Laboratorium Podstaw Robotyki

Politechnika Poznańska
Katedra Sterowania i Inżynierii Systemów

ĆWICZENIE 4

System sterowania robotem mobilnym MTracker[1]

Celem ćwiczenia jest poznanie sposobów sterowania nieholonomicznym, dwukołowym robotem mobilnym MTracker. Podczas ćwiczenia poruszane będą zagadnienia związane z planowaniem trajektorii w przestrzeni zadania oraz jej realizacją w torze otwartym.

1 Model kinematyczny robota MTracker

Autonomiczny pojazd kołowy posiadający zdolność zautomatyzowanego ruchu związanego z realizacją postawionego mu zadania nazywany jest kołowym robotem mobilnym. Realizacja zadania związana jest ścisle z przestrzenią, w której zadanie to zostało zdefiniowane. W przypadku kołowych robotów mobilnych naturalną przestrzenią zadania jest trójwymiarowa podprzestrzeń kartezjańska, w której robot się porusza. W tym ćwiczeniu rozważania zostaną ograniczone do ruchu robota na płaszczyźnie.

Do planowania i realizacji trajektorii w przestrzeni zadania niezbędna jest znajomość modelu kinematyki rozważanego robota mobilnego. Uwaga nasza zostanie skupiona na modelu dwukołowego robota mobilnego MTracker przedstawionego na Rys. 1. Schemat kinematyczny robota MTracker przedstawia Rys. 2. Na rysunku tym zaznaczono lokalny $\{x_L, y_L\}$ oraz globalny $\{x_G, y_G\}$ układ współrzędnych. Środek układu lokalnego przywiązano na osi kół i w połowie odległości pomiędzy kołami. Pozycję oraz orientację platformy wyrażoną w układzie globalnym oznaczono odpowiednio jako x, y oraz φ (zwrot strzałki oznacza dodatni przyrost kąta orientacji). Zaznaczono także prędkości kątowe kół: prawego ω_P i lewego ω_L oraz prędkości związane z całą platformą: prędkość postępową v początku układu lokalnego (znak + oznacza dodatni zwrot prędkości) oraz prędkość kątową ω (znak + oznacza dodatni kierunek obrotu platformy).

Przy założeniu braku poślizgu kół (zarówno poślizgu wzdłużnego jak i poprzecznego), zależności między prędkościami liniową v i kątową ω platformy robota a prędkościami kątowymi koła lewego ω_l i prawego ω_p są następujące [1]:

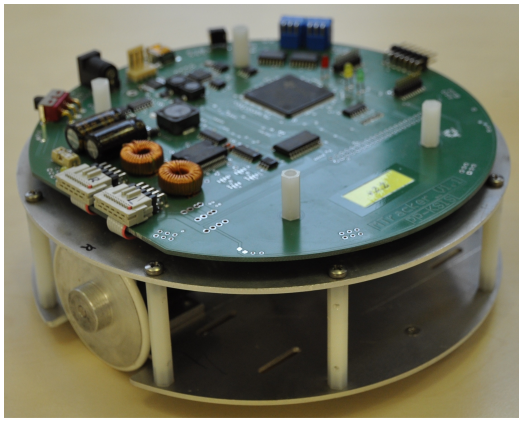
$$v = \frac{\omega_p + \omega_l}{2} R, \qquad \omega = \frac{\omega_p - \omega_l}{b} R, \tag{1}$$

gdzie R jest promieniem kół, a b jest rozstawem między punktami styku kół z podłożem (patrz rys. 2). Przekształcając równania (1) otrzymuje się wyrażenia opisujące prędkości kół robota jako funkcje prędkości liniowej v i kątowej ω całej platformy:

