Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Rozprawa doktorska

Algorytmy wysokiej dokładności śledzenia trajektorii robota przemysłowego

mgr inż. Wojciech Zwonarz

Promotor

prof. dr hab. inż. Andrzej Turnau

Składam serdeczne podziękowania mojemu promotorowi Panu profesorowi Andrzejowi Turnauowi za cenne wskazówki, wyrozumiałość oraz ogromną cierpliwość, bez której ta praca nigdy by nie powstała.

Streszczenie

Praca poświęcona jest konstrukcji algorytmów wysokiej dokładności śledzenia trajektorii robota przemysłowego. Przedstawiono w niej opisy trajektorii końcówki robota przemysłowego w przestrzeni przegubowej robota. Pokazano trzy podstawowe trajektorie: ruch wzdłuż prostej, ruch po linii śrubowej oraz ruch wokół elipsy. W celu ograniczenia stosowania równań kinematyki odwrotnej zaproponowano dwie metody aproksymacji trajektorii w przestrzeni przegubowej. Kolejno są to metody PTP (ang. Point to Point – punkt do punktu) oraz spline. Obydwie metody zostały przedstawione w dwóch wariantach, z użyciem wielomianów stopnia trzeciego oraz piątego. Jako rozszerzenie metod wielomianowych wprowadzono metodę aproksymacji przy pomocy funkcji Beziera.

Po zaprezentowaniu różnych rodzajów trajektorii przedstawiono wybrane algorytmy sterujące. W literaturze istnieje wiele typów regulatorów nadążnych. Od prostych typu PD czy PID, poprzez regulatory adaptacyjne, po wyrafinowane konstrukcje wspierane sieciami neuronowymi. W wielu wypadkach znajomość dynamiki manipulatora oraz trajektorii a priori jest użyta w konstrukcji algorytmu, aby minimalizować błąd nadążny.

W pracy przybliżono podstawowe pojęcia z zakresu kinematyki oraz dynamiki robotów, które były niezbędne, aby przedstawić algorytmy sterowania klasycznego, adaptacyjnego oraz predykcyjnego.

Algorytmom predykcyjnym poświęcono w pracy szczególną uwagę. Pokazano ich ogólną strukturę oraz poszczególne kroki niezbędne w syntezie sterowania. Przedstawiono szczegółowe analizy predykcji numerycznej dokonanej na modelu manipulatora podczas pracy regulatora. Eksperymenty na robocie rzeczywistym ilustrują podejście, które już na etapie wstępnym generuje sterowania na podstawie numerycznego poszukiwania minimum funkcji kary. W ten sposób wprowadza się mechanizm unikania przeszkód na ścieżce ruchu robota, ograniczenia pola ruchu związane z geometrią konstrukcji robota i otoczenia robota. Powyższe rozważania i rozwiązania są oryginalnym wkładem autora w dziedzinę nieliniowych regulatorów predykcyjnych, których gwałtowny rozwój ma miejsce obecnie. Ponadto przeprowadzono analizę czasów wykonania algorytmu predykcyjnego, zilustrowaną wykresami jitteru.

Aby opis algorytmów predykcyjnych był kompletny, w pracy przedstawiono dwa sposoby identyfikacji parametrycznej robota przemysłowego Mitsubishi RV-2F. Pokazano identyfikację metodami klasycznymi i z użyciem sieci neuronowych. Bez modelu robota algorytmy predykcyjne nie mogły by istnieć, dlatego identyfikacja stanowi klucz do budowy sterowników tego rodzaju.

Pracę zamyka opis algorytmów wizyjnych i przetwarzania obrazu podczas pracy manipulatora. Przedstawiono, jak śledzi się położenia przegubów i końcówki robota niezależnie od elementów elektromechanicznych oraz równocześnie przeszkody należące do środowiska, których położenie może być uwzględniane w algorytmach predykcyjnych w postaci funkcji kary.

Abstract

This paper is devoted to the algorithms of high accuracy industrial robot trajectory tracking. The details of the different trajectories of the robot effector are described. Author focused on the three basic trajectories: straight helix and elliptical. In order to minimalize the need of the reverse kinematics equation usage, the two basic methods of the trajectory approximation were proposed. The PTP (Point To Point) method and the spline method. Both methods were shown in two variants. With the usage of the 3rd and 5th polynomial. As the extension of the polynomial methods the Bezier functions were introduced.

After trajectory description the selected regulators are presented. A lot of types of the controllers is present in the accessible literature. Starting from the simple PD and PID controllers through the adaptive controllers up to sophisticated algorithms which makes use of the neural network. In many cases the a priori knowledge of the robot dynamics and the chosen trajectory is used in control algorithm to minimalize the tracking error.

In the paper the basic concepts in the field of robot kinematics and dynamic were introduced. The introduced concepts were needed to show how the classic, adaptive and predictive controllers work.

The nonlinear predictive controllers take important place in the paper. Their general structure is shown and the steps necessary to generate the control algorithm. The analyze of the numerical prediction is shown. The experiments on the real robotic system were performed to illustrate the approach which allows to control the robot based only on the penalty function minimalization. In this way the collision avoidance algorithm was introduced to the controller structure without additional steps. The approach shown in the chapter are the original content made by author. In addition, the real-time analysis of the controller behavior is made which is illustrated with the jitter plots.

To make the NPC depiction completed in the paper two methods of parametric identification of the industrial robot Mitsubishi RV-2F were shown. The classical and with the usage of the neural network methods. Without the model of the robot the predictive control algorithms could not exist, that is the reason why the parametric identification is the key to the synthesis of this regulator's group.

The last part of the paper is description of the visual systems and the image processing during the robot work. It is shown how the effector and robot position can be tracked without the need to use any electromechanical sensors. In the same time the environmental obstacles can be detected and tracked. Their position may be used in the collision avoidance algorithms as the input for the penalty function calculation.

Spis treści

Tezy pracy

Teza 1: Dzięki znajomości modelu matematycznego robota przemysłowego, możliwym jest zaprojektowanie regulatora przewidującego zachowanie końcówki manipulacyjnej robota i obliczanie sterowania, które pozwala zwiększyć precyzję ruchu.

Teza 2: Do przewidywania dynamiki ramienia robotycznego można używać całkowań numerycznych, które pozwalają na obliczanie sterowania w czasie rzeczywistym.

Teza 3: Horyzont predykcji dynamiki ruchu jest zależny od planowanego typu oraz przebiegu trajektorii manipulatora robotycznego.

Teza 4: Planowanie trajektorii w sposób parametryczny pozwala uprościć algorytmy sterowania predykcyjnego.

Wstęp

Podążanie za zadaną trajektorią jest jednym z podstawowych zadań robotycznych. Ścieżka ruchu robota dobierana jest zawsze w taki sposób, aby narzędzie zamontowane na nim było w stanie wykonać zaplanowane zadanie. Gdy znamy zadanie postawione przed systemem, możliwy jest wybór najlepszej trajektorii do jego wykonania. Istnieje niemal nieskończona liczba trajektorii do wyboru. Niektóre z nich łączą punkty w przestrzeni w najkrótszym możliwym czasie. Inne pozwalają na uzyskanie jak najmniejszych przyspieszeń w przegubach. Jeszcze inne podążają za zadanym kształtem geometrycznym.

Gdy właściwa dla zadania trajektoria została już wybrana, kolejnym krokiem jest konstrukcja algorytmu sterowania, który będzie w stanie za nią podążać, czy też inaczej mówiąc, śledzić. W literaturze istnieje wiele typów regulatorów nadążnych. Od prostych typu PD czy PID, poprzez regulatory adaptacyjne, po wyrafinowane konstrukcje wspierane sieciami neuronowymi. W wielu wypadkach znajomość dynamiki manipulatora oraz trajektorii a priori jest użyta w konstrukcji algorytmu, aby minimalizować błąd nadążny. Szczególnie jest to widoczne w algorytmach predykcyjnych, które dzięki znajomości przyszłych położeń oraz sił oddziałujących na narzędzie, są w stanie reagować na sytuacje krańcowe z wyprzedzeniem.

Niniejsza praca poświęcona jest konstrukcji algorytmów wysokiej dokładności śledzenia trajektorii robota przemysłowego. Składa się ona z ośmiu rozdziałów.

W rozdziale pierwszym opisano metody wyznaczania kinematyki robotów przemysłowych o otwartym łańcuchu kinematycznym. Przedstawiono pojęcia zadania prostego kinematyki oraz zadania odwrotnego. Dla zadania prostego pokazano systemowe metody wyznaczania tego zadania. Wprowadzono pojęcie jakobianu robota przemysłowego, które służy do wyznaczania dynamiki manipulatora w kolejnych rozdziałach. Uzupełnieniem rozdziału pierwszego jest przedstawienie idei postaci normalnych kinematyki jako systemowego podejścia upraszczającego pracę z punktami osobliwymi w przestrzeni zmiennych przegubowych robota.

Zagadnienia wyznaczania dynamiki manipulatorów-robotów scharakteryzowano w rozdziale drugim. Przedstawiono dwa formalizmy: Eulera-Lagrange'a oraz Newtona-Eulera, które dają równoważne rezultaty, jednak prowadzą do nich dwiema różnymi drogami. Dynamika manipulatora-robota jest przedstawiona w postaci nieliniowego równania macierzowego, szeroko stosowanego w literaturze. Specyficzne właściwości oraz cechy równania dynamiki opisano szczegółowo. Pokazano również metodę linearyzacji w torze sprzężenia zwrotnego, która jest stosowana w dalszych częściach pracy. Przedstawiono uproszczoną dynamikę siłowników elektromechanicznych w robotyce.

W rozdziale *Aproksymacje trajektorii robota* zawarto opisy trajektorii końcówki robota przemysłowego w przestrzeni przegubowej robota. Pokazano trzy podstawowe trajektorie: ruch wzdłuż prostej, ruch po linii śrubowej oraz ruch wokół elipsy. W celu ograniczenia stosowania równań kinematyki odwrotnej zaproponowano dwie metody aproksymacji trajektorii w przestrzeni przegubowej. Kolejno są to metody PTP (ang. Point to Point – punkt do punktu) oraz spline. Obydwie metody zostały przedstawione w dwóch wariantach,

z użyciem wielomianów stopnia trzeciego oraz piątego. Jako rozszerzenie metod wielomianowych wprowadzono metodę aproksymacji przy pomocy funkcji Beziera.

Rozdział czwarty z wstępnymi uwagami o sieci neuronowej dodano przed piątym rozdziałem o identyfikacji robotów przemysłowych. Sieć neuronowa posłuży do identyfikacji robota w następnym rozdziale, stąd pojawiła się potrzeba zaznaczenia, czym ona jest.

Identyfikacja parametryczna robota przemysłowego Mitsubishi RV-2F w rozdziale piątym jest prowadzona metodami klasycznymi i z użyciem sieci neuronowych. Bez identyfikacji parametrycznej modelu robota trudno wyobrazić sobie śledzenie trajektorii robota z wysoką dokładnością. Istnieje bowiem konieczność porównania trajektorii rzeczywistej robota ze wzorcem, czyli trajektorią modelową symulowaną. Dlatego identyfikacja stanowi klucz do budowy algorytmów wysokiej dokładności do śledzenia trajektorii robota.

Szósty rozdział poświęcono metodom sterowania robotami przemysłowymi. Na początku przywołano proste metody niezależnego sterowania poszczególnymi członami robota przemysłowego. Jako przykład takich regulatorów podano liniowe typu PD oraz PID. Pokazano, że takie regulatory mogą być stosowane z powodzeniem. Następnie przedstawiono bardziej złożone regulatory wykorzystujące podczas wyznaczania sterowania metodę wyliczonego momentu. Na jej podstawie zilustrowano sterowanie wieloma członami jednocześnie, gdy znana jest zadana trajektoria. Kolejno odniesiono się do regulacji adaptacyjnej, która pozwala na redukcję błędu sterowania nawet w sytuacji niepewności parametrów modelu. Jest to ważna grupa sterowników, gdyż błędy są zawsze obecne w zastosowaniach rzeczywistych. Kontynuując temat sterowania robotami, przedstawiono użycie sieci neuronowych. Pokazano regulator PD wsparty wielowarstwową siecią uczącą się.

Kolejny rozdział poświęcono regulatorom predykcyjnym. Pokazano ich ogólną strukturę oraz poszczególne kroki niezbędne w syntezie sterowania. Przedstawiono szczegółowe analizy predykcji numerycznej dokonanej na modelu manipulatora podczas pracy regulatora. Eksperymenty na robocie rzeczywistym ilustrują podejście, które już na etapie wstępnym generuje sterowania na podstawie numerycznego poszukiwania minimum funkcji kary. W ten sposób wprowadza się mechanizm unikania przeszkód na ścieżce ruchu robota, ograniczenia pola ruchu związane z geometrią konstrukcji robota i otoczenia robota. Powyższe rozważania i rozwiązania są oryginalnym wkładem autora w dziedzinę nieliniowych regulatorów predykcyjnych, których gwałtowny rozwój ma miejsce obecnie. Rozdział uzupełniono wykresami czasu wykonania algorytmu predykcyjnego.

Ostatni rozdział dotyczy algorytmów wizyjnych i przetwarzania obrazu podczas pracy manipulatora. Przedstawiono, jak śledzi się położenia przegubów i końcówki robota niezależnie od elementów elektromechanicznych. Równocześnie śledzi się przeszkody należące do środowiska, których położenie może być uwzględniane w algorytmach predykcyjnych w postaci funkcji kary. Rozdział zamykają, podobnie jak w poprzedniej części, wyniki pomiarów czasów wykonania algorytmów.

Pracę kończą: Podsumowanie, Bibliografia oraz cztery dodatki: A, B, C i D.

1. Kinematyka robotów przemysłowych

Kinematyka jest dedykowana badaniu związków pomiędzy ilością stopni swobody a położeniem, prędkością oraz przyspieszeniem wszystkich przegubów łączących człony robota. Kinematyka służy planowaniu akcji sterowania oraz pozwala na wyznaczenie sił i momentów działających na jednostki wykonawcze takie jak silniki, siłowniki etc. Uwzględnienie mas i momentów bezwładności jest nieodłącznie związane tylko z modelem dynamicznym robota. W modelu kinematycznym masy i momenty bezwładności nie występują.

