcji s_c w szerokich korytarzach mapy. Podana definicja funkcji heurystycznej h skutkuje jej niedopuszczalnością, czyli przeszacowaniem w porównaniu z normą euklidesową. Należy jednak zwrócić uwagę na to, że dla komórek o równych co do elementów zbiorach D, tj. komórek równoodległych od granic środowiska zachodzi: $s_f = 1 \wedge s_c = const \implies f = Cm_c g||\boldsymbol{q}^*_{dN} - \boldsymbol{\lambda}(c)||$, gdzie C jest stałą. W tym wypadku różnica wartości funkcji f pomiędzy bezpośrednio sąsiadującymi komórkami wynika jedynie z odległości między tymi komórkami, odległości od punktu końcowego i współczynnika m_c, podobnie jak w klasycznym algorytmie A*. Aby algorytm z pierwszego etapu był kompletny, musi zachodzić $w_k > 2\left(\frac{1}{2}\phi + o_r\right)$, gdzie $o_r = \sqrt{a_r^2 + b_r^2}$ jest średnicą okręgu opisującego prostokąt zajmowany przez robota, ϕ jest ziarnem przetwarzania mapy, a w_k jest szerokością najwęższego korytarza w przeszukiwanym środowisku. Wynika to z tego, że najwęższy korytarz, który może być bezkolizyjnie przemierzony przez okrąg okalający robota musi być szerszy niż o_r. Ze względu na dyskretyzację mapy, planując pozycję punktu podczas wyszukiwania popełniamy błąd $\pm\frac{1}{2}\phi$, który wpływa na wymaganą szerokość korytarza. Wymagana szerokość w_k jest dodatkowo podwojona, ze względu na to, że wyszukana łamana nie może zawierać kątów prostych, co sprawia że korytarze zawierające kąty proste muszą być szersze, aby pozwolić odpowiednią zmianę kierunku w dwóch iteracjach.

3.2. Etap II: planowanie orientacji i współczynnika naprowadzania

Zadane orientacje θ_{di} punktów referencyjnych są planowana zgodnie z algorytmem podanym w [9]. Zaplanowane wcześniej pozycje punktów referencyjnych dzielą przejazd na N segmentów ruchu. Załóżmy, że robot idealnie odtwarza orientację wyznaczoną przez pole zbieżności, tj. $\forall t \geq 0 \quad \theta(t) = \theta_a(t)$, wtedy aby zachować ciągłość orientacji pomocniczej dla punktu $\boldsymbol{q}_{i-1}$ musi zachodzić $\theta_{di-1} = \theta_{ai}(t_{i-1})$. Względny współczynnik naprowadzania μ_i wpływa na kształt ścieżki, wzdłuż której robot osiągnie punkt referencyjny $\boldsymbol{q}_{di}$. Jego wpływ na ścieżkę robota pokazano na rys. 1a (patrz [9]). Współczynnik $\mu_i \in (0,1)$ powinien być wybrany tak, aby wynikowa ścieżka robota była dobrze dopasowana do łamanej wyszukanej w pierwszym etapie planowania. Jednocześnie wynikowa ścieżka robota powinna być gładka, tj. charakteryzować się możliwie niską wartością $\dot{\theta}$ na całej swej długości. Zgodnie z rys. 1a, przyjęcie niskiego współczynnika μ_i skutkuje ścieżką o bardzo dobrym (idealnym dla $\mu_i \to 0$) dopasowaniu do wyszukanej łamanej. Z drugiej strony, przyjęcie μ_i dającego gładką ścieżkę robota w danym segmencie ruchu powoduje, że odbiega ona od wyszukanej łamanej. Kryteria te są przeciwstawne, więc wyznacza się μ_i jako średnią ważoną:

$$\mu_i \triangleq \frac{k_f||\boldsymbol{q}^*_i - \boldsymbol{q}^*_{di-1}||\mu_{min} + ||\boldsymbol{q}^*_{di-1} - \boldsymbol{q}^*_{di-2}||\mu_{ai}}{k_f||\boldsymbol{q}^*_i - \boldsymbol{q}^*_{di-1}|| + ||\boldsymbol{q}^*_{di-1} - \boldsymbol{q}^*_{di-2}||}, \tag{10}$$

