

Opis ćwiczeń na laboratorium obiektów ruchomych

Implementacja algorytmu sterowania robotem w środowisku symulacyjnym gry robotów w piłkę nożną stworzonym w Katedrze Systemów Automatyki Politechniki Gdańskiej

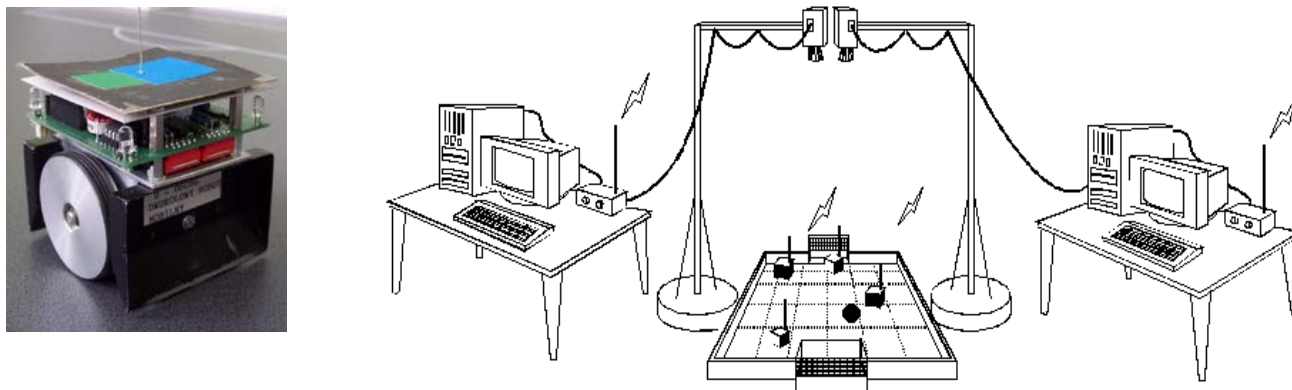
mgr inż. Piotr Fiertek

pfiertek@wp.pl

1. Wstęp

Ćwiczenie ma za zadanie zapoznanie się studentów z problemami występującymi podczas syntezy i implementacji algorytmu sterowania dwukołowym robotem mobilnym. Ćwiczenie składa się z dwóch części. W części pierwszej tworzony jest program sterujący robotem mobilnym w środowisku symulacyjnym, służącym do symulacji zachowania się grupy robotów na boisku do gry w piłkę nożną. W części drugiej planowana jest implementacja stworzonego algorytmu sterowania w rzeczywistym układzie sterowania robotem.

Środowisko symulacyjne nazywane serwerem symulacji nawiązuje do rozgrywek robotów w lidze MiroSot organizacji FIRA (Federation of International Robosoccer Association) (www.fira.net). W lidze tej roboty (7,5cm x 7,5cm x 7,5cm) grają na boisku o rozmiarze 150cm x 130cm. Mecz jest rozgrywany przez dwie drużyny. W skład każdej drużyny wchodzi trzy roboty oraz komputer z systemem wizyjnym (framegraber oraz kamera umieszczona nad boiskiem) (rys.1). System wizyjny ma za zadanie obserwować sytuację na boisku i dzięki odpowiednio napisanemu oprogramowaniu do obróbki obrazu, określać położenie i orientację kątową poszczególnych zawodników oraz położenie piłki. Uzyskane dane przekazywane są następnie do aplikacji odpowiedzialnej za sterowanie robotami. Programy sterujące po przeanalizowaniu otrzymanych danych wyznaczają sygnały sterujące, które następnie drogą radiową przesyłane są do robotów.



Rys. 1. Rozgrywki robotów w piłkę nożną w lidze MiroSot organizacji FIRA

W celu umożliwienia łatwego tworzenia i testowania programów sterujących pracujących w wyżej opisanym środowisku, stworzony został serwer symulacji naśladowujący środowisko gry robotów w piłkę nożną. Opisany serwer symulacji zastępuje rzeczywiste boisko, roboty, piłkę, system wizyjny oraz kanał komunikacji radiowej między komputerem a robotami. Programy sterujące otrzymują od serwera informację na temat stanu symulowanej gry (położenie i orientacja kątowa robotów, położenie piłki, czas wirtualny, stan gry) jednocześnie mają możliwość wpływania na przebieg gry poprzez wysyłanie sygnałów sterujących do poszczególnych robotów.