W kinematyce robotów stosuje się prawa geometrii w analizie wieloczłonowych łańcuchów kinematycznych stanowiących konfigurację wybranego robota. Podczas tworzenia modelu kinematyki robota zakłada się, że każdy z członów jest bryłą sztywną, a ich połączenia są idealnymi przesunięciami lub obrotami, tzn. nie występują poślizgi oraz niepożądane wychylenia.

Dla każdego typu robota istnieją osobne grupy ściśle z nimi powiązanych kinematyk. W przemyśle główne zastosowanie znalazły roboty o otwartym oraz zamkniętym łańcuchu kinematycznym. Te dwie grupy będą szerzej przedstawione w dalszych rozdziałach. Należy jednocześnie zaznaczyć, że istnieje wiele innych grup robotów o kinematykach istotnie odmiennych, i budowanych w inny sposób, niż dwie wskazane grupy. Do takich kategorii należą m.in. roboty mobilne, humanoidalne, podwodne czy też wężom-podobne.

W analizie geometrycznej konstrukcji robota odnajduje się kinematykę prostą i odwrotną.

Zależnie od typu robota analiza może mieć różny stopień złożoności. W przypadku robotów o otwartym łańcuchu kinematycznym zadanie proste kinematyki jest mniej złożone niż zadanie odwrotne. W zadaniu z zamkniętym łańcuchem kinematycznym złożoność staje się problemem właśnie w zadaniu prostym, które jest często niemal nierozwiązywalne, a wyniki obliczeń podaje się zwykle w przybliżeniu przez obliczenia numeryczne.

Równania dla otwartego łańcucha kinematycznego

Roboty o otwartym łańcuchu kinematycznym są grupą robotów najczęściej stosowaną w przemyśle [26]. Należą tu wszystkie roboty o układzie kartezjańskim, sferycznym i_cylindrycznym. W otwartym łańcuchu kinematycznym pracują: manipulator o konfiguracji Stanford, manipulator antropomorficzny i robot SCARA. [51]

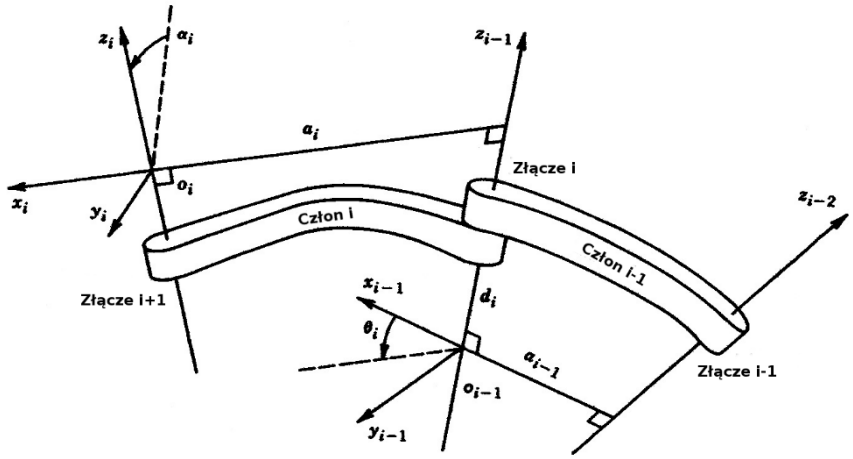
Kinematyka prosta

Opis otwartego łańcucha kinematycznego został sformalizowany już w latach sześćdziesiątych dwudziestego wieku. Wówczas pojawiły się zunifikowane systemy oznaczania członów robotów oraz schematy generowania wzorów kinematyki prostej [16] oraz [66]. Największą popularność zyskał system nazwany notacją Denavita–Hartenberga (DH).

Notacja Denavita–Hartenberga

Podstawowymi założeniami notacji DH są: sztywność poszczególnych członów, ruch odbywa się tylko względem osi X lub względem osi Z, oraz pojedyncze przekształcenie zawsze

wyrażone jest przez zestaw czterech parametrów (kąt obrotu względem osi Z, przesunięcie względem osi Z, przesunięcie względem osi X oraz kąt obrotu względem osi X). Na rysunku 1.1 przedstawiono sekwencję generowanych przez metodę układów współrzędnych.



Rys. 1.1 Ilustracja układów współrzędnych dla złącza *i* dla metody DH

Transformacja pomiędzy złączem *i-1* a *i* w postaci macierzowej:

$$A_i^{i-1} = RotZ(\Theta_i)\,TranZ(d_i)\,TranX(a_i)\,RotX(\alpha_i),$$

gdzie A_i^{i-1} jest wynikową macierzą transformacji, *Rot** oraz *Tran** są elementarnymi macierzami transformacji względem wskazanej osi.

Zmodyfikowana notacja Denavita–Hartenberga

Notacja zmodyfikowana od standardowej różni się kolejnością wykonywania przekształceń geometrycznych oraz innym przypisaniem układów współrzędnych. Kolejność przekształceń opisana jest wzorem:

$$A_i^{i-1} = RotX(\alpha_i)\,TranX(a_i)\,RotZ(\Theta_i)\,TranZ(d_i).$$

W notacji zmodyfikowanej, układ współrzędnych *i* jest przytwierdzony do początku członu *i*, nie zaś jak w notacji oryginalnej, gdzie jest on powiązany z jego końcem.

Pomimo prostoty założeń, notacja ta jest z powodzeniem używana nawet przy opisie tak skomplikowanych mechanizmów jak egzoszkielety [60].

Kinematyka odwrotna

W przypadku kinematyki odwrotnej nie istnieją znormalizowane równania poprawnie definiujące każdy typ robota należący do grupy robotów o łańcuchu otwartym. Każda konfiguracja członów jest indywidualnie analizowana, a generowane równania są jej dedykowane. [52]

Jakobian

Jakobian jest macierzą pochodnych cząstkowych funkcji. W ogólnym przypadku wyraża się wzorem:

$$[J]=\left[\frac{\partial f}{\partial x_1}\cdots\frac{\partial f}{\partial x_n}\right]=\begin{bmatrix}\frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n}\\ \vdots & \ddots & \vdots\\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n}\end{bmatrix}$$

W robotyce macierz tę tworzą pochodne po czasie równań kinematyki robota. Ten typ jakobianu nazywany jest jakobianem analitycznym. Opisuje przekształcenie prędkości zmiennych przegubowych w przestrzeń współrzędnych zewnętrznych. Spełnia zatem równanie:

$$\dot{y}=J^a(x)\dot{x}_1$$

Dla robotów możemy, poza jakobianem analitycznym, zdefiniować również jakobian geometryczny, spełniający równanie:

$$\begin{bmatrix}v_s\\ \omega_s\end{bmatrix}=J^s(q)\dot{q}$$

Jakobian geometryczny można uporządkować przez podział na dwie mniejsze macierze dla równań ruchu liniowego i obrotowego. Dla przemieszczeń linowych otrzymuje się:

$$J_v=\begin{bmatrix}J_{v_1} & \cdots & J_{v_n}\end{bmatrix}$$

gdzie:

$$J_{v_i}=\begin{cases}z_{i-1}(o_n-o_{i-1}), dla\ obrotowego\ i\\ z_i, dla\ liniowego\ i\end{cases}$$

Dla przemieszczeń obrotowych otrzymuje się:

$$J_\omega=\begin{bmatrix}J_{\omega_1} & \cdots & J_{\omega_n}\end{bmatrix}$$

gdzie:

$$J_{\omega_i}=\begin{cases}z_{i-1}, dla\ obrotowego\ i\\ 0, dla\ liniowego\ i\end{cases}$$

Na ogół Jakobian jest macierzą prostokątną co uniemożliwia jego odwracanie. Można jednak wyznaczyć macierz pseudoodwrotną, która spełnia wymagania dla macierzy odwrotnej. Postać pseudoodwrotna Jakobianu wygląda następująco:

$$JJ^+=I$$
$$J^+=J^T(JJ^T)^{-1}$$

gdzie macierz J^+ jest macierzą kwadratową.

Konfiguracje osobliwe

Przez konfiguracje osobliwe rozumie się położenia robota, w których rząd jego jakobianu analitycznego jest mniejszy od ilości przegubów robota. Zbiór takich położeń można zdefiniować jako:

$$S_m = \left\{ x \in \boldsymbol{R}^n \middle| rank\, J^a(x) < m \right\}$$

Gdy robot znajduje się w położeniu osobliwym nie można jednoznacznie rozwiązać zadania kinematyki odwrotnej. W punktach osobliwych wyznacznik macierzy dynamiki zeruje się. Badania nad konfiguracjami osobliwymi stanowią rozległą dziedzinę wiedzy w robotyce. Badane są m.in. metody unikania punktów osobliwych, czyli algorytmy wyznaczania kinematyki odwrotnej bez przejść przez punkty osobliwe.

Unikanie punktów osobliwych, narzędzia analityczne

Uniknięcie punktu osobliwego *x* jest możliwe wtedy i tylko wtedy, gdy istnieje taka nieosobliwa konfiguracja *x'*, że spełniona jest zależność:

$$k(x) = k(x')$$

Jeśli konfiguracja *x'* nie istnieje, to nie istnieje również możliwość uniknięcia konfiguracji osobliwej. Gdy *x'* znajduje się w bliskim otoczeniu wówczas mówimy, że *x* jest lokalnie możliwa do uniknięcia. Aby sprawdzić czy punkt osobliwy jest możliwy do uniknięcia można posłużyć się narzędziami analitycznymi [73] oraz [74].

Gdy stopień redundancji manipulatora wynosi jeden, wówczas możemy zbadać zachowanie układu dynamicznego pola własnego. Układ ten wyraża się wzorem: $\dot{x} = S(x)$ takim, że:

$$S(x) = (S_1(x), \ldots, S_n(x))^T$$
$$S_i(x) = (-1)^{i+1} det J^{ai}(x),\ i = 1, \ldots, n$$

gdzie J^{ai} to jakobian analityczny robota z pominiętą *i*-tą kolumną. Punktami równowagi układu są konfiguracje osobliwe. Gdy punkt równowagi jest asymptotycznie stabilny, wówczas możemy stwierdzić, że konfiguracja jest możliwa do uniknięcia. Jeśli punkt jest stabilny w sensie Lapunowa wówczas punkt osobliwy jest niemożliwy do zmiany.

W sytuacji, gdy stopień redundancji robota jest większy niż jeden, możliwość ominięcia konfiguracji osobliwej można sprawdzić przy pomocy badania stabilności miejsc zerowych stowarzyszonego z kinematyką hamiltonowskiego pola wektorowego. Aby zbudować powyższy aparat należy wybrać ciąg liczb całkowitych takich, że $1 \le i_1 < \ldots < i_{m+1} \le n$ i określić pole wektorowe

$$X_{i_1 \ldots i_{m+1}}(x) = \left(X_1(x), \ldots, X_n(x) \right)$$

gdzie:

$$X_i(x) = \begin{cases} 0, \ jeżeli\ i \neq i_1, i_2 \ldots, i_{m+1} \\ (-1)^{r+1} det\, J_{i_r}(x), \ jeżeli\ dla\ pewnego\ r = 1,2,\ldots,m+1\ zachodzi\ i = i_r \end{cases}$$

gdzie J_{ir} jest macierzą powstałą z kolumn jakobianu analitycznego o numerach i_1, …, i_{m+1} z usuniętą kolumną numer i_r. Korzystając z właściwości pól hamiltonowskich, można stwierdzić, że konfiguracja osobliwa x_o jest możliwa do uniknięcia, gdy istnieje takie pole $X_{i_1 \ldots i_{m+1}}$, że x_o nie jest stabilnym, w sensie Lapunowa, punktem równowagi układu dynamicznego $\dot{x} = X_{i_1 \ldots i_{m+1}}(x)$.

Postaci normalne

Postać normalna rozwiązania osobliwego kinematyki odwrotnej robota jest to odwzorowanie równoważne do oryginalnego, często określane jako bardziej „estetyczne", ale przede wszystkim umożliwiające rozwiązanie zadania oryginalnego. Dwa gładkie odwzorowania *f, g* nazywamy równoważnymi, jeśli istnieją dyfeomorfizmy φ, ψ takie, że poniższy diagram jest przemienny:

$$\begin{array}{ccc} \mathbb{R}^n & \xrightarrow{f} & \mathbb{R}^m \\ {\scriptstyle\varphi}\downarrow & & \downarrow{\scriptstyle\psi} \\ \mathbb{R}^n & \xrightarrow{g} & \mathbb{R}^m \end{array}$$

Powyższa równoważność nazywana jest RL (ang. right-left). Klasa odwzorowań równoważnych dla *f* może być reprezentowana, przez odpowiednio dobranego reprezentanta nazywanego postacią normalną. Postacie normalne opisane zostały m.in. w [71] oraz [72].

Dla kinematyki $k:\mathbb{R}^n \to \mathbb{R}^m,\ y=k(x),\ n \geq m$ definiujemy lokalne układy współrzędnych, zakładając, że x_0 jest konfiguracją regularną:

$$\xi=\varphi(x)=(k(x)-k(x_0), x_{m+1},\ldots,x_n)^T,\ \eta=\psi(y)=y-k(x_0)$$

Odwzorowanie φ jest lokalnym dyfeomorfizmem przekształcającym otoczenie x_0 na otoczenie $\mathbf{0}\in\mathbb{R}^n$, co wynika z założenia regularności konfiguracji x_0. Należy zauważyć, że ψ również jest dyfeomorfizmem. Podstawiając otrzymujemy:

$$\eta=k_0(\xi)=\psi\circ k\circ\varphi^{-1},\ k_0(\xi)=(\xi_1,\ldots,\xi_m)$$

Powyższe równanie definiuje przekształcenie sprowadzające kinematykę nieosobliwą do postaci liniowej projekcji $\boldsymbol{R}^n$ na $\boldsymbol{R}^m$, będącej jej postacią normalną.

Roboty stosowane w pracy

W pracy użyto trzy typy konfiguracji robotów o otwartym łańcuchu kinematycznym:

1. Manipulatora-robota w układzie Stanforda
2. Robota przemysłowego IRp-6, konfiguracja z pantografem
3. Robota przemysłowego Mitsubishi RV-2DF, konfiguracja antropomorficzna

Ich równania ruchu oraz Jakobiany przedstawiono w dodatku A.