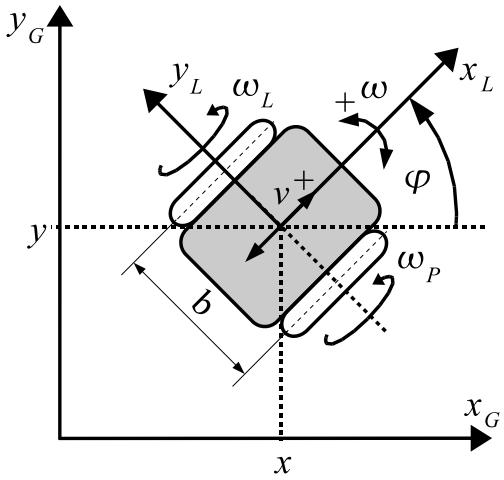
$$\omega_p(v, \omega) = \frac{v + \frac{b}{2}\omega}{R}, \qquad \omega_l(v, \omega) = \frac{v - \frac{b}{2}\omega}{R}. \tag{2}$$

Zależności funkcyjne $\omega_p = \omega_p(v, \omega)$ oraz $\omega_l = \omega_l(v, \omega)$ mają kluczowe znaczenie z punktu widzenia realizacji trajektorii zdefiniowanej w przestrzeni zadania. Pozwalają one na przeliczenie prędkości

[1] Wersja 13.03.2018



Rys. 1: Dwukołowy robot mobilny MTracker.



Rys. 2: Schemat kinematyczny dwukołowego robota mobilnego w globalnym układzie współrzędnych.

platformy do przestrzeni prędkości napędów, które ostatecznie mogą zostać zrealizowane przez pokładowy system sterowania.

Model kinematyki w przestrzeni zadania dla dwukołowej platformy mobilnej z rysunku 2 uwzględniający założenie o braku poślizgu kół względem podłoża ma następującą postać [3]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega + \begin{bmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{bmatrix} v, \tag{3}$$

gdzie $\boldsymbol{q} \triangleq [x\ y\ \varphi]^T$ jest wektorem stanu platformy moblinej, a $\boldsymbol{u} \triangleq [v\ \omega]^T$ jest wektorem sterowań w przestrzeni zadania.

Korzystając z Rys. 2 można zauważyć kilka użytecznych relacji wiążących poszczególne sygnały związane z kinematyką platformy robota [2]:

$$\left.\begin{array}{rcl} \dot{x} \equiv v_x & = & v\cos\varphi \\ \dot{y} \equiv v_y & = & v\sin\varphi \end{array}\right\} \quad \Rightarrow \quad |v| = \sqrt{v_x^2 + v_y^2}, \tag{4}$$

$$\tan\varphi = \frac{\dot{y}}{\dot{x}} \quad \Rightarrow \quad \varphi = \mathrm{Atan2}\left(sgn(v)\dot{y}, sgn(v)\dot{x}\right), \tag{5}$$

$$\omega = \frac{d\varphi}{dt} \quad \Rightarrow \quad \varphi(t) = \varphi(0) + \int_0^t \omega(\tau)d\tau, \tag{6}$$

przy czym $\mathrm{Atan2}\left(\cdot,\cdot\right)$ jest czterocwiartkową funkcją arctan.

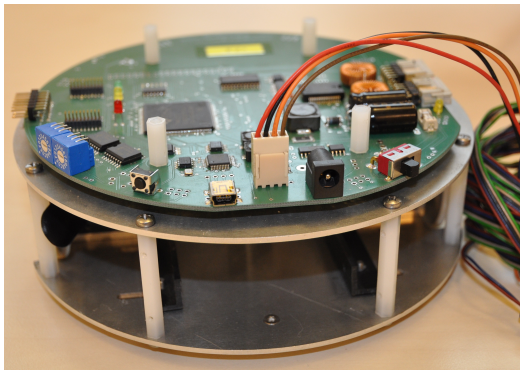
Parametry geometryczne robota MTracker występujące w równaniach wiążących prędkości platformy z prędkościami napędów są następujące:

$$R = 0.025[m], \qquad b = 0.145[m]. \tag{7}$$

2 Środowisko programistyczne i sterowanie robotem MTracker

Komunikacja nadrzędnego układu sterowania (PC) z systemem pokładowym sterownika robota zrealizowana jest poprzez port szeregowy. Poprawny sposób podłączenia robota zaprezentowany jest na Rys. 3. Druga strona widocznego na rysunku przewodu płaskiego powinna zostać podłączona do portu szeregowego komputera oraz do zasilania.