$$\mu_{ai} \triangleq \mu_{min} \quad \text{dla} \quad \hat{\mu}_{ai} < \mu_{min}, \quad \mu_{max} \quad \text{dla} \quad \hat{\mu}_{ai} > \mu_{max}, \quad \hat{\mu}_{ai} \quad \text{dla innych}, \tag{11}$$

$$\hat{\mu}_{ai} = \frac{x^L_{di-1} - y^L_{di-1}\,\mathrm{ctg}\left(\mathrm{Atan2}\left(\boldsymbol{q}^*_{di-1} - \boldsymbol{q}^*_{i-2}\right)\right)}{||\begin{bmatrix} x^L_{di-1} & y^L_{di-1} \end{bmatrix}^\top||}, \tag{12}$$

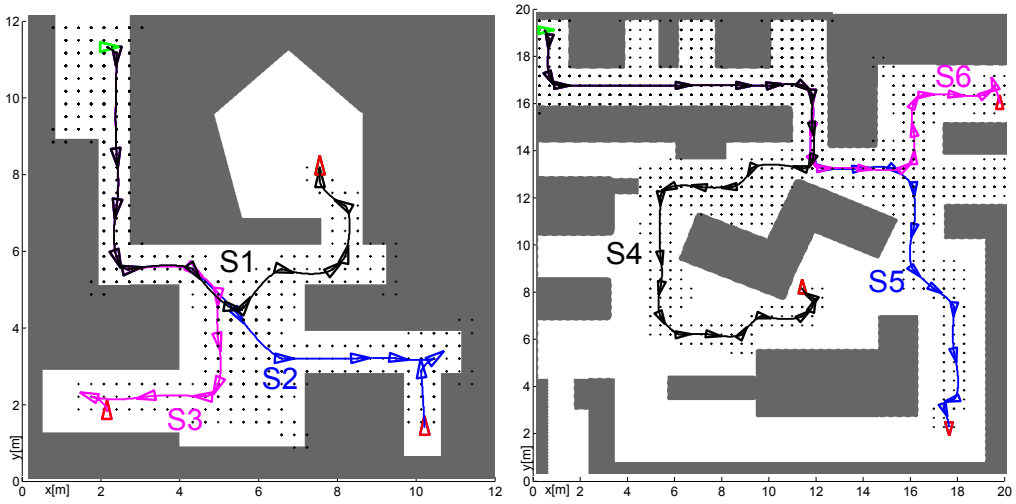
gdzie $k_f \in (0,\infty)$ jest parametrem projektowym określającym priorytet dopasowania wynikowej ścieżki robota do łamanej wyszukanej w pierwszym etapie planowania, $\hat{\mu}_{ai}$ jest współczynnikiem naprowadzania zapewniającym najlepsze dopasowanie ścieżki robota do poprzedniego segmentu wyszukanej łamanej, μ_{ai} oznacza wartość $\hat{\mu}_{ai}$ ograniczoną do przedziału (μ_{min},μ_{max}), a x^L_{di-1} i y^L_{di-1} są współrzędnymi pozycji punktu $\boldsymbol{q}_{di-1}$ w układzie lokalnym związanym z punktem $\boldsymbol{q}_{di}$. Wartości ograniczeń $\mu_{min} \in (0,1)$ i $\mu_{max} \in (\mu_{min},1)$ są parametrami projektowymi. Wybór niskiej wartości k_f skutkuje chwilowo znacznie odbiegającymi od wyszukanej łamanej (potencjalnie kolizyjnymi) ścieżkami robota.