2. Dynamika robotów o otwartym łańcuchu kinematycznym

Kinematyka opisuje geometryczne zależności położeń ramion robota w ruchu z pominięciem oddziaływań wywieranych przez masę oraz momenty bezwładności. Z kolei znajomość równań dynamiki pozwala na wyznaczenie sił i momentów sterujących manipulatorem-robotem.

Do wyznaczenia dynamiki są użyte równania Eulera-Lagrange'a, które sprowadzają zależności mechaniczne do postaci więzów holonomicznych. Równania Eulera-Lagrange'a w przypadku ogólnym mogą być wyprowadzone na podstawie energii w układzie manipulatora, zapisane w postaci równania Lagrange'a. Ze względu na uniwersalność oraz popularność poniższych wzorów, autor nie wprowadzał unikatowych oznaczeń a użył zgodnych z [43] oraz [67].

Równania Eulera-Lagrange'a w robotyce

W ogólnym przypadku równanie Lagrange'a jest dany wzorem:

$$L = E_k - E_p \tag{2.1}$$

Gdzie E_k to suma wszystkich energii kinetycznych w układzie, a E_p to suma wszystkich energii potencjalnych układu. Energia kinetyczna wyraża się równaniem:

$$E_K(q,\dot{q}) = \frac{1}{2}\dot{q}^T \sum_{i=1}^{n} \left[m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q) \right] \dot{q}$$

Gdzie m_i jest masą kolejnych członów, J_{v_i} jest jakobianem od prędkości liniowych dla środka masy członu i, J_{ω_i} jest jakobianem od prędkości obrotowych dla członu i, $R_i(q)$ jest macierzą orientacji i-tego członu względem podstawy, I_i jest macierzą inercji członu i:

$$I_i = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

E_K jest formą kwadratową, którą zapisać można jako:

$$E_K(q,\dot{q}) = \frac{1}{2}\dot{q}^T D(q) \dot{q}$$

Macierz $D(q)$ o wymiarze nxn jest symetryczna i dodatnio określona dla każdego $q \in \mathbb{R}^n$, nazywana jest macierzą inercji robota.

Przy założeniu, że energia potencjalna nie zależy od q', suma wszystkich energii potencjalnych w układzie wyraża się wzorem:

$$E_P(q) = \sum_{i=1}^{n} g^T r_{ci} m_i$$

gdzie g jest wektorem siły grawitacji, r_{ci} jest położeniem środka ciężkości członu względem podstawy, a m_i jest masą członu robota. Podstawiając (2.1) do równanie Eulera-Lagrange'a:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = \tau \tag{2.2}$$

gdzie τ to siły działające spoza układu lub inaczej sterowanie, otrzymujemy:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k}=\sum_i d_{kj}\ddot{q}_j+\sum_j \frac{d}{dt}d_{kj}\dot{q}_j$$

$$\frac{\partial L}{\partial q_k}=\frac{1}{2}\sum_{i,j}\frac{\partial d_{ij}}{\partial q_k}\dot{q}_i\dot{q}_j-\frac{\partial P}{\partial q_k}$$

Wprowadzając symbole Christoffela o postaci:

$$c_{ijk}:=\frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_i}+\frac{\partial d_{ki}}{\partial q_j}-\frac{\partial d_{ij}}{\partial q_k}\right)$$

Otrzymujemy uproszczoną postać równania:

$$\sum_i d_{kj}\ddot{q}_j+\sum_{i,j} c_{ijk}(q)\dot{q}_i\dot{q}_j-\frac{\partial P}{\partial q_k}=\tau_k,\quad k=1,\ldots,n \tag{2.3}$$

które składa się z trzech elementów: inercyjnego związanego z przyspieszeniami przegubów robota, sił odśrodkowych i Coriolisa związanych z iloczynem prędkości kątowych oraz zmiennych przegubowych nie związanych z prędkościami. W literaturze formuła (2.3) jest często przedstawiana w postaci macierzowej. Przyjmuje ona wówczas postać:

$$D(q)\ddot{q}+C(q,\dot{q})\dot{q}+g(q)=\tau \tag{2.4}$$

gdzie:

$$c_{kj}=\sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i=\sum_{i=1}^{n}\frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_i}+\frac{\partial d_{ki}}{\partial q_j}-\frac{\partial d_{ij}}{\partial q_k}\right)\dot{q}_i$$

Warto zaznaczyć, że dla specjalnego przypadku, gdy macierz *D* jest diagonalna i nie zależy od położeń przegubów, formuła (2.3) redukuje się do:

$$d_{kk}\ddot{q}-\frac{\partial P}{\partial q_k}=\tau_k,\quad k=1,\ldots,n$$

siły Coriolisa oraz odśrodkowe, są w tym przypadku zerowe.

Przykłady rozwiązania kilku zadań dynamiki zamieszczono w załączniku 1.

Wybrane właściwości równań robotyki

Można zauważyć, że pomiędzy macierzami *D* oraz *C* zachodzi relacja, której wynikiem jest macierz skośnie symetryczna:

$$N(q,\dot{q})=\dot{D}(q)-2C(q,\dot{q}) \tag{2.5}$$

Ponieważ elementy macierzy $D'(q)$, można zapisać jako:

$$\dot{d}_{kj}=\sum_{i=1}^{n}\frac{\partial d_{kj}}{\partial q_i}\dot{q}_i$$

to poszczególne elementy macierzy $N(q,q')$ wyrażają się wzorem:

$$\dot{n}_{kj}=\sum_{i=1}^{n}\left[\frac{\partial d_{ij}}{\partial q_k}-\frac{\partial d_{ki}}{\partial q_j}\right]\dot{q}_i$$

Ta informacja jest niezbędna do wykazania, że (2.4) jest układem pasywnym. Układ może zostać uznany za pasywny, gdy istnieje taki współczynnik *β*, że spełniona jest nierówność:

$$\int_0^T \langle \dot{q}(t),\tau(t)\rangle dt \geq -\beta, \quad \forall T>0 \qquad (2.6)$$

Powyższa całka jest całkowitą energią wygenerowaną przez układ w przedziale [0, *T*]. Pasywność układu oznacza, że całkowita energia wygenerowana przez układ jest ograniczona. Całkowitą energię w układzie można zapisać jako sumę energii kinetycznych oraz potencjalnych:

$$E=E_K(q,\dot{q})+E_p(q)=\frac{1}{2}\dot{q}^T D(q)\dot{q}+P(q)$$

Zmiana energii wyraża się jako pierwsza pochodna po czasie energii całkowitej:

$$\dot{E}=\dot{q}^T\left[\tau-C(q,\dot{q})-g(q)\right]+\frac{1}{2}\dot{q}^T\dot{D}(q)\dot{q}+\dot{q}^T\frac{\partial P}{\partial q}$$

Co po zgrupowaniu wyrażeń oraz podstawieniu $g(q)=\frac{\partial P}{\partial q}$ może być przedstawione jako:

$$\dot{E}=\dot{q}^T\tau+\frac{1}{2}\dot{q}^T\left[\dot{D}(q)-2C(q,\dot{q})\right]\dot{q}$$

Na podstawie wcześniejszej obserwacji o skośnej symetryczności wyrażenia (2.5) forma kwadratowa redukuje się co pozwala całkowitą zmianę energii w układzie przedstawić jako:

$$\dot{E}=\dot{q}^T\tau$$

Powyższa formuła może być wykorzystana do sprawdzenia warunku (2.6) w następujący sposób:

$$\int_0^T \langle \dot{q}(t),\tau(t)\rangle dt=E(T)-E(0)\geq -E(0)$$

Ponieważ całkowita energia układu nie może być ujemna, formuła (2.6) jest spełniona dla *β*=*E*(0).

Macierz inercji *D*(*q*) robota jest symetryczna i dodatnio określona, oznacza to, że dla stałego układu robota *q* wartości własne macierzy *D* można uporządkować w rosnącej kolejności:

$$0<\lambda_1(q)\leq\ldots\leq\lambda_n(q)$$

Macierz *D*(*q*) jest ograniczona przez skrajne wartości własne:

$$\lambda_1(q)I_{nxn}\leq D(q)\leq\lambda_n(q)I_{nxn}$$

Jeśli wszystkie połączenia w robocie są obrotowe, wówczas wszystkie funkcje w macierzy inercji są ograniczone. W takim przypadku uzyskuje się niezależne od położenia robota stałe ograniczające macierz inercji robota:

$$\lambda_d I_{nxn} \leq D(q) \leq \lambda_g I_{nxn} \leq \infty$$

Opory ruchu w układzie robotyki

Równanie (4) w swojej podstawowej formie nie uwzględnia sił tarcia. Te mogą być dodane w postaci członu:

$$F(\dot{q}) = F_v(\dot{q}) + F_d(\dot{q})$$

gdzie F_v jest macierzą oporu kinetycznego, a F_d jest tarciem ślizgowym. Tarcie ślizgowe w ogólnym przypadku jest wyrażone wzorem:

$$F_d(\dot{q}) = K_d \cdot sgn(\dot{q})$$

gdzie K_d jest macierzą diagonalną ze współczynnikami tarcia ślizgowego na diagonali.

Postać układu równań stanu

Równania dynamiki robota na ogół przedstawia się jako układ nieliniowych równań stanu:

$$\dot{x} = f(x, u, t)$$

Jeśli dla równania (4) spełniony jest warunek $\forall q, |D(q)| \neq 0$ postać tę można przekształcić do:

$$\ddot{q} = D^{-1}(q)\left[\tau - C(q, \dot{q})\dot{q} - g(q) - f(\dot{q})\right] \tag{2.7}$$

którą dalej po podstawieniu $\dot{x}_1 = \dot{q} = x_2$ modyfikuje się do postaci:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = D^{-1}(x_1)\left[\tau - C(x_1, x_2)x_2 - g(x_1)\right] \end{cases} \tag{2.8}$$

Używa się jej do konstrukcji regulatorów nadążnych przy zdefiniowanej trajektorii zadanej.

Linearyzacja w torze sprzężenia zwrotnego

W celu uzyskania postaci liniowej, zaproponowano wyjście układu y o postaci:

$$y = h(q) + s(t) \tag{2.9}$$

$h(q)$ jest funkcją przekształcającą zmienne stanu robota, a $s(t)$ jest wcześniej ustaloną funkcją czasu. Zadanie sterowania polega na sprowadzeniu wartości wyjścia y do zera. Funkcja sprowadzająca wyjście do zera może być zdefiniowana na wiele sposobów. Zmiana jej elementów pozwala na uzyskanie, różnych efektów. Przykładem może być nadążanie za zadaną trajektorią $q_d(t)$., wówczas y przyjmie formę:

$$y = -q(t) + q_d(t) = e(t)$$

W takim przypadku, funkcja wyjścia jest błędem nadążania za trajektorią.

Aby powiązać *y* z równaniami dynamiki robota, różniczkujemy (2.9) dwukrotnie po czasie. Po pierwszym różniczkowaniu otrzymujemy:

$$\dot{y}=\frac{\partial h}{\partial q}\dot{q}+\dot{s}:=J\dot{q}+\dot{s}$$

Gdzie *J* jest jakobianem funkcji transformującej zmienne przegubowe *h*(*q*), wyrażonym wzorem:

$$J(q)=\frac{\partial h}{\partial q}=\left[\frac{\partial h}{\partial q_1},\frac{\partial h}{\partial q_2},\ldots,\frac{\partial h}{\partial q_n}\right]$$

Po powtórnym zróżniczkowaniu otrzymujemy:

$$\ddot{y}=\dot{J}\dot{q}+J\ddot{q}+\ddot{s}$$

Podstawiając do ostatniego wyrażenia (7) uzyskujemy:

$$\ddot{y}=\dot{J}\dot{q}+JD^{-1}(q)\left[\tau-C(q,\dot{q})\dot{q}-g(q)\right]+\ddot{s} \qquad (2.10)$$

Przy zmiennych stanu dobranych tak jak w (8):

$$x=\begin{bmatrix}y\\ \dot{y}\end{bmatrix}$$

Równanie (10) może być zapisane w postaci:

$$\frac{d}{dt}\begin{bmatrix}y\\ \dot{y}\end{bmatrix}=\begin{bmatrix}0 & I\\ 0 & 0\end{bmatrix}\begin{bmatrix}y\\ \dot{y}\end{bmatrix}+\begin{bmatrix}0\\ I\end{bmatrix}u \qquad (2.11)$$

Gdzie *u* stanowi prawą strona równania (2.10). Równanie (2.11) ma klasyczną formę równań stanu. Aby wyznaczyć moment niezbędny do realizacji sterowania, wzór na sterowanie przekształca się do postaci:

$$\tau=DJ^{+}\left[u-\ddot{s}-\dot{J}\dot{q}\right]+C(q,\dot{q})\dot{q}+g(q)$$

W powyższy sposób, można otrzymywać momenty sterujące dla ramienia robota na podstawie regulatora minimalizującego wyrażenie (2.9). Jeśli regulator generujący sterowanie *u* przyjąć w formie klasycznego regulatora PD to u jest dane wzorem:

$$u(y)=-K_v\dot{y}(t)-K_p y(t)$$

Wyrażenie (2.9) w swojej formie umożliwia dokonywanie różnego rodzaju operacji na zmiennych przegubowych, m.in. przeliczania współrzędnych do innych układów odniesienia lub dodawanie niezbędnych przesunięć.

Dynamika członów wykonawczych

We wszystkich dotychczasowych rozważaniach zakładano, że sterowanie jest bezpośrednio przyłożone do członów robota i jest natychmiastowo przekładane na działanie mechanizmu. Od tej chwili równania dynamiki zostaną zmodyfikowane przez dodanie siłowników elektrycznych i wprowadzenie elastyczności przełożenia napędu.