Programowanie robota odbywa się w środowisku MATLAB. Przepływ danych realizowany jest w trakcie symulacji przez funkcję *MTracker*, która odpowiada także za prezentację wyznaczonej trajektorii zadanej oraz rzeczywistej realizowanej przez pojazd. Wewnątrz funkcji *MTracker* inicjalizowane są również parametry symulacji: numer portu COM, czas próbkowania o domyślnej wartości $T_s = 0.04\,\mathrm{s}$ oraz czas symulacji T_f. Funkcja *TrajectoryGenerator* powinna otrzymywać na wejściu aktualny czas symulacji, natomiast na wyjściu zwracać kolejno wektor konfiguracji referencyjnej robota $\boldsymbol{q}_d = [x_d\ y_d\ \theta_d]^\top$ oraz wektor referencyjnych sygnałów sterujących $\boldsymbol{u}_d = [v_d\ \omega_d]^\top$. Funkcja *Controller* w przypadku tego ćwiczenia ma za zadanie przesyłać wartości obliczonych wcześniej referencyjnych sygnałów sterujących u_d jako rzeczywisty sygnał sterujący u. Funkcje *ComputeWheelsVelocities* oraz *ComputeRobotVelocities* powinny wykonywać odpowiednie transformacje między prędkością liniową v i obrotową ω platformy robota oraz prędkościami obrotowymi kół robota ω_p, ω_l.

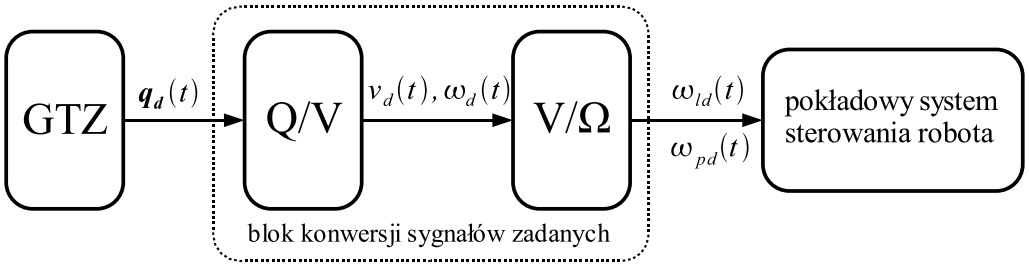


Rys. 3: Dwukołowy robot mobilny MTracker .

3 Generator trajektorii zadanej

Zadaniem generatora trajektorii jest obliczanie chwilowych wartości sygnałów pozycji $x_d(t), y_d(t)$ i orientacji $\varphi_d(t)$ platformy, które mają zostać zrealizowane w układzie: *system sterowania robota* $\leftrightarrow$ *robot*. Ze względu na konstrukcję robota oraz strukturę pokładowego systemu sterowania, realizacja trajektorii zadanej wymaga dodatkowo obliczania chwilowych wartości zadanych prędkości platformy $v_d(t), \omega_d(t)$ (wynikających bezpośrednio z zadanej trajektorii) i przeliczania ich na chwilowe prędkości poszczególnych kół robota $\omega_{pd}(t), \omega_{ld}(t)$. Wartości zadane prędkości robota są odpowiednio przekształcane na jednostce pokładowej robota w celu otrzymania zadanych wartości prędkości kół ω_{dp}, ω_{dl} realizowanych przy pomocy pokładowych prędkościowych pętli regulacyjnych typu PI ze sprzężeniem wyprzedzającym.