Dodatkowo serwer symulacji umożliwia umieszczenie na boisku dodatkowych obiektów takich jak: skrzynie, ściany oraz słupy. Dodatkowe obiekty umożliwiają rozszerzenie repertuaru zadań jakie mogą wykonywać roboty w stworzonym środowisku symulacyjnym (np. przejechanie przez labirynt, współpraca dwóch robotów w celu przepchnięcia ciężkiej skrzyni).

2. Przygotowanie do pracy z serwerem symulacji

W celu rozpoczęcia pracy z serwerem symulacji należy wpieryw zapoznać się instrukcją obsługi serwera symulacji dołączoną do opisu ćwiczenia oraz z opisem implementacji prostego algorytmu sterowania przy pomocy dostarczonego wraz z serwerem symulacji kodu źródłowego biblioteki DLL lub dostarczonego kodu źródłowego programu komunikującego się z serwerem symulacji za pomocą protokołu UDP.

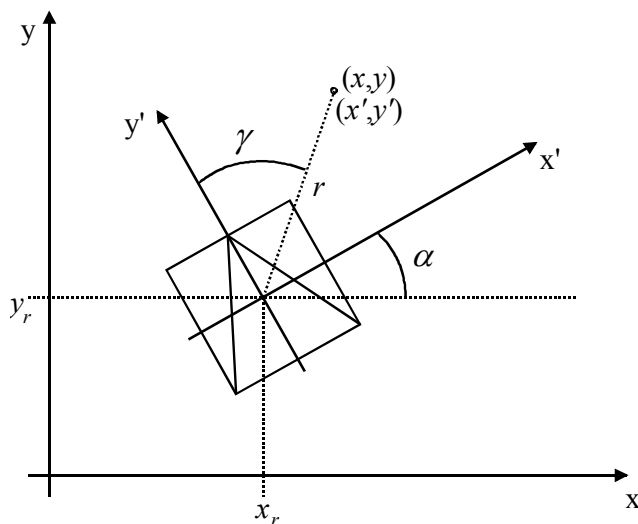
Następnie wysłać pojedyncze rozkazy do serwera symulacji w celu upewnienia się czy program sterujący prawidłowo łączy się z serwerem symulacji i czy sterowany robot prawidłowo wykonuje polecenia programu sterującego.

Kolejnym krokiem jest opracowanie algorytmu sterowania robotem, realizującego zleczone przez prowadzącego zadanie.

Po opracowaniu odpowiedniego algorytmu sterowania, należy zaimplementować algorytm w dowolnym języku programowania (w szczególności w języku C++), skompilować program sterujący, uruchomić serwer symulacji i sprawdzić czy roboty zachowują się zgodnie z naszymi oczekiwaniami. Jeżeli roboty zachowują się niezgodnie z naszymi założeniami, należy wrócić do punktu implementacji lub opracowania algorytmu sterowania.

3. Podążanie robota do punktu docelowego

Naszym celem jest przestawienie robota z położenia początkowego (x_0, y_0) do położenia końcowego (x, y) . Dla uproszczenia zakładamy, że po drodze nie ma przeszkód, które mogłyby zakłócić ruch naszego robota. W celu realizacji postawionego zadania przyjmujemy następujący opis położenia punktu docelowego robota (rys.2):



Rys.2. Położenie punktu docelowego w lokalnym układzie współrzędnych robota.

Na rysunku 2 widzimy robota, którego środek leży w punkcie (x_r, y_r) oraz punkt (x, y) będący punktem docelowym przedstawionym w globalnym układzie współrzędnych. (x', y') jest to ten sam punkt przedstawiony w lokalnym układzie odniesienia (z punktu widzenia robota). Można go również przedstawić za pomocą współrzędnych biegunowych γ i r .