Siłowniki elektryczne są opisane przez równanie:

$$
\begin{aligned}
&A_m\ddot{q}_m+B_m\dot{q}_m+F_m+R_m\tau=K_m v\\
&A_m=diag\{A_{m_i}\}\\
&B_m=diag\{B_{m_i}+K_{b_i}K_{m_i}/R_{a_i}\}\\
&R_m=diag\{r_i\}\\
&K_m=diag\{K_{m_i}/R_{a_i}\}\\
&F_m=vec\{F_{m_i}\}
\end{aligned}
\qquad (2.12)
$$

Gdzie *i* oznacza *i*-ty silnik z momentem bezwładności A_m, ze stałą tłumienia B_m, ze stałą elektromechaniczną K_b, ze stałą mechaniczną K_m z opornością uzwojenia *Ra*, z napięciem sterującym *v* przyłożonym do silnika i z tarciem F_m wewnątrz silnika. Dla uproszczenia pominięto indukcyjność uzwojeń silnika, które w mechanizmach o niskiej mocy jest pomijalna.

Należy zauważyć, że w większości przypadków, w robotach są stosowane reduktory, które zmieniają moment wyjściowy, oraz ilość obrotów silnika. Jako q_m oznaczono położenie wału silnika przed reduktorem, natomiast *q* reprezentuje zmienną przegubową za przekładnią. Zależność pomiędzy *q* i q_m wyraża się:

$$q=R_m q_m$$

Gdzie R_m to diagonalna macierz przełożeń poszczególnych siłowników. Po podstawieniu (2.12) do (2.4) i uwzględnieniu sił oporu, równanie dynamiki manipulatora-robota, uwzględniające dynamikę elektrycznych członów wykonawczych wyraża się wzorem:

$$(A_m+R_m^2D)\ddot{q}+(B_m+R_m^2C)\dot{q}+(RF_m+R_m^2F)+R_m^2g=R_mK_mv$$

Ta dynamika zakłada, że człony oraz elementy wykonawcze połączone są ze sobą w sposób sztywny. W rzeczywistości często zdarza się, że przekładnie wprowadzają sprężystość do układu. Elastyczność modeluje się przy użyciu równań sprężyny, gdzie jako wychylenie sprężyny traktuje się różnice między położeniem przed i za reduktorem:

$$
\tau=B_s(\dot{q}_m-\dot{q})+K_s(q_m-q)\quad \tau=B_s(\dot{q}_m-\dot{q})+K_s(q_m-q)\quad
\begin{aligned}
&B_s=diag\{b_{s_i}\}\\
&K_s=diag\{k_{s_i}\}
\end{aligned}
\qquad (2.13)
$$

gdzie b_s jest współczynnikiem tłumienia, a k_s jest współczynnikiem sprężystości sprężyny w modelu zastępczym przekładni.

Formalizm Newtona-Eulera

Formalizm Newtona-Eulera jest alternatywnym podejściem do zagadnienia wyznaczania dynamiki manipulatora. W swojej istocie daje on takie same wyniki jak przedstawiony uprzednio formalizm Eulera-Lagrange'a, jednak sposób otrzymania rozwiązania jest inny. W formalizmie Eulera-Lagrange'a równania dynamiki generuje się przy pomocy różniczkowania funkcji Lagrange'a, która jest w istocie różnicą pomiędzy energią kinematyczną i potencjalną w układzie. Robot jest przez tę metodę traktowany jako jedna całość. W formalizmie Newtona-Eulera dynamika powstaje przez analizę powiązań dynamicznych pomiędzy poszczególnymi członami robota w sposób rekurencyjny. Oba formalizmy dają tożsame wyniki.

Formalizm Newtona-Eulera opiera się na trzech podstawowych prawach:

1. Każdej akcji towarzyszy reakcja o takiej samej wartości i kierunku oraz przeciwnym zwrocie;
2. Szybkość zmiany siły liniowej równa się całkowitej sile przyłożonej do obiektu;
3. Szybkość zmiany momentu obrotowego równa się całkowitemu momentowi przyłożonemu do obiektu.

Punkty 2 oraz 3 można w prosty sposób przedstawić w postaci poniższych równań:

$$\frac{d(mv)}{dt}=\boldsymbol{f}\ ,\ \frac{d(I_0\omega_0)}{dt}=\boldsymbol{\tau_0}\ .$$

Należy zauważyć, że o ile równanie dla ruchu liniowego jest zrozumiałe o tyle równanie dla ruchu obrotowego wprowadza pewną trudność. I_0 jest macierzą bezwładności członu w odniesieniu do jego środka ciężkości, a ω_0 jest prędkością obrotową członu. Jeśli człon zostanie umieszczony w sztywnej zewnętrznej ramie wówczas jego moment bezwładności będzie równy:

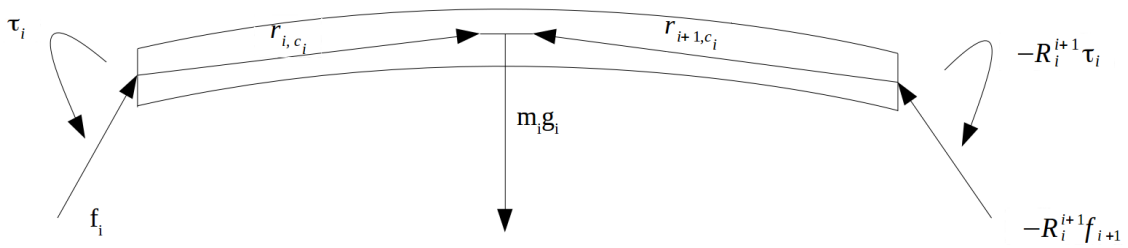
$$I_0=RIR^T$$

Gdzie I jest niezmiennym momentem członu a R jest macierzą przejścia do układu odniesienia ramy. Oznacza to, że I_0 może być zmienne w czasie. Jednym z możliwych sposobów rozwiązania tego problemu jest zapisanie prędkości obrotowej ω bezpośrednio w odniesieniu do sztywnej ramy:

$$I\dot{\omega}+\omega\times(I\omega)=\boldsymbol{\tau}$$

wówczas I jest niezmiennym momentem bezwładności wobec ramy odniesienia, a τ jest całkowitym momentem oddziaływującym na człon.

Na rysunku 2.1 przedstawiono schematyczny obraz pojedynczego i-tego członu, który jest wykorzystywany podczas wyznaczania dynamiki w systemie Newtona-Eulera. Poszczególne oznaczenia sił to: g_i przyspieszenie ziemskie względem ramy i, f_i siła wywierana przez człon i-1 na człon i, τ_i moment obrotowy wywierany przez człon i-1 na człon i a R_i^{i+1} jest macierzą rotacji członu i+1 do członu i, oznaczenia własności fizycznych członu to: m masa, I_i macierz inercji i-tego członu, $r_{i,ci}$ wektor pomiędzy przegubem i a środkiem ciężkości i, $r_{i+1,ci}$ wektor pomiędzy przegubem i+1 a środkiem ciężkości członu i.



Rys. 2.1 Schemat sił działających na pojedynczy człon w formalizmie Newtona-Eulera

Przy pomocy powyższego rysunku oraz na podstawie pierwszego prawa formalizmu NE możemy wyznaczyć równanie siły liniowej działającej na człon i

$$f_i - R_i^{i+1} f_{i+1} + m_i g_i = m_i a_{c,i}\,, \tag{2.14}$$

gdzie $a_{c,i}$ jest przyspieszeniem liniowym środka ciężkości *i*-tego członu. W podobny sposób możemy zapisać równanie dla momentów siły

$$\tau_i - R_i^{i+1}\tau_{i+1} + f_i \times r_{i,c_i} - (R_i^{i+1} f_{i_1}) \times r_{i+1,c_i} = \alpha_i + \omega_i \times (I_i \omega_i)\,, \tag{2.15}$$

gdzie ω_i to prędkość obrotowa członu *i*, a α_i to przyśpieszenie kątowe członu *i*.

Znając równania (2.14) oraz (2.15) można przejść do określenia sił działających na poszczególne człony robota. Aby tego dokonać należy przyjąć, że człon *n*+1 nie istnieje, a zatem $f_{n+1} = 0$ oraz $\tau_{n+1} = 0$, następnie podstawiając *i*=*n*, *n*-1, … ,1 można uzyskać równania na siły oraz momenty wewnątrz manipulatora.

Następnym krokiem jest uzyskanie zależności pomiędzy zmiennymi przegubowymi q oraz ich pochodnymi a zmiennymi $a_{c,i}$, ω_i oraz α_i. W tym celu procedurę iteracyjną należy zacząć od początku łańcucha kinematycznego robota, podstawić odpowiednio: *i*=1, 2, …, n, oraz założyć, że: $a_{c,0} = 0$, $\omega_0 = 0$ oraz $\alpha_0 = 0$.

$$\omega_i = (R_{i-1}^i)^T \omega_{i-1} + b_i \dot{q}_i\,,$$
$$\alpha_i = (R_{i-1}^i)^T \alpha_{i-1} + b_i \ddot{q}_i + \omega_i \times b_i \dot{q}_i\,,$$

gdzie:

$$b_i = R_0^i z_{i-1}$$

Ostatnim krokiem jest wyznaczenie przyspieszeń liniowych dla środków ciężkości kolejnych członów:

$$a_{c,i} = (R_{i-1}^i)^T a_{e,i-1} + \dot{\omega}_i \times r_{i,c_i} + \omega_i + \omega_i \times (\omega_i \times r_{i,c_i})$$

gdzie $a_{e,i}$ jest przyspieszeniem liniowym końcówki *i*-tego członu i może być wyrażone przez podstawienie $r_{i,i+1}$ zamiast $r_{i,ci}$:

$$a_{e,i} = (R_{i-1}^i)^T a_{e,i-1} + \dot{\omega}_i \times r_{i,i+1} + \omega_i + \omega_i \times (\omega_i \times r_{i,i+1})$$

W powyższy sposób uzyskać można wszystkie elementy opisujące dynamikę manipulatora-robota przy pomocy formalizmu Newtona-Eulera.

3. Aproksymacje trajektorii robota

Planowanie trajektorii jest podstawowym zadaniem robotyki. Kocówka robota porusza się zawsze po ustalonej wcześniej drodze, zależnej od zadania wykonywanego przez robota przemysłowego. Kiedy zadanie jest znane można wyznaczyć wybraną trajektorię poruszania się końcówki. Istnieje wiele możliwych trajektorii do wyboru. Niektóre z nich łączą punkty w przestrzeni, w taki sposób, aby zminimalizować czas ruchu pomiędzy nimi, inne podążają za ustalonymi wzorcami geometrycznymi [76], a jeszcze inne pozwalają na minimalizację szarpnięcia [55]. Można wyróżnić kilak rodzajów aproksymacji trajektorii. Podstawową jest aproksymacja wielomianowa PTP. Wzdłuż trajektorii wyznaczane są punkty węzłowe, pomiędzy którymi wyznacza się przejścia przy pomocy funkcji w postaci wielomianu. Następnie pokazane są aproksymacje sklejane, najpierw klasyczna a następnie spliny typu B. W literaturze można spotkać również inne aproksymacje takie jak w [59].

Rozdział dotyczy planowania różnych trajektorii robota. Omawia się następujące ruchy: prostoliniowy oraz po zadanym odcinku łuku. Rozważa się sposoby na znajdowanie trajektorii łączących punkty w przestrzeni optymalnie w czasie ze względu na kinematykę manipulatora. [12].

Aproksymacja wielomianowa PTP

Najprostszą metodą aproksymacji położenia przestrzeni trójwymiarowej w przegubowej jest aproksymacja wielomianowa typu point-to-point (ang. od punkt do punktu). Opiera się ona na wyznaczaniu położenia punktów pośrednich, przez znaleziony wielomian aproksymacyjny, przy założeniu, że końcówka ramienia znajduje się w zatrzymaniu na końcu oraz początku odcinka. Omówione zostaną dwa przybliżenia z tej grupy. Podstawową różnicą pomiędzy nimi będzie różny stopień znalezionego wielomianu.

Pierwszym przybliżeniem jest aproksymacja wielomianem stopnia trzeciego. Równania definiujące położenie kiści robota definiowane są jako:

$$q(t)=a_0+a_1t+a_2t^2+a_3t^3$$
$$\dot{q}(t)=a_1+2a_2t+3a_3t^2$$
$$\ddot{q}(t)=2a_2+6a_3t$$

Współczynniki wielomianu można obliczyć przy pomocy równania macierzowego:

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} q_0 \\ V_0 \\ q_f \\ V_f \end{bmatrix}$$

Naturalnym rozwinięciem powyższej aproksymacji jest aproksymacja wielomianowa typu PTP stopnia piątego. Dzięki zwiększeniu stopnia wielomianu, druga pochodna położenia końcówki robota jest funkcją ciągłą. Równania dla aproksymacji PTP stopnia piątego są postaci:

$$q(t)=a_0+a_1t+a_2t^2+a_3t^3+a_4t^4+a_5t^5$$
$$\dot{q}(t)=a_1+2a_2t+3a_3t^2+4a_4t^3+5a_5t^4$$
$$\ddot{q}(t)=2a_2+6a_3t+12a_4t^2+20a_5t^3$$

Równanie macierzowe dla wyznaczania współczynników wielomianu pokazano poniżej:

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} q_0 \\ V_0 \\ a_0 \\ q_f \\ V_f \\ a_f \end{bmatrix}$$

Można zauważyć, ze dla równo podzielonego odcinka, względem czasu, postać macierzowa uprasza się, co pozwala zmniejszyć ilość obliczeń niezbędnych do wyznaczenia współczynników wielomianu. Uproszczone równanie macierzowe ma postać:

$$A=\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & dt & dt^2 & dt^3 & dt^4 & dt^5 \\ 0 & 1 & 2dt & 3dt^2 & 4dt^3 & 5dt^4 \\ 0 & 0 & 2 & 6dt & 12dt^2 & 20dt^3 \end{bmatrix}, a_i=\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}, b_i=\begin{bmatrix} q_0 \\ V_0 \\ a_0 \\ q_f \\ V_f \\ a_f \end{bmatrix}$$

Jeżeli założymy, że cała trajektoria robota, zostanie podzielona na *n* odcinków, współczynniki wszystkich wielomianów można oznaczyć przy pomocy macierzy diagonalno-blokowej. Równanie współczynników dla całego toru ma wówczas postać:

$$\begin{bmatrix} A & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & A & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \mathbf{0} & \ddots & \mathbf{0} & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & A & \mathbf{0} \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & A \end{bmatrix}_{nxn} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Aproksymacja funkcjami sklejanymi

Jedną z cech aproksymacji typu punkt do punktu jest znajomość prędkości, dla aproksymacji wyższego rzędu oraz znajomość przyspieszenia, w punktach pośrednich aproksymacji. Wprowadza to utrudnienie aplikacyjne, gdy robot podąża za trajektorią złożoną z odcinków, pomiędzy którymi nie powinno mieć miejsca zatrzymanie końcówki. Grupą aproksymacji, które pozwalają na złożenie wieloodcinkowej trajektorii, bez konieczności wiedzy o dalszych pochodnych ruchu w punktach węzłowych, są aproksymacje typu sklejanego (ang. spline). [1], [28], [29], [45], [54].