4. Wyniki badań symulacyjnych

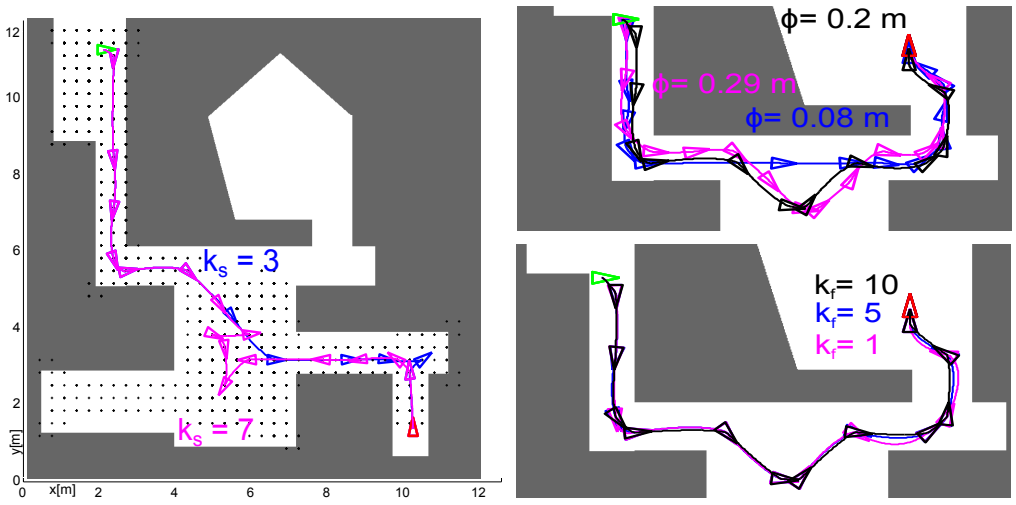
Przeprowadzono szereg symulacji weryfikujących skuteczność proponowanej metody w różnych scenariuszach. Poniżej przedstawiono wyniki wybranych scenariuszy symulacyjnych S1-S6. Przyjęto następującą konwencję oznaczeń na rysunkach: kolorowe trójkąty oznaczają konfiguracje robota punktach referencyjnych, zielony trójkąt oznacza warunek początkowy $\boldsymbol{q}_0$, czerwony trójkąt oznacza punkt $\boldsymbol{q}_{dN}$, czarne znaki "'+"' oznaczają komórki mapy sprawdzone w pierwszym etapie, a kolorowe krzywe odpowiadają wynikowej ścieżce robota uzyskanej poprzez realizację planu z wykorzystaniem sterownika VFO z [9]. Dla wszystkich scenariuszy przyjęto parametry: $k_s = 1$, $k_f = 5$, $k_p = 5$, $k_a = 10$, $a_r = 0.2$ m, $b_r = 0.3$ m i $\varepsilon_i = 0.001$ m, chyba że zaznaczono inaczej. Dla scenariuszy S1-S3 przyjęto ziarno $\phi = 0.29$ m, natomiast dla scenariuszy S4-S6 przyjęto ziarno $\phi = 0.4$ m. Rys. 2 ilustruje skuteczność prezentowanego algorytmu planowania dla sztucznych środowisk ustrukturyzowanych. Prezentowany algorytm przetwarza znacząco mniej komórek niż algorytm Dijkstry. Scenariusz S4 z rys. 2b pokazuje globalność algorytmu planowania, która objawia się tym, że algorytm zaplanował ścieżkę dłuższą niż najkrótsza ścieżka równoodległa od przeszkód, lecz prowadzącą przez szersze korytarze mapy. Rys. 3a pokazuje wpływ współczynnika k_s na planowanie punktów nawrotu robota. Przy przyjęciu dostatecznie dużego współczynnika k_s, algorytm wybiera plan z większą ilością zmian strategii ruchu, skutkujący wcześniejszą zmianą strategii ruchu robota w szerokim korytarzu mapy. Górna część rys. 3b ilustruje wpływ zmniejszania ziarna ϕ na generowane plany. Wygenerowane plany zmieniają się wraz ze zmianą ziarna, lecz nadal pozostają bezpieczne. Zmiany występują głównie w szerokich korytarzach środowiska. Na rys. 3b w części dolnej można zaobserwować dokładność odtwarzania łamanej wyszukanej w pierwszym etapie w zależności od parametru k_f. Zwiększanie tego współczynnika zmniejsza gładkość ścieżki i powoduje lepsze odtwarzanie wyszukanej łamanej.