Przeliczenie na lokalny układ współrzędnych (wyznaczenie współrzędnych punktu docelowego w lokalnym układzie odniesienia) otrzymujemy z wzoru (1).

$$\begin{aligned} a &= \sin(\alpha) & b &= \cos(\alpha) \\ \begin{cases} x' = (y - y_r)a + (x - x_r)b \\ y' = (y - y_r)b - (x - x_r)a \end{cases} & & (1) \\ \begin{cases} v'_X = b \cdot v_X + a \cdot v_Y \\ v'_Y = -a \cdot v_X + b \cdot v_Y \end{cases} & & \end{aligned}$$

gdzie: x_r, y_r - współrzędne robota w globalnym układzie współrzędnych; a - orientacja kątowa; x, y - współrzędne punktu docelowego w globalnym układzie współrzędnych; x', y' - współrzędne punktu docelowego w lokalnym układzie współrzędnych; v_X, v_Y, v'_X, v'_Y - przeliczenie wektorów prędkości z układu globalnego na układ lokalny

W najprostszym przypadku możemy przyjąć następującą strategię sterowania:

1. Gdy kąt γ jest duży, należy zmniejszyć prędkość liniową robota i ustalić prędkość obrotową gwarantującą zmniejszenie kąta γ .
2. Gdy kąt γ jest mały, należy zmniejszyć prędkość obrotową robota i zwiększyć prędkość liniową robota do v_z (maksymalna prędkość robota).
3. Gdy robot dojedzie do punktu docelowego należy zatrzymać robota lub przełączyć się na następny punkt docelowy (śledzenie trajektorii).
4. Ponieważ robot może poruszać się również do tyłu, w wielu przypadkach dążenie do spełnienia warunku $\gamma = 0$ jest nieuzasadnione. Jeśli $\gamma = 170^\circ$, w celu obrócenia robota przodem do punktu docelowego należy go obrócić w lewo o kąt 170° . W tym przypadku lepiej jest jednak wykonać obrót w prawo o 10° i rozpocząć jazdę do tyłu.

Pierwsze dwie zasady może spełnić następujący przepis na sygnał sterujący:

$$\begin{aligned}
 k(t) &= k_1 \cdot \gamma(t) \\
 \text{if } (k(t) < -\frac{\pi}{2}) \text{ then } k(t) &= -\frac{\pi}{2} \\
 \text{if } (k(t) > \frac{\pi}{2}) \text{ then } k(t) &= \frac{\pi}{2} \\
 v_s(t) &= k_3 \cdot \psi(t) \cdot v_z \cdot \cos(k(t)) \\
 \omega_s(t) &= -k_2 \cdot \omega_z \cdot \gamma(t)
 \end{aligned} \tag{2}$$

$\omega_s(t)$ - wymagana prędkość obrotowa; $v_s(t)$ - wymagana prędkość liniowa robota; ω_z - maksymalna prędkość kątowa; v_z - maksymalna prędkość liniowa; $k(t)$ - zmienna pomocnicza; k_1, k_2 i k_3 - parametry sterownika

Aby zrealizować wymaganie 4 wprowadzamy dodatkową zmienną $\psi(t)$, która przyjmuje wartość 1, gdy robot porusza się do przodu oraz -1, gdy chcemy poruszać się do tyłu. Aby sterownik pracował poprawnie musimy wprowadzić mechanizm zmiany kierunku ruchu. Naturalnym wydaje się sprawdzenie wartości y' (rys.2) a następnie wyznaczenie kąta γ po odpowiednich modyfikacjach wartości x' i y' , jeśli jest to konieczne (3).

$$\begin{aligned}
 \text{if } (y' < 0) \text{ then } \{ \quad \psi = -1 \quad y' = -y' \quad x' = -x' \quad \} \\
 \text{if } (y' \geq 0) \text{ then } \{ \quad \psi = 1 \quad \}
 \end{aligned} \tag{3}$$

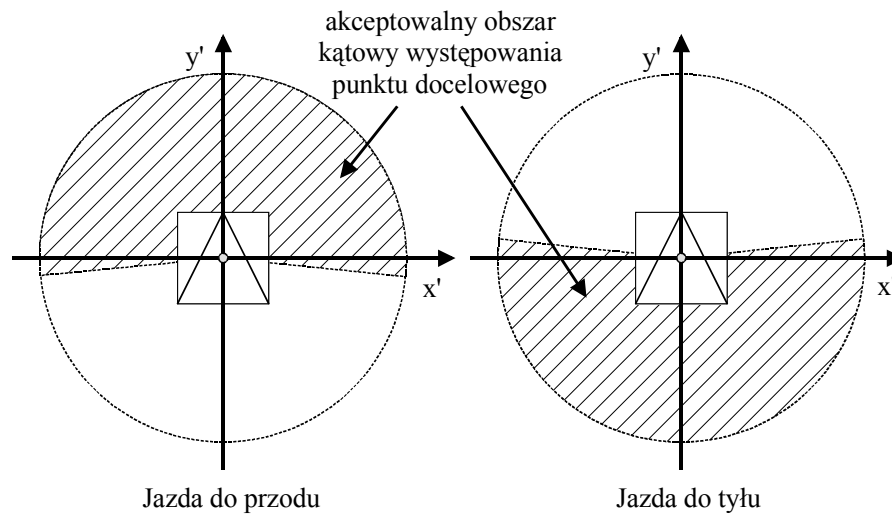
gdzie x' i y' są współrzędnymi punktu docelowego w układzie lokalnym robota.