Aproksymacja funkcjami sklejanymi jest interpolacją wielomianową. Stopień wielomianu definiuje jednocześnie ilość niezbędnych danych wejściowych dla algorytmu oraz ilość ciągłych pochodnych interpolacji. Dla tego typu interpolacji, pochodne w punktach węzłowych są ciągłe i obliczane podczas wyznaczania współczynników wielomianów sklejanych.

Funkcje sklejane trzeciego rzędu

Gdy jako funkcję sklejaną użyje się wielomianu stopnia trzeciego, wówczas równanie trajektorii na odcinku pomiędzy punktami węzłowymi jest postaci:

$$q(X(t_i)) = y_i$$
$$q(X(t_i)) = a_i + b_i t + c_i t^2 + d_i t^3 \quad \tau = t - t_i$$

Jeśli czas trwania odcinka zdefiniujemy jako:

$$h_i = t_{i+1} - t_i$$

to do wyznaczenia współczynników wszystkich wielomianów wykorzystanych w aproksymacji można zastosować poniższe formuły:

$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(2c_i - c_{i+1}), \quad i \in \langle 1, n-1 \rangle$$

$$\begin{bmatrix} 2 & \frac{h_{i+1}}{h_{i+1}+h_i} & \cdots & 0 \\ \frac{h_i}{h_{i+1}+h_i} & 2 & \ddots & 0 \\ \vdots & \ddots & 2 & \frac{h_{n-1}}{h_{n-1}+h_{n-2}} \\ 0 & 0 & \frac{h_{n-2}}{h_{n-1}+h_{n-2}} & 2 \end{bmatrix} \begin{bmatrix} c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} a_3 - a_2 h_2 - \frac{a_2 - a_1}{h_1} \\ \vdots \\ \frac{a_n - a_{n-1}}{h_{n-1}} - \frac{a_{n-1} - a_{n-2}}{h_{n-2}} \end{bmatrix}$$

$$d_i = \frac{1}{3h_i}(c_{i+1} - c_i), \quad i \in \langle 1, n-1 \rangle$$

Wielomian stopnia trzeciego gwarantuje ciągłość trajektorii oraz jej pierwszych dwóch pochodnych.

Funkcje sklejane piątego rzędu

Użycie w interpolacji typu sklejanego wielomianów stopnia piątego pozwala na zwiększenie regularności funkcji aproksymacyjnej do rzędu czwartego. Dla końcówki manipulacyjnej oznacza to, że zachowana zostanie płynność zmiany przyspieszenia podczas całego okresu trwania ruchu.

Równanie ruchu dla poszczególnych odcinków trajektorii dla tego typu interpolacji jest postaci:

$$q(X(t_i))=y_i$$
$$q(X(t_i))=a_i+b_i t+c_i t^2+d_i t^3+e_i t^4+f_i t^5 \tau=t-t_i$$

Wyznaczenie współczynników wielomianów jest w tym przypadku bardziej złożone niż dla interpolacji stopnia trzeciego. Aby poprawnie wyznaczyć wszystkie stałe konieczna jest znajomość współczynnika b_i, co w przypadku manipulatora oznacza prędkość w punkcie węzłowym. Prędkość może zostać wyznaczona przy pomocy równania jakobianowego:

$$b_i=J_v^{-1}\vec{k}, i\in\langle 1,n\rangle$$

Przy takim samym oznaczeniu czasów trwania odcinka ruchu jak dla interpolacji stopnia trzeciego (h_i), równania dla współczynników są następującej postaci:

$$Mc_{2..n-1}=Y$$

gdzie:

$$M=\begin{bmatrix} 3\left(\frac{1}{h_{i-1}}+\frac{1}{h_i}\right) & \frac{-1}{h_i} & \cdots & 0 \\ \frac{-1}{h_{i-1}} & 3\left(\frac{1}{h_{i-1}}+\frac{1}{h_i}\right) & \ddots & 0 \\ \vdots & \ddots & 3\left(\frac{1}{h_{n-3}}+\frac{1}{h_{n-2}}\right) & -\frac{1}{h_{n-1}} \\ 0 & 0 & \frac{-1}{h_{n-2}} & 3\left(\frac{1}{h_{n-2}}+\frac{1}{h_{n-1}}\right) \end{bmatrix}$$

$$Y_i=10\left[\frac{a_{i+2}-a_{i+1}}{h_{i+1}^3}-\frac{a_{i+1}-a_i}{h_i^3}\right]+4\left[\frac{b_i}{h_i^2}-\frac{3}{2}\left(\frac{1}{h_{i+1}^2}-\frac{1}{h_i^2}\right)b_{i+1}-\frac{b_{i+2}}{h_{i+i}^2}\right], i\in\langle 2,n-1\rangle$$

Pozostałe współczynniki są wyznaczane przy pomocy równań:

$$d_i=\frac{10}{h_i^3}(a_{i+1}-a_i)-\frac{2}{h_i^2}(2b_{i+1}+3b_i)+\frac{1}{h_i}(c_{i+1}-3c_i), i\in\langle 1,n-1\rangle$$
$$d_n=d_{n-1}-\frac{2}{h_{n-1}^2}(b_n-b_{n-1})+\frac{2}{h_{n-1}}(c_n-c_{n-1})$$
$$e_i=\frac{1}{2h_i^3}(b_{i+1}-b_i)-\frac{1}{h_i^2}c_i+\frac{1}{4h_i}(d_{i+1}+5d_i), i\in\langle 1,n-1\rangle$$
$$f_i=\frac{1}{10h_i^3}(c_{i+1}-c_i-3d_i h_i-6e_i h_i^2), i\in\langle 1,n-1\rangle$$

Funkcje B-sklejane

Krzywe B-sklejane są szeroko stosowane w aproksymacji kształtów w grafice komputerowej oraz rozpoznawaniu obrazów. Od kilkudziesięciu lat prowadzi się prace nad wykorzystaniem tych uniwersalnych metod aproksymacyjnych w robotyce. [19], [20], [21], [31], [64]. Znalazły on również zastosowanie w niezwykle skomplikowanych przypadkach kinematycznych takich jak model robot humanoidalnego [60].

Krzywe B-sklejane są to ,,sklejone'' wielomiany niskiego stopnia, zwykle nie większego niż trzeci. Są one regularne w rzędzie zależnym od stopnia wielomianu.

Podstawowy wzór opisujący aproksymowaną drogę efektora ma postać:

$$p(t)=\sum_{i=0}^{m-n-1} p_i N_i^n(t), \quad t\in\langle u_n, u_{m-1}\rangle$$

gdzie $N_i^n(t)$ dane jest wzorem Mansfielda-de Boora Coxa:

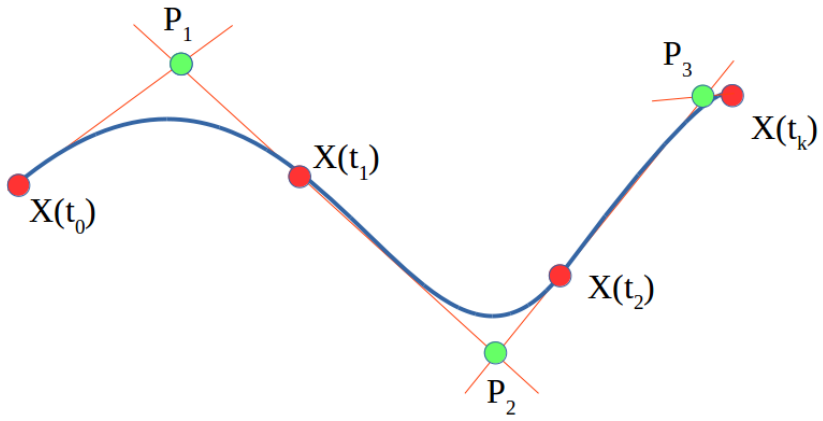
$$N_i^0(t)=\begin{cases}1 & dla \quad t\in[u_i,u_{i+1})\\ 0 & dla \quad t\notin[u_i,u_{i+1})\end{cases}$$

$$N_i^n(t)=\frac{t-u_i}{u_{i+n}-u_i}N_i^{n-1}(t)+\frac{u_{i+n+1}-t}{u_{i+n+1}-u_{i+1}}N_{i+1}^{n-1}(t), \quad dla \quad n>0$$

Dla krzywej Beziera drugiego rzędu powyższy wzór można zapisać jako:

$$p(t)=\begin{bmatrix}P_1 & P_2 & P_3\end{bmatrix}\begin{bmatrix}1 & -2 & 1\\ -2 & 2 & 0\\ 1 & 0 & 0\end{bmatrix}\begin{bmatrix}t^2\\ t\\ 1\end{bmatrix}$$

W celu wyznaczenia krzywych Beziera drugiego rzędu dla każdego odcinka niezbędne są trzy punkty. Dwa punkty, pierwszy oraz końcowy fragmentu trajektorii, są węzłami interpolacji. Problemem jest wyznaczenie trzeciego punktu generującego krzywiznę funkcji Beziera. Aby go uzyskać konieczne jest użycie stycznych do trajektorii w punktach węzłowych. Punkt przecięcia stycznych w punktach krańcowych segmentu jest jednocześnie trzecim punktem kontrolnym. Należy tutaj zauważyć, że współczynniki kierunkowe powyższych prostych są jednocześnie prędkością przegubu robota w punkcie węzłowym. W celu zapewnienia ciągłości pierwszej pochodnej, pośrednie punkty kontrolne kolejnych segmentów (P_i, P_{i+1}) powinny leżeć na jednej prostej. Użycie stycznej spełnia ten warunek. Na rysunku 3.1 przedstawiono przykład generacji pośrednich punktów kontrolnych.



Rys. 3.1 Przykład generacji punktów pomocniczych dla B-splinu drugiego rzędu. Na czerwono zaznaczono punkty węzłowe rozmieszczone równomiernie, na zielono zaznaczono punkty pomocnicze wygenerowane na przecięciach stycznych do trajektorii w punktach węzłowych

Równanie na położenie punktu pośredniego segmentu *i*-tego:

$$P_i = \begin{cases} \dfrac{X(t_i) - X(t_{i-1}) - v(t_i)t_i + v(t_{i-1})t_{i-1}}{v(t_{i-1}) - v(t_i)} \; dla \; v(t_{i-1}) \neq v(t_i) \\ \dfrac{X(t_i) - X(t_{i-1})}{2} \; dla \; v(t_{i-1}) = v(t_i) \end{cases}$$

W przeciwieństwie do interpolacji wielomianowych opisanych poprzednio, punkty węzłowe krzywych B-sklejanych nie powinny być wybierane w sposób równomierny. Aby uzyskać najlepsze dopasowanie do zamierzonej trajektorii, powinny zostać zastosowane odpowiednie algorytmy poszukujące punkty kontrolne krzywej.

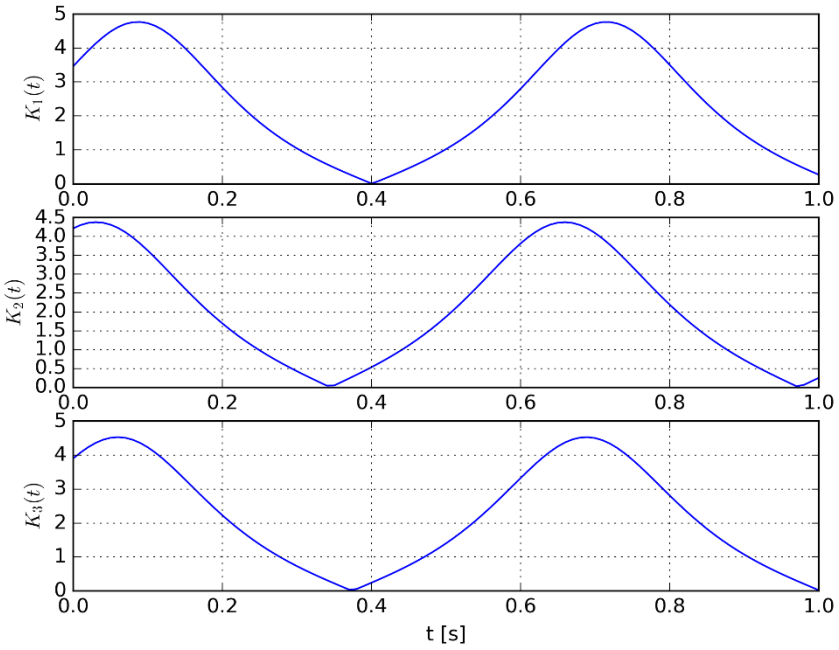
Jedną z możliwości doboru punktów na krzywej, jest badanie przebiegu wyznacznika krzywizny krzywej (ang. curvature). W ogólnym przypadku jest on zapisywany jak poniżej:

$$K = \frac{|det(\gamma', \gamma'')|}{\|\gamma'\|^3},$$

gdzie γ jest parametrycznym równaniem krzywej. Dla krzywej jednowymiarowej mamy:

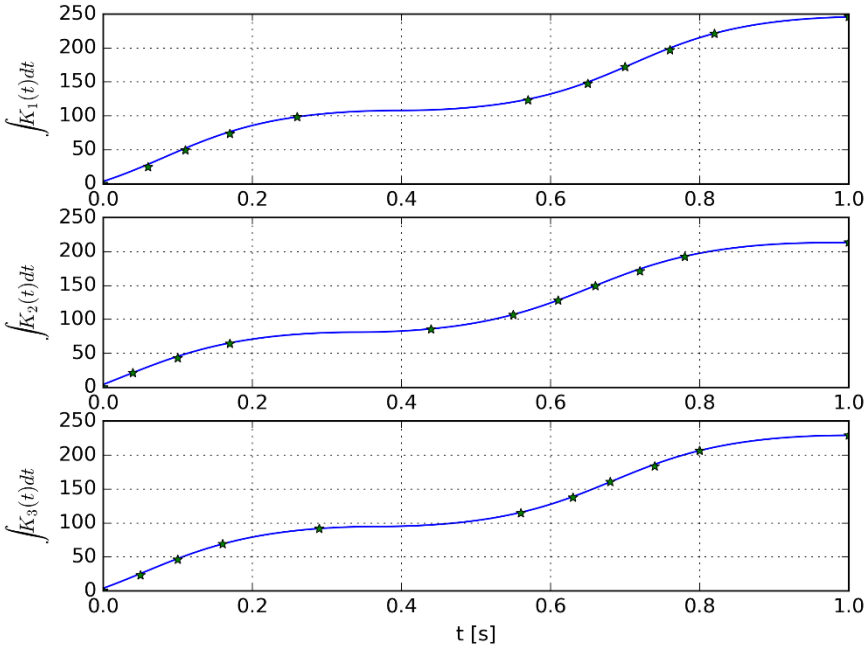
$$K = \frac{|\ddot{y}|}{\sqrt{1+\dot{y}^3}}$$

W pracy [72] przedstawiono dogłębną analizę zagadnienia doboru punktów kontrolnych na podstawie przebiegu funkcji krzywizny. Na rysunku 3.2 przedstawiono, jak wygląda zmiana krzywizny w czasie dla krzywej w kształcie elipsy o parametrach $X_0 = [0.6, 0, 0.6]^T$, $U = [1, 1, 1]^T$, $V = [1.5, 0.5, 1]^T$, $a = 0.3$, $b = 0.1$.