5. Podsumowanie

W artykule przedstawiono globalny, dwuetapowy algorytm planowania przejazdu przez zbiór punktów referencyjnych dedykowany do współpracy ze sterownikiem VFO. Pozwala on na planowanie zmian strategii ruchu i generuje plany o wysokim stopniu bezpieczeństwa. Właściwości wygenerowanego planu mogą być zmieniane w intuicyjny



Rys. 2: a– Scenariusze symulacyjne S1, S2 i S3, b– Scenariusze symulacyjne S4, S5 i S6



Rys. 3: a– Scenariusz symulacyjny S2 dla $k_s = 3$ i $k_s = 7$, b– Fragment scenariusza S1 dla różnych k_f oraz ϕ

sposób przy pomocy parametrów projektowych. Ze względu na własności algorytmu A* przy braku ograniczeń na funkcję heurystyczną, pierwszy etap planowania ma wykładniczą złożoność obliczeniową względem długości wyszukanej ścieżki w grafie dla przypadku pesymistycznego. Drugi etap planowania ma liniową złożoność obliczeniową względem ilości wierzchołków ścieżki wyszukanej w pierwszym etapie. Wydaje się, że zastosowanie prezentowanej metody planowania do mobilnych robotów ratowniczo-eksploracyjnych przyniesie dobre wyniki. Autorzy pracują nad dowodem na bezkolizyjność ścieżek robota przy określonych wartościach k_f.

LITERATURA

[1] M. Cikes, M. Dakulovic, I. Petrovic. The path planning algorithms for a mobile robot based on the occupancy grid map of the environment - A comparative study. In: Proc. XXIII International Symposium on Information, Communication and Automation Technologies (ICAT). *Proceedings*, 2011, s. 1–8.

[2] Zhou Feng. A novel path planning for robots based on rapidlyexploring random tree and particle swarm optimizer algorithm. IAES International Journal of Robotics and Automation (IJRA), 2014, wolumen 3, numer 2.

[3] Daniel Harabor, Alban Grastien. Online graph pruning for pathfinding on grid maps. In: 25th Conference on Artificial Intelligence (AAAI-11). *Proceedings*, 2011.

[4] P.E. Hart, N.J. Nilsson, B. Raphael. A formal basis for the heuristic determination of minimum cost paths. Systems Science and Cybernetics, IEEE Transactions on, 1968, wolumen 4, numer 2, s. 100–107.

[5] N. Jouandeau, Zhi Yan. Decentralized waypoint-based multi-robot coordination. In: Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2012 IEEE International Conference on. *Proceedings*, 2012, s. 175–178.

[6] R.A. Knepper, A. Kelly. High performance state lattice planning using heuristic look-up tables. In: Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on. *Proceedings*, 2006, s. 3375–3380.

[7] S.M. LaValle. *Planning Algorithms*. Cambridge University Press 2006.

[8] StevenM. LaValle, MichaelS. Branicky. On the relationship between classical grid search and probabilistic roadmaps. In: *Algorithmic Foundations of Robotics V* Red. Jean-Daniel Boissonnat et al, wolumen 7 serii *Springer Tracts in Advanced Robotics*, s. 59–76. Springer Berlin Heidelberg 2004.

[9] Maciej Michałek, Krzysztof Kozłowski. Motion planning and feedback control for a unicycle in a way point following task: The VFO approach. Int. J. Appl. Math. Comput. Sci., 2009, wolumen 19, numer 4, s. 533–545.

Planning the waypoing-following task for robotized inspection

The paper presents two-stage, global motion planning algorithm for the waypoint-following task realized by a unicycle in a structured environment. During the first stage, a modified version of A* pathfinding algorithm is used to find a safe path in two-dimensional occupancy grid. During the second stage, waypoint orientations are planned. Orientation planning exploits properties of the VFO controller used for subsequent motion realization. Proposed two-stage algorithm allows robot reversals and has lower computational complexity than straightforward full task space search. Properties of plans can be modified using tunable parameters.