Aby prześledzić sposób generowania sygnałów zadanych, rozważmy równania generatora trajektorii zadanej $\boldsymbol{q}_d(t) \triangleq [\varphi_d(t)\ x_d(t)\ y_d(t)]^T$ w przestrzeni zadania na przykładzie trajektorii o kształcie zbliżonym do sinusoidy kreślonej na płaszczyźnie (x, y). Niech zadane współrzędne pozycji plat-



Rys. 4: Schemat ilustrujący sposób przygotowania sygnałów zadanych do realizacji w systemie sterownia robotem MTracker . Przyjęto oznaczenia: GTZ – generator trajektorii zadanej, $\boldsymbol{q}_d(t) = [\varphi_d(t)\ x_d(t)\ y_d(t)]^T$, Q/V – blok konwersji *trajektoria/prędkości platformy*, V/Ω – blok konwersji *prędkości platformy/prędkości kół robota.*

formy opisane będą następującymi równaniami:

$$\begin{array}{rcl} x_d(t) & \triangleq & At, \\ y_d(t) & \triangleq & B\sin(2\pi f_d t), \end{array} \tag{8}$$

gdzie $A, B > 0$ są parametrami projektowymi, a f_d jest żądaną częstotliwością zadanej sinusoidy. Aby określić przebieg zadanej orientacji platformy robota $\varphi_d(t)$ wzdłuż trajektorii pozycji (8), należy określić przebiegi składowych prędkości $v_{xd}(t)$ oraz $v_{yd}(t)$ wzdłuż tej trajektorii. Różniczkowanie równań (8) względem czasu pozwala zapisać:

$$\begin{array}{rcccl} \dot{x}_d(t) & = & v_{xd}(t) & = & A\ =\ const, \\ \dot{y}_d(t) & = & v_{yd}(t) & = & 2\pi f_d B\cos(2\pi f_d t). \end{array} \tag{9}$$

Na podstawie powyższych składowych prędkości można wyznaczyć przebieg orientacji zadanej $\varphi_d(t)$ zgodnie z zależnością (5):

$$\varphi_d(t) = \mathrm{Atan2}\left(sgn(v_d)\dot{y}_d, sgn(v_d)\dot{x}_d\right)\ \in [-\pi, \pi), \tag{10}$$

gdzie

$$v_d(t) = \pm\sqrt{v_{xd}^2(t) + v_{yd}^2(t)}, \tag{11}$$

a znak prędkości $v_d(t)$ determinuje wybór strategii ruchu robota (*ruch przodem/ruch tyłem*), którą określa projektant trajektorii. Zadana prędkość kątowa platformy wynika z różniczkowania równania (10):

$$\omega_d(t) \ = \ \frac{a_{yd}(t)v_{xd}(t) - a_{xd}(t)v_{yd}(t)}{v_d^2(t)}, \tag{12}$$

gdzie $a_{xd}(t)$ oraz $a_{yd}(t)$ są składowymi sygnałami przyspieszenia zadanego i wynikają z różniczkowania prędkości (9):

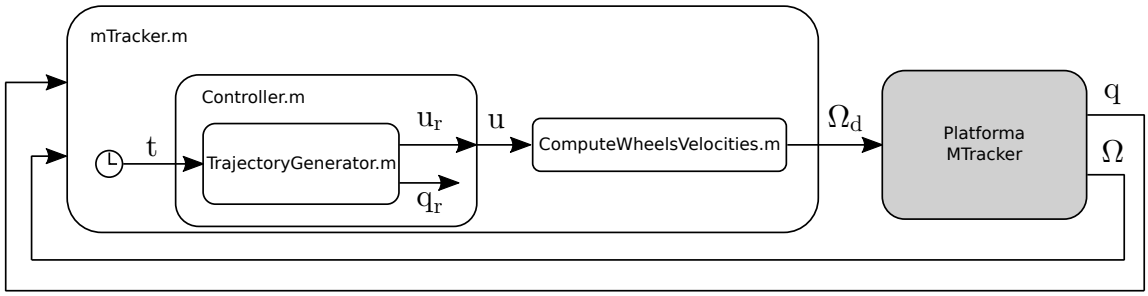
$$\begin{array}{rcl} a_{xd}(t) & \equiv & 0, \\ a_{yd}(t) & = & -(2\pi f_d)^2 B\sin(2\pi f_d t). \end{array} \tag{13}$$

Korzystając z faktu, że zachodzi $\omega_d(t) = \frac{d\varphi_d(t)}{dt}$, przebieg orientacji zadanej możemy także obliczać następująco:

$$\varphi_d(t) = \varphi_d(0) + \int_0^t \omega_d(\tau)d\tau\ \in \mathbb{R}, \tag{14}$$

gdzie $\omega_d(t)$ wynika z (12). Należy zauważyć istotną różnicę między definicjami (10) oraz (14) wynikającą z dziedzin, do których należy sygnał $\varphi_d(t)$.