Rys. 3.2 Zmienność krzywizny toru w kształcie elipsy w czasie

Na rysunku 3.3 przedstawiono, całkę funkcji krzywizny toru. Można zauważyć, że przebieg jest niemalejący. Na podstawie krzywizny toru można wyznaczyć położenia punktów węzłowych. Na rysunku zaznaczono dziesięć węzłów rozłożonych równomiernie w całym zakresie krzywizny.



Rys. 3.3 Punkty węzłowe rozmieszczone równomiernie względem zmiany krzywizny

Przykłady aproksymacji

Aby zilustrować działanie opisanych algorytmów przedstawiono, jak zachowują się poszczególne aproksymacje dla robota IRP, zainstalowanego w Katedrze Automatyki i Robotyki, Wydziału Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej AGH.

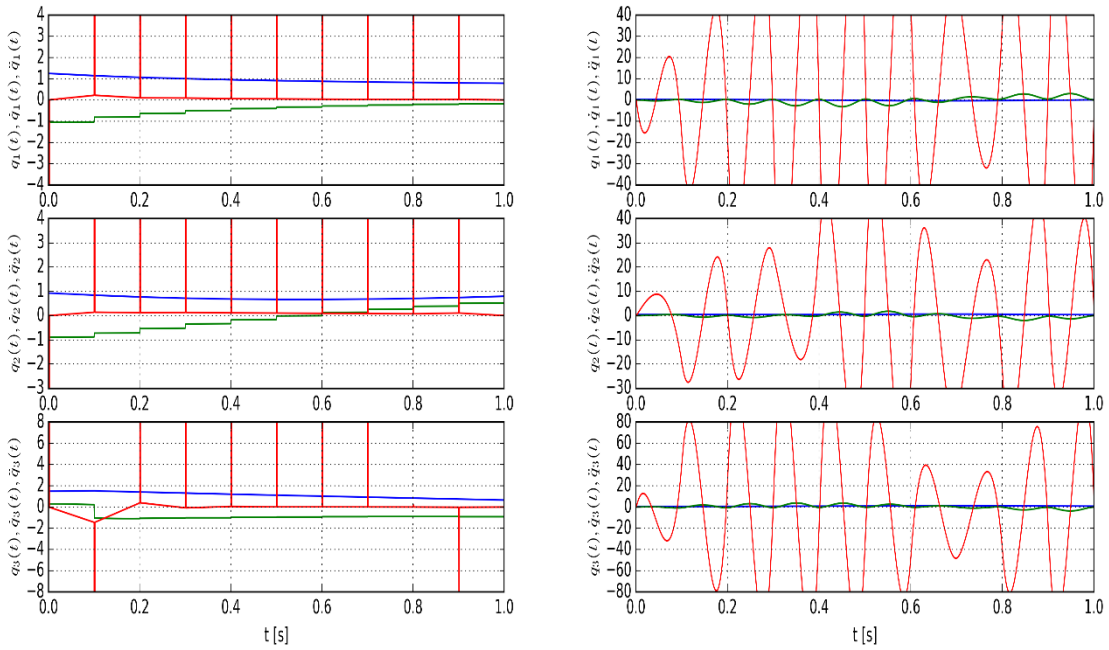
Ruch po prostej

Poniżej opisano trajektorię prostoliniową. Trajektoria tego typu ma wiele zastosowań przemysłowych, takich ja k prowadzenie aerografu lub poruszanie się przez wąskie przejścia. Trajektorie prostoliniowe są również wyzwaniem, ponieważ ich reprezentacja w przestrzeni przegubowej robota, jest złożona dla większości robotów. Trajektoria prostoliniowa wyraża się wzorem parametrycznym:

$$l: X(t)=\vec{k}\cdot t, \vec{k}=\frac{1}{T_k}(X_{end}-X_{start}), t\in\langle 0, T_k\rangle ,$$

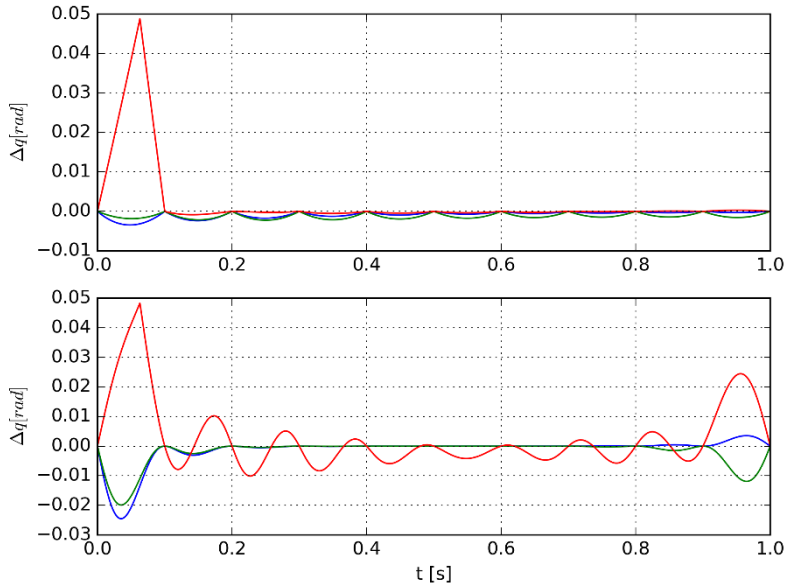
gdzie k jest wektorem współczynników kierunkowych, uzyskanym poprzez odjęcie zamierzonego położenia końcowego X_{end} oraz położenia początkowego X_{start}, a następnie podzielenie różnicy przez skalar T_k będący czasem trwania ruchu. Na rysunku 3.4 przedstawiono w jaki sposób zachowuje się trajektoria prostoliniowa aproksymowana przy pomocy dwóch metod: splinu trzeciego i piątego stopnia. W obu wypadkach użyto 11 węzłów

interpolacyjnych rozłożonych równomiernie w czasie trwania ruchu. Trajektoria definiowana jest przez następujące parametry: $X_{start}=[0.1, 0.3, 0.3]^T$, $X_{end}=[0.6,0.6,0.6]^T$ oraz $T_k=1.0$.



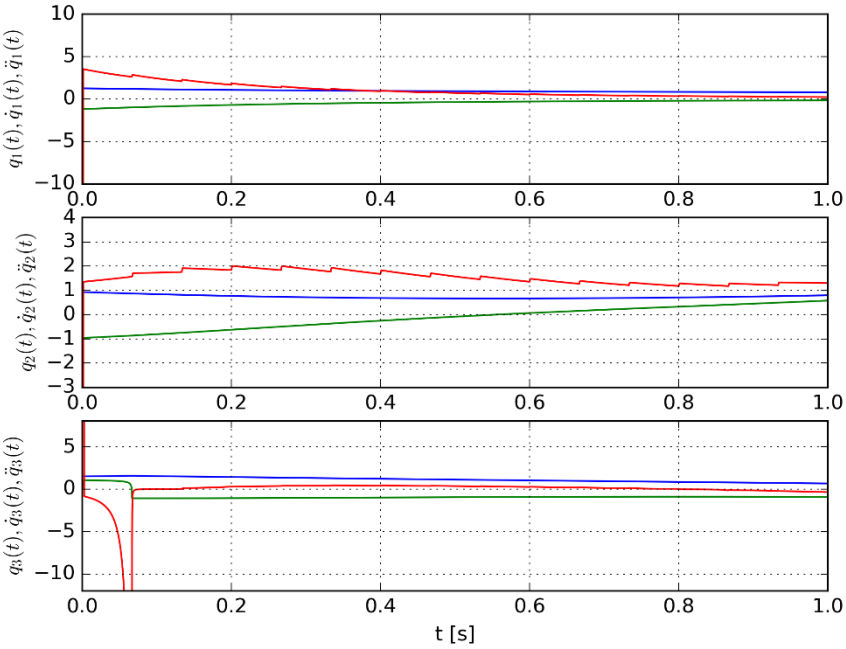
Rys. 3.4 Porównanie aproksymacji splinami trzeciego rzędu (po lewej) i 5 rzędu (po prawej)

Na rysunku 3.5 przedstawiono błąd towarzyszący interpolacjom trzeciego i piątego rzędu. Można zaobserwować, że początek ruchu oraz jego koniec dla obu typu aproksymacji jest obarczony znaczącym błędem. Wynika to z konieczności rozpędzenia i wyhamowania końcówki robota. Dodatkowo interpolacja piątego rzędu jest obarczona większymi błędami na odcinakach między węzłami interpolacyjnymi.

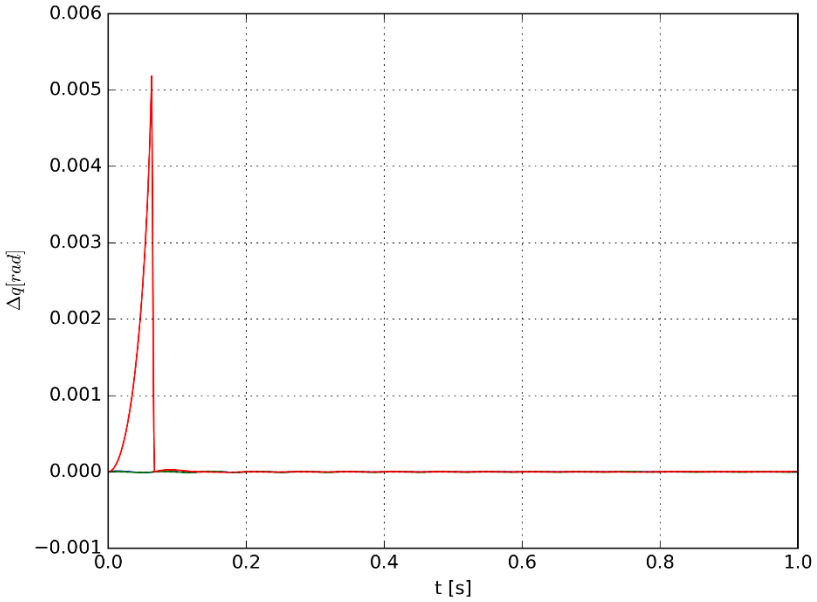


Rys. 3.5 Porównanie błędów aproksymacji splinami 3 rzędu (u góry) i 5 rzędu (u dołu)

Rysunki 3.6 oraz 3.7 przedstawiają kolejno działanie aproksymacji sklejanej oraz błędu jej towarzyszącego. Istotną informacją jest, że pomimo zmniejszenia szarpnięcia oraz ograniczenia drugiej pochodnej, błąd podczas ruchu nie wzrósł jak w wypadku interpolacji wielomianowej piątego rzędu.



Rys. 3.6 Aproksymacja B-splinami drugiego rzędu



Rys. 3.7 Błąd aproksymacji B-splinem drugiego rzędu

Ruch po łuku

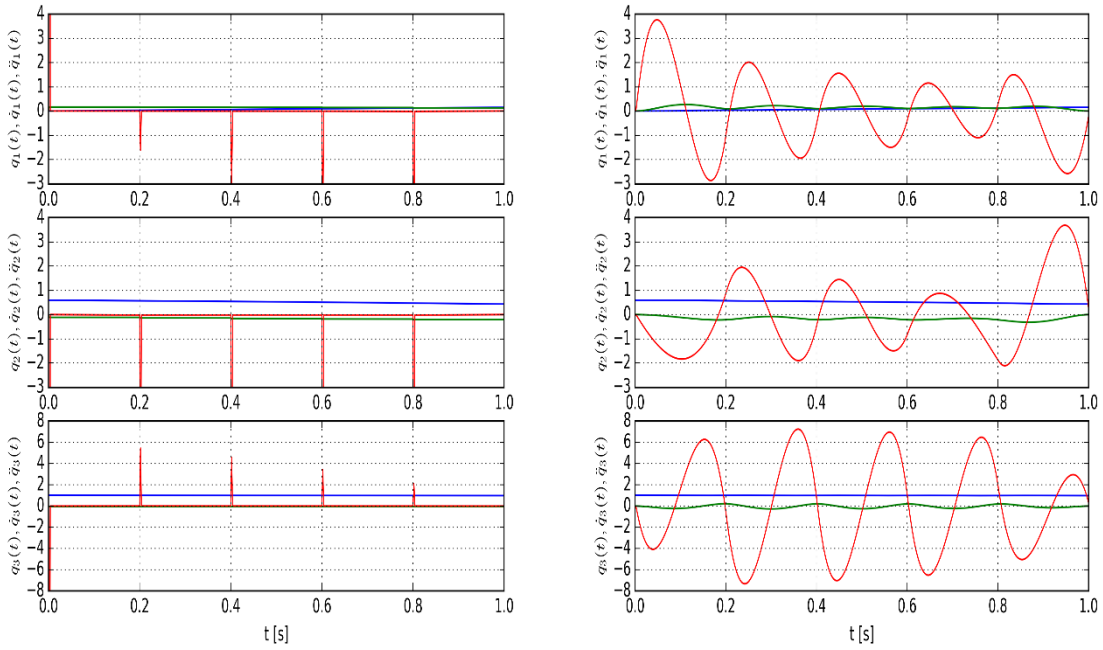
Ruch po łuku jest skomplikowany w porównaniu z ruchem po prostej. Literatura wyróżnia wiele rodzajów krzywych w przestrzeni, które można zakwalifikować jako łuk. W rozdziale poniżej przedstawiono opisy dwóch podstawowych typów łuku: helisy oraz elipsy.