Dla przedstawionego przypadku, przełączenie między jazdą do przodu i do tyłu nastąpi za każdym razem gdy kąt γ , pod jakim widziany jest punkt docelowy, będzie leżał poza przedziałem $\langle -90^\circ; 90^\circ \rangle$. Mechanizm ten, choć bardzo prosty ma pewną wadę. Dla kąta $\gamma \approx \pm 90^\circ$ oraz zerowej prędkości liniowej, może dojść do wielokrotnych przełączeń kierunku jazdy robota. Układ sterujący będzie próbował obrócić robota w kierunku punktu docelowego. Ponieważ kąt γ znajduje się na linii przełączającej ($\gamma \approx \pm 90^\circ$), z powodu bezwładności robota i układu sterującego, robot będzie próbował obrócić się raz w prawo raz w lewo. Rozwiązaniem tego problemu jest wprowadzenie pętli histerezy w ustaleniu kierunku jazdy (rys.3). Szerokość pętli histerezy wynosi 10° . Oznacza to, że jeśli robot jedzie do przodu ($\psi(t)=1$), przełączenie na jazdę do tyłu ($\psi(t)=-1$) nastąpi w momencie gdy kąt γ pod którym widziana jest przeszkoda będzie większy od 95° lub mniejszy od -95° . Przełączenie w drugą stronę nastąpi gdy kąt γ będzie należał do przedziału $\langle -85^\circ; 85^\circ \rangle$.

W drugim przypadku najpierw zmieniamy kąt α w modelu lokalnym robota, tak aby odpowiadał on aktualnemu kierunkowi jazdy (4):

$$\text{if } (\psi = -1) \text{ then } \{ \quad \alpha = \alpha + \pi \quad \} \tag{4}$$

Następnie wyznaczamy współrzędne lokalne (x', y') punktu docelowego i wyznaczamy kąt γ . Gdy kąt γ leży poza obszarem $\langle -95^\circ; 95^\circ \rangle$ następuje przełączenie kierunku jazdy, tzn. zmienna $\psi(t) = -\psi(t)$; $x' = -x'$; $y' = -y'$; i ponowne wyznaczenie kąta γ .



Rys.3. Przełączanie kierunku jazdy robota w zależności od położenia punktu docelowego

4. Opis proponowanych ćwiczeń

Protokół opisujący komunikację między programem sterującym a serwerem symulacji umożliwia podanie przez program sterujący nazwy drużyny lub wyświetlenie napisu w oknie komunikatów serwera. Przy zaliczaniu zadań z projektu, wymagane jest aby program sterujący, zaraz po uruchomieniu symulacji, przedstawił autorów programu oraz treść realizowanego zadania.

Ćwiczenie 1 (ocena 3-4-5)

Napisać program mający na celu poprowadzenie robota wzdłuż trajektorii. Zadana trajektorię robota stanowią punkty kontrolne, które ma odwiedzić robot w określonej kolejności. Robot w momencie dojechania do punktu kontrolnego ma się zatrzymać i wykonać obrót o 180° wokół własnej osi. Punkty trajektorii nie są z góry znane i muszą być wpiery wyznaczone. Nie jest również określone położenie początkowe sterowanego robota.

Ustalenie trajektorii robota:

- Na boisku znajduje się kilka robotów (liczba robotów nie jest z góry określona). Rzutujemy położenia robotów i piłki na oś y ($x=0$). Wymaganą trajektorię robota uzyskujemy po posortowaniu otrzymanych punktów w kolejności od najmniejszej do największej wartości współrzędnej y .
- Trajektoria robota ma być tak wyznaczona, aby robot objechał piłką 2 razy w kierunku ruchu wskazówek zegara, dojechał do środka boiska a następnie wrócił i objechał dwukrotnie piłkę 2 razy w kierunku przeciwnym do ruchu wskazówek zegara. W tym przypadku nie jest wymagane aby robot obracał się po dotarciu do każdego punktu zaplanowanej trajektorii.
- Schemat zachowania robota ustalony wraz z prowadzącym projekt.