Na podstawie wyznaczonych zadanych prędkości: liniowej $v_d(t)$ i kątowej $\omega_d(t)$ platformy i korzystając z zależności (2), można teraz określić funkcje zadanych prędkości $\omega_{pd}(t)$ i $\omega_{ld}(t)$ dla poszczególnych kół robota, które mogą zostać fizycznie zrealizowane przez pokładowy system sterowania robota.



Rys. 5: Schemat blokowy wywoływanych funkcji w środowisku Matlab

Table 1: Spis oznaczeń z Rys. 5

Nazwa	Opis
t	Czas
$\boldsymbol{q}_r \triangleq [x_r\ y_r\ \varphi_r]^\top$	Wektor konfiguracji referencyjnej
$\boldsymbol{u}_r \triangleq [\omega_r\ v_r]^\top$	Wektor prędkości referencyjnych platformy robota w układzie lokalnym
$\boldsymbol{u} \triangleq [\omega\ v]^\top$	Wektor sygnałów sterujących, odpowiadający prędkościom robota w układzie lokalnym
$\boldsymbol{\Omega}_d \triangleq [\omega_{pd}\ \omega_{ld}]^\top$	Wektor prędkości zadanych kół robota
$\boldsymbol{q} \triangleq [x\ y\ \varphi]$	Wektor konfiguracji platformy robota
$\boldsymbol{\Omega} \triangleq [\omega_p\ \omega_l]^\top$	Wektor prędkości kół robota

3.1 Do własnego katalogu roboczego skopiować i rozpakować plik C:\laboratorium\Robotyka\cw4\MTracker_matlab_interface.zip, a następnie wykonać poszczególne zadania

- Podłączyć robota (zgodnie z Rys. 3).
- Pprzy pomocy menedżera urządzeń sprawdzić numer portu szeregowego przydzielonego platformie MTracker.
- Zapoznać się z kodem skryptu MTracker.m, szczególną uwagę zwrócić na pierwszą linię kodu, w której należy wpisać odpowiedni numer portu COM. Ustawić odpowiednie wartości zmiennej T_f określającej długość trwania eksperymentu oraz T_s określającej czas trwania jednej pętli sterowania.

3.2 W funkcji *ComputeWheelsVelocities* przyjmującej na wejściu prędkości robota $\boldsymbol{u} = [v\ \omega]^\top$, zaimplementować (zgodnie z wzorami podanymi we wstępie teoretycznym) przekształcenie zwracające na wyjściu wartości prędkości obrotowych kół $\boldsymbol{\Omega}_d = [\omega_{pd}\ \omega_{lf}]^\top$. Dodatkowo, w pliku ComputeRobotVelocities.m zaimplementować transformację prędkości obrotowych kół robota na prędkości platformy, tzn. $\boldsymbol{\Omega}_d \rightarrow \boldsymbol{u}$.

3.3 Przystosować funkcję *Controller*, pobierającą na wejściu chwilowe wartości wektora konfiguracji $\boldsymbol{q}$ oraz aktualny czas symulacji t w taki sposób, aby na wyjściu zwrócone zostały referencyjne prędkości robota $\boldsymbol{u}_r \triangleq [\omega_r \ v_r]^\top$ wyznaczone przy pomocy funkcji *TrajectoryGenerator*. Przebieg sygnałów powinien być zgodny z Rys. 5

Uwaga 1. *Funkcja TrajectoryGenerator powinna zwracać na wyjściu referencyjny wektor konfiguracji $\boldsymbol{q}_r$ oraz referencyjne prędkości zadane robota $\boldsymbol{u}_r$.*