Helisa - linia śrubowa

Parametryczne równanie linii śrubowej:

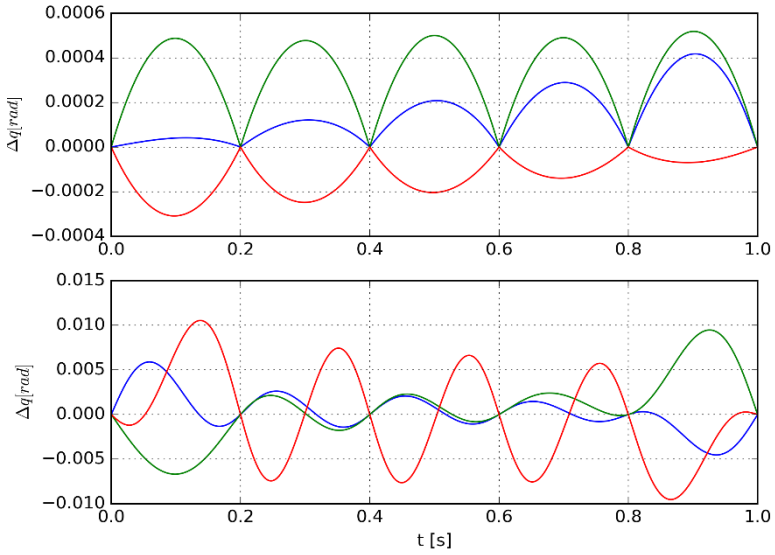
$$p: X(t) = \begin{bmatrix} a\cos(t) \\ b\sin(t) \\ ct \end{bmatrix} + X_0, \quad t \in \langle 0, T_k \rangle ,$$

gdzie Współczynniki trajektorii użytej do przedstawienia działania aproksymacji to: $a = b = 0.1m$, $c = 0.05m$, $X_0 = [0.5, 0, 0.5]^T$ oraz $T_k = 1s$. Interpolacje były uzyskiwane na podstawie 6 węzłów interpolacyjnych równomiernie rozłożonych w czasie. Na rysunku 3.8 przedstawiono porównanie działania interpolacji funkcji różnymi rzędami wielomianu. Można zaobserwować, że wielomian piątego rzędu nie posiada nieciągłości drugiej pochodnej zmiennych przegubowych.



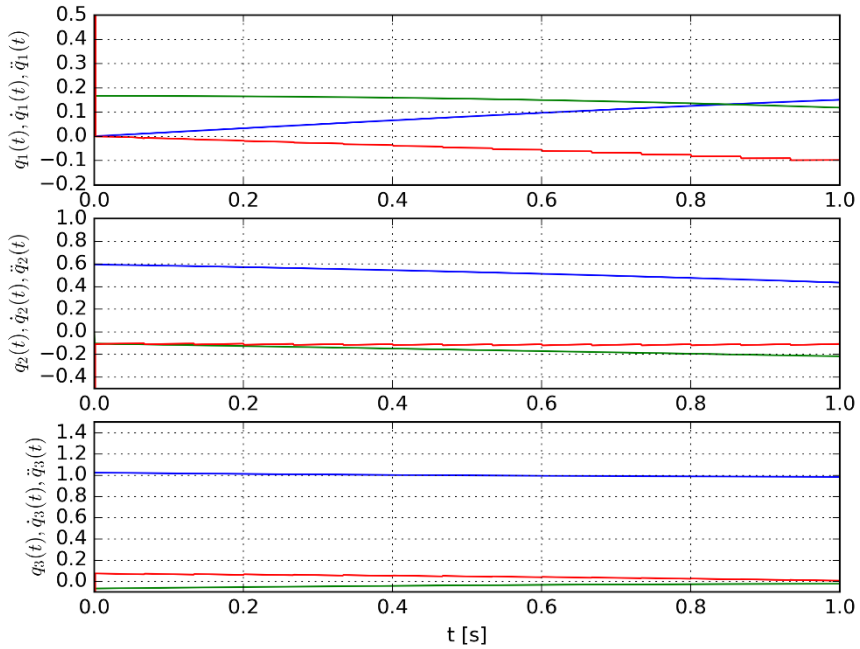
Rys. 3.8 Porównanie aproksymacji splinami trzeciego rzędu (po lewej) i 5 rzędu (po prawej)

Na rysunku 3.9 podobnie jak na rysunku 3.5 przedstawiono porównanie błędów interpolacji wielomianem trzeciego i piątego rzędu. Można zaobserwować, że pomimo zmiany typu trajektorii wykres błędu zachowuje się w podobny sposób. Ponownie wielomian stopnia piątego gorzej odwzorowuje oryginalną trajektorię.

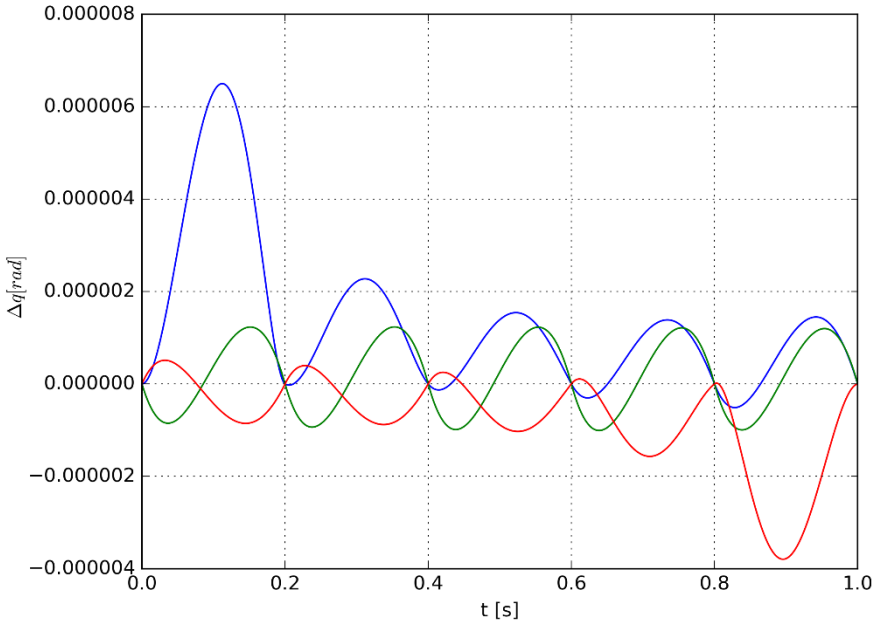


Rys. 3.9 Porównanie błędów aproksymacji splinami trzeciego rzędu (u góry) i 5 rzędu (u dołu)

Rysunki 3.10 oraz 3.11 przedstawiają podobnie jak rysunki 3.6 oraz 3.7 aproksymację uzyskana przy pomocy B-splinu drugiego rzędu. Szczególnie interesującym jest wykres błędu. Maksymalny błąd aproksymacji helisy jest o kilka rzędów mniejszy niż ten uzyskany przy interpolacji wielomianowej.



Rys. 3.10 Aproksymacja B-splinem drugiego rzędu



Rys. 3.11 Błąd aproksymacji B-splinem drugiego rzędu

Elipsa - krzywa na elipsoidzie

Równanie elipsoidy w swojej oryginalnej postaci:

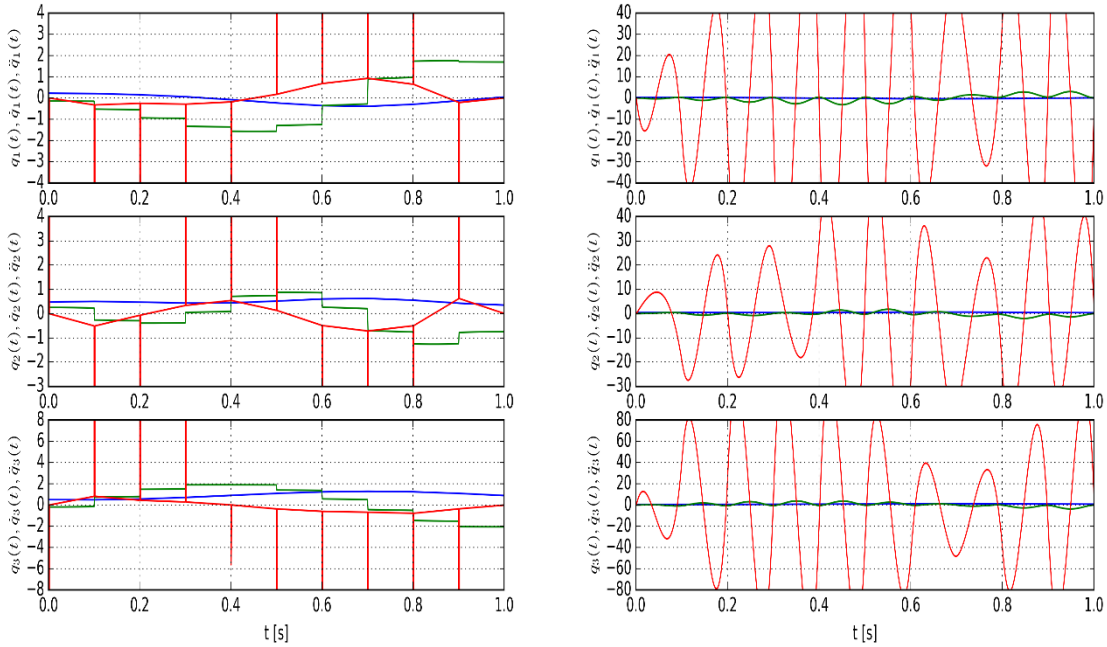
$$p: \frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}+\frac{(z-z_0)^2}{c^2}=1$$

przedstawia pewną trudność w użyciu. Dużo przydatniejsza jest jej reprezentacja parametryczna, która pozwala na proste wprowadzenie czasu:

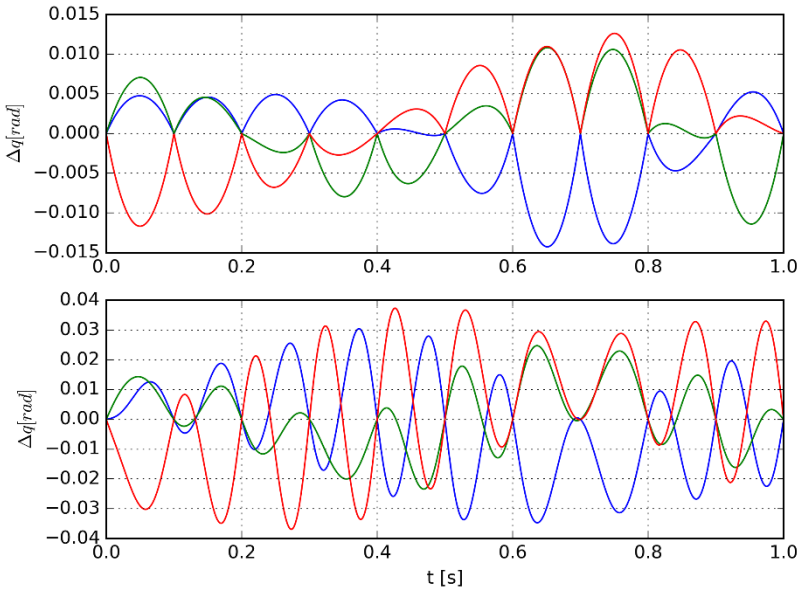
$$p: X(t)=X_0+a\sin(t)\cdot U+b\cos(t)\cdot V,\ t\in\langle 0, T_k\rangle$$

gdzie X_0 jest punktem środkowym elipsy, a i b są współczynnikami elipsy, U oraz V są wektorami o jednostkowej długości wyznaczającymi kierunki półosi wielkiej i półosi małej.

Na rysunkach 3.12 oraz 3.13 ponownie przedstawiono wyniki działania interpolacji wielomianowych. Współczynniki użyte podczas eksperymentów: $X_0 = [0.6, 0, 0.6]^T$, $U = [0.58, 0.58, 0.58]^T$, $V = [1.5, 0.5, 1]^T$, $a = 0.3$, $b = 0.1$, $T_k = 1$. Użyto jedenastu węzłów interpolacyjnych. Rezultaty dla elipsy pokrywają się z obserwacjami dla poprzednich typów trajektorii. Ponownie interpolacja niższego rzędu odzwierciedla oryginalną trajektorię z mniejszym błędem jednak wprowadza nieciągłości drugiej pochodnej.

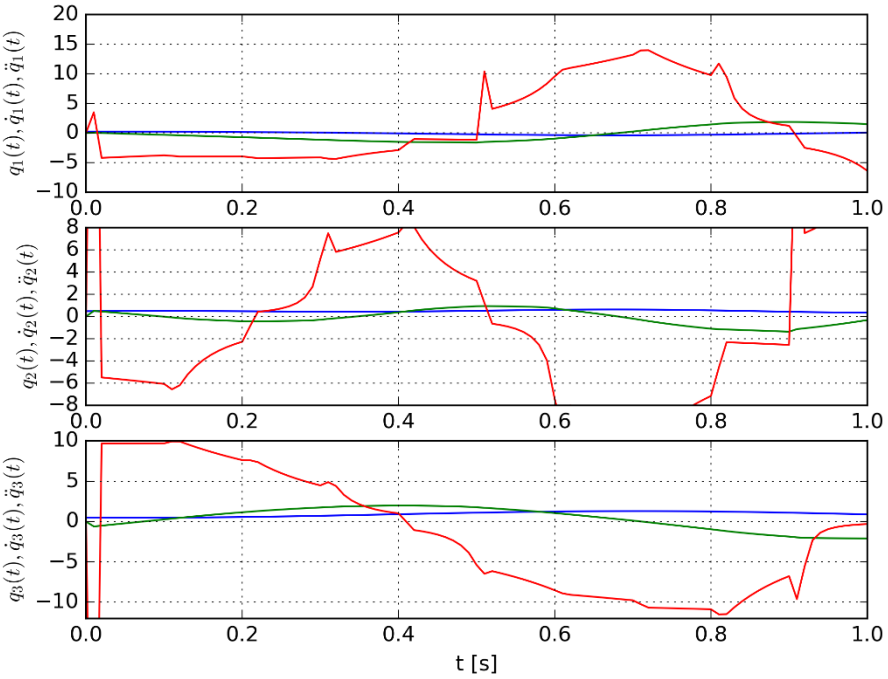


Rys. 3.12 Porównanie aproksymacji splinami trzeciego rzędu (po lewej) i piątego rzędu (po prawej)

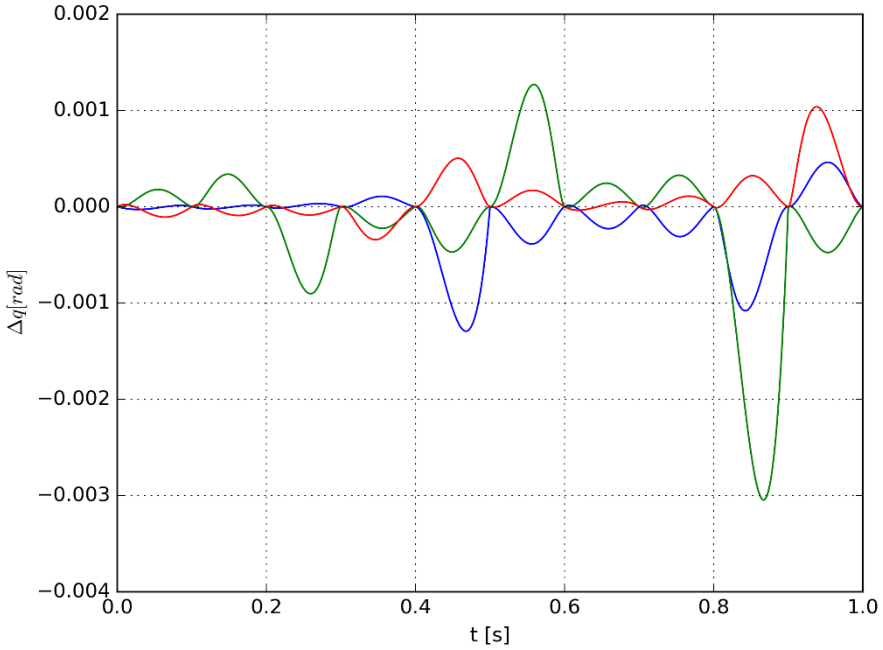


Rys. 3.13 Porównanie błędów aproksymacji splinami trzeciego rzędu (u góry) i piątego rzędu (u dołu)

Rysunek 3.14 przedstawia działanie aproksymacji B-splinem drugiego rzędu dla ruchu wzdłuż elipsy. Można zauważyć, że w wypadku elipsy, przyspieszenie kątowe jest wciąż mniejsze niż gdy trajektoria była interpolowana wielomianami. Na rysunku 3.15 przedstawiono błąd trajektorii aproksymującej, który dla tego typu trajektorii również jest mniejszy niż błąd interpolacji wielomianowej.



Rys. 3.14 Aproksymacja B-splinem drugiego rzędu



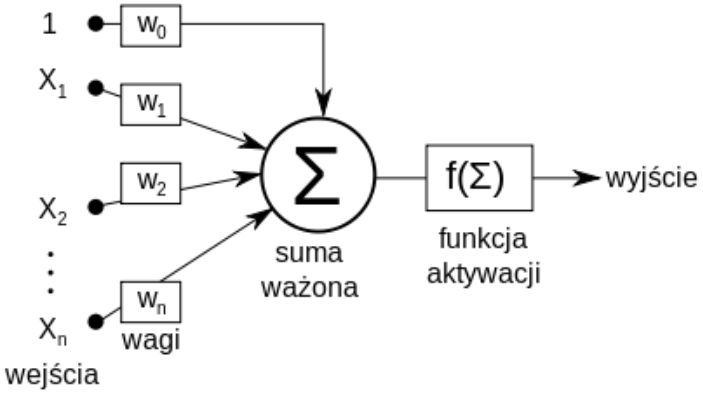
Rys. 3.15 Błąd aproksymacji B-splinem drugiego rzędu

4. Wstępne uwagi o sieci neuronowej

Siecie neuronowe są to rozproszone systemy przetwarzające informacje składające się z dużej liczby połączonych ze sobą identycznych lub podobnych do siebie, prostych jednostek obliczeniowych, które są ułożone w hierarchiczną strukturę. Korzenie tej metody sięgają neurobiologii, większość struktur ma swoje odpowiedniki w układach biologicznych. Warto zauważyć, że w zastosowaniach inżynieryjnych pochodzenie tej metody ma małe znaczenie.

Istotną cechą sieci neuronowych jest zdolność adaptacyjna, która pozwala im na zdobywanie "wiedzy" przez oddziaływania z otoczeniem zwane dalej uczeniem. Podczas tworzenia sieci neuronowej zamiast stosowania tradycyjnych metod programowania lub generacji algorytmów wykorzystuje się proces uczenia do osiągnięcia zadanych rezultatów. [69], [70],

W sieciach neuronowych jako podstawowe jednostki obliczeniowe stosuje się neuron McCullocha-Pittsa, dalej zwany po prostu neuronem. Składa się on z liniowego sumatora oraz nieliniowej funkcji aktywacji. Schemat przedstawiono na rysunku 4.1.



Rys. 4.1 Struktura neuronu McCullocha-Pittsa

Dla porządku przedstawiono równanie sumatora:

$$s = w_0 + \sum_{i=1}^{n} x_i w_i$$

s jest iloczynem skalarnym wektora wejść *x* i wektora wag *w. S*ygnał *s* podany zostaje na z nieliniową funkcję aktywacji. Wartości $x_0=1$ oraz w_0 służą jako dodatkowy offset. Stosuje się je, gdy wartość wyjścia funkcji aktywacji jest zależna od znaku wyjścia sumatora. Używa się różnych funkcji aktywacji, kilka wybranych zawarto w tabelce poniżej:

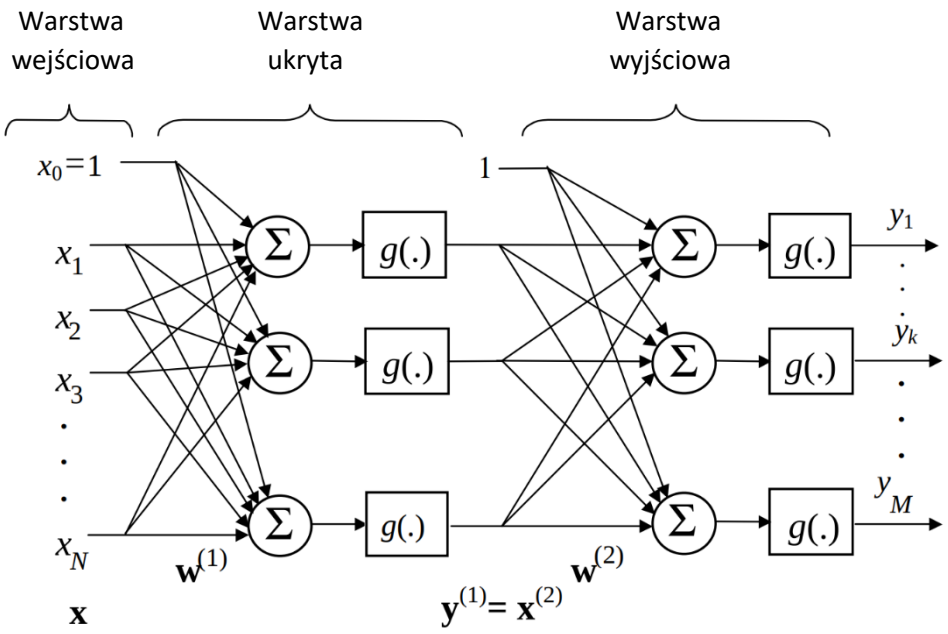
Nazwa	Formuła
Funkcja liniowa	$y(s)=as+b$
Obcięta funkcja liniowa	$y(s)=\begin{cases}-1, & dla\ \ s<-1\\ s, & dla\ \ -1\le s\le 1\\ 1, & dla\ \ s>1\end{cases}$
Funkcje progowe	$y(s)=\begin{cases}0, & dla\ \ s<a\\ 1, & dla\ \ s\ge a\end{cases}$

	$y(s)=\begin{cases}-1, & dla \quad s<a\\ 1, & dla \quad s\geq a\end{cases}$
Sigmoidalna funkcja unipolarna	$y(s)=\frac{1}{1+e^{-\beta s}}$
Tangens hiperboliczny	$y(s)=\frac{1-e^{-\beta s}}{1+e^{-\beta s}}$
Funkcja Gaussa	$y(s)=ae^{\frac{-(x-b)^2}{2c^2}}$
Prostownik (funkcja ReLU ang. Rectifier Linear Unit)	$f(s)=max(0,s)$

Tab. 4.1 Funkcje aktywacji sieci neuronowej

Struktury sieci neuronowych

W celu identyfikacji systemów stosuje się architektury sieci neuronowych ze sprzężeniem zwrotnym. Pomijając chwilowo to ostatnie opis zaczniemy od sieci jednokierunkowych. Na rysunku 4.2 przedstawiono strukturę sieci z propagacją w przód oraz jedną warstwą ukrytą.



Rys. 4.2 Wielowarstwowa sieć neuronowa z propagacją w przód

Równanie opisujące zależność wyjścia od wejścia dwóch złożonych warstw sieci:

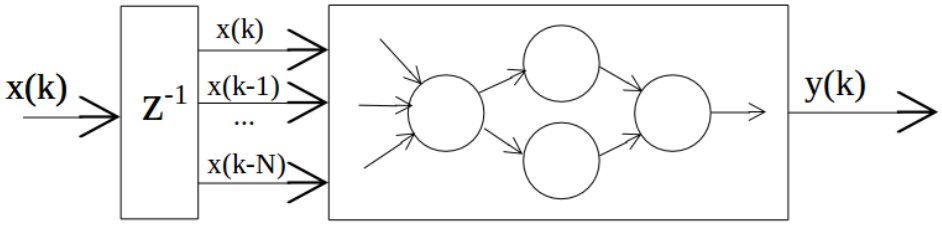
$$y=W^T f(V^T x), \tag{4.1}$$

gdzie *W* oraz *V* są to wektory wag poszczególnych wejść. Z analizy wzoru (4.1) można wyciągnąć wniosek, że sieć neuronowa posiada nieliniowe parametry układu w skrócie NLIP (ang. Nonlinear In Parameters). Aby zapewnić liniowość parametrów układu, należy usztywnić jedną z warstw sieci. Po podstawieniu do wzoru (4.1) $g(x)=f(V^T x)$ uzyskujemy:

$$y=W^T g(x)$$

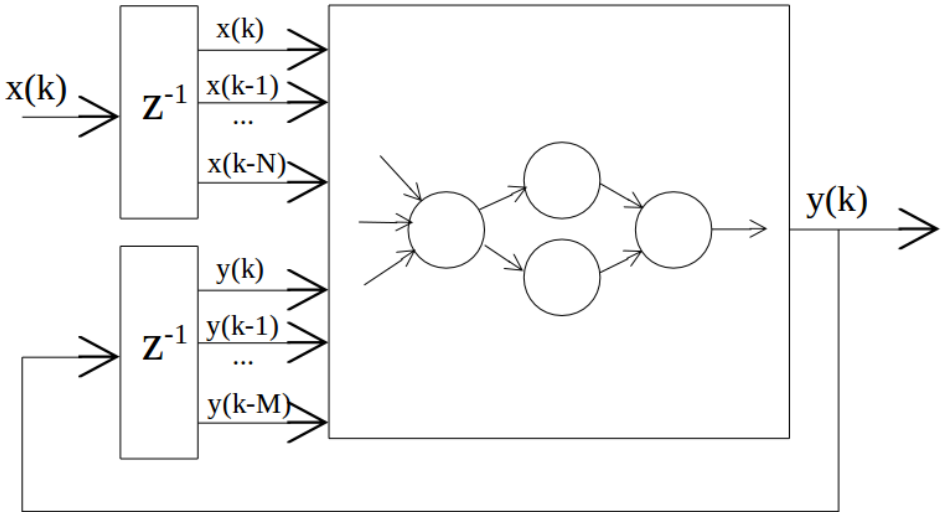
który spełnia założenie liniowości parametrów LIP (ang. Linear In Parameters).

Na rysunku 4.3 przedstawiono sieć neuronową z komórkami pamięci na wejściu. Jest ona pierwszym krokiem do uzyskania możliwości aproksymacji systemów dynamicznych, czyli z pamięcią. Można zauważyć, że wraz ze wzrostem komórek pamięci, zwiększa się wielkość warstwy wejściowej sieci.



Rys. 4.3 Sieć neuronowa z propagacją w przód wzbogacona o N komórek pamięci

Na rysunku 4.4 przedstawiono najprostszy typ sieci neuronowej ze sprzężeniem zwrotnym oraz komórkami pamięci. Sieć taka pozwala na aproksymację układów dynamicznych.



Rys. 4.4 Sieć neuronowa ze sprzężeniem zwrotnym z N komórkami pamięci wejścia oraz M komórkami pamięci wyjścia

Twierdzenie o uniwersalnej aproksymacji

Sieci z propagacją w przód z funkcją aktywacji typu sigmoidalnego oraz z jednym liniowym neuronem w warstwie wyjściowej są w stanie odwzorować każda ciągłą funkcję: $f: \mathbb{R}^n \rightarrow \mathbb{R}$ z dowolną dokładnością. Twierdzenia o zdolności aproksymacyjnej sieci tego typu zostały przedstawione m.in. w pracach: [15], [24], [32] i [42]. Użyto go między innymi w [47] oraz w [42].

Prace te przedstawiają możliwości