Ćwiczenie 2 (ocena 3-4)

Napisać program sterujący robotem, który będzie wbijał nieruchomą piłkę do bramki przeciwnika. Na boisku znajdować się będzie tylko jeden robot i piłka. Początkowe położenie piłki i robota nie jest z góry określone.

Ćwiczenie 3 (ocena 4-5)

Napisać program sterujący robotem, który będzie omijał slalomem tor przeszkód składający się z czterech nieruchomych robotów i na koniec uderzał w nieruchomą piłkę. Wszystkie roboty i piłka będą ustawione wzdłuż linii (w szczególnym przypadku wyznaczającej oś symetrii boiska). Położenie początkowe sterowanego robota oraz piłki jest określone. Dokładne położenie pozostałych czterech robotów nie jest znane. Wiadomo tylko, że będą one umieszczone wzdłuż odcinka łączącego sterowanego robota i piłkę. Odległość pomiędzy kolejnymi robotami będzie na tyle duża, aby robot mógł swobodnie przejechać między nimi. W trakcie pokonywania slalomu nie można zahaczyć o żadnego z robotów.

Ćwiczenie 4 (ocena 4)

Napisać program sterujący robotem, który będzie realizował funkcję bramkarza. Położenie początkowe robota nie jest znane. Nie jest również znane położenie i prędkość piłki. Wiadomo tylko, że początkowe położenie i prędkość piłki oraz początkowe położenie robota będą tak dobrane, aby doszło do strzelenia gola w momencie gdyby robot nie wykonał odpowiedniej akcji.

Ćwiczenie 5 (ocena 4-5)

Napisać program sterujący dwoma robotami. Jeden z nich będzie realizował funkcję bramkarza. Drugi, funkcję napastnika strzelającego gole. Położenie początkowe robotów nie jest znane. Nie jest również znane położenie i prędkość piłki. Oba roboty nie mogą się ze sobą zderzać.

Ćwiczenie 6 (ocena 4-5-6)

Napisać program sterujący dwoma robotami bawiącymi się w berka. Jeden robot będzie robotem goniącym a drugi robotem uciekającym. W momencie gdy robot goniący dogoni robota uciekającego ma nastąpić zmiana ról. Aby umożliwić oddalenie się robota uciekającego od robota goniącego na bezpieczną odległość, po złapaniu robota, robot goniący ma stać bez ruchu przez czas równy 5s. Na boisku znajduje się również piłka, która powinna być omijana przez oba roboty. Możliwe jest też umieszczenie na boisku dodatkowych, nieruchomych robotów, które również powinny być omijane przez goniące się roboty. Początkowe położenie robotów i piłki nie jest znane.

Ćwiczenie 7 (ocena 5-6)

Napisać program sterujący robotem, który będzie wbijał poruszającą się piłkę do bramki przeciwnika. Na boisku znajdować się będzie tylko jeden robot i piłka. Początkowe położenie piłki i robota nie jest z góry określone. Początkowe położenie i prędkość piłki będą tak dobrane aby robot miał szansę przejąć piłkę w ruchu i skierować ją w kierunku bramki.

Ćwiczenie 8 (ocena 5-6)

Napisać program sterujący trzema robotami, które po rozpoczęciu gry ustawią się w rogach trójkąta a następnie będą sobie nawzajem podawały piłkę. W trakcie podawania piłki, roboty mogą zamieniać się miejscami.

Ćwiczenie 9 (ocena 5-6)

Napisać program sterujący robotem, którego zadaniem jest przepchanie skrzyni z jednego punktu boiska do innego punktu boiska. W zależności od trudności zadania na boisku mogą znajdować się dodatkowe skrzynie, ściany lub słupy (w najprostszej wersji na boisku znajduje się jedynie robot, skrzynia i piłka).

Ćwiczenie 10 (ocena 6)

Napisać program sterujący robotem, którego zadaniem jest przejechanie przez labirynt składający się ze ścian. Punkt startowy i końcowy robota oraz położenie ścian labiryntu nie jest z góry określone.