3.4 Zmodyfikować funkcję *TrajectoryGenerator* w taki sposób, aby zrealizowana została następująca sekwenca sygnałów zadanych prędkości kół robota:

$$\begin{cases} \omega_{pd} = \omega_{ld} = +20[rad/s], & \text{dla} \quad t \in [0, 2), \\ \omega_{pd} = \omega_{ld} = -20[rad/s], & \text{dla} \quad t \in [2, 4), \\ \omega_{pd} = \omega_{ld} = 0[rad/s], & \text{dla} \quad t \in [4, \infty) \end{cases} \tag{15}$$

gdzie czas t wyrażony został w sekundach. Powyższe przebiegi prędkości kół odpowiadają trajektorii w postaci odcinka prostoliniowego na płaszczyźnie (x, y) (ruch *do przodu/do tyłu* ze stałą orientacją platformy).

Uwaga 2. *Podana we wzorze* (15) *sekwencja prędkości kół robota powinna zostać zaimplementowana w funkcji TrajectoryGenerator, następnie korzystając z funkcji ComputeRobotVelocities (zaimplementowanej w zadaniu 3.2) powinien zostać wyznaczony sygnał $\boldsymbol{u}_r$ odpowiadający aktualnym sygnałom ω_{pd} i ω_{ld}.*

Uwaga 3. *Wartość referencyjnego wektora konfiguracji powinna być ustawiona na $\boldsymbol{q}_r = [0, 0, 0]^T$. Ze względu na to, że przebieg referencyjnego wektora konfiguracji nie jest obliczany, w trakcie eksperymentu jedynym sygnałem rysowanym on-line będzie aktualna pozycja robota.*

Po przeprowadzonym eksperymencie:

- Wykreślić i porównać przebiegi sygnałów zadanych $\omega_{pd}(t), \omega_{ld}(t)$ oraz odczytanych sygnałów rzeczywistych $\omega_p(t), \omega_l(t)$. Doświadczenie wykonać dla kół obracających się bez kontaktu z podłożem oraz dla robota umieszczonego na płycie boiska. Badania można powtórzyć dla innych poziomów prędkości $\omega_{pd}(t), \omega_{ld}(t)$ rejestrując kilka okresów sygnałów zadanych. Czy rzeczywista orientacja robota w obu przypadkach jest stała? Jakie zjawisko można zaobserwować w przypadku sygnałów $\omega_p(t), \omega_l(t)$ i jak można je uzasadnić?
- Wykreślić przebieg odczytanej orientacji $\varphi(t)$ robota. Co może być przyczyną dryfu orientacji przy zmianie wartości zadanych prędkości kół robota?

3.5 Odnosząc się do wzorów z części teoretycznej, zmodyfikować funkcję *TrajectoryGenerator* w taki sposób aby możliwe było wykonanie przez robota:

- trajektorii kołowej
- trajektorii o kształcie ósemki (zdefiniowanej przy pomocy krzywych Lissajous)

Zarejestrować wyniki pomiarów i przeanalizować jakość realizacji zadanej trajektorii w przestrzeni zadania.

- Czy trajektoria w kształcie koła/ósemki jest realizowana idealnie? Jeśli nie, co wpływa na niedokładności odtwarzania trajektorii?
- Czy realizacja trajektorii w Matlabie odpowiada trajektorii rzeczywistj realizowanej przez robota? Jak uzasadnić ewentualne różnice?

References

[1] K. Kozłowski, J. Majchrzak. Nowe algorytmy sterowania nieholonomicznym robotem mobilnym. *XIV Krajowa Konferencja Automatyki*, strony 625–632, Zielona Góra, 2002.

[2] M. Michałek, K. Kozłowski. Sterowanie nieholonomicznym robotem mobilnym metodą orientowania pola wektorowego. K. Tchoń, redaktor, *Postępy Robotyki*, strony 235–246. Wydawnictwa Komunikacji i Łączności, Warszawa, 2005.

[3] K. Tchoń, A. Mazur, I. Dulęba, R. Hossa, R. Muszyński. *Manipulatory i roboty mobilne. Modele, planowanie ruchu, sterowanie.* Akademicka Oficyna Wydawnicza PLJ, Warszawa, 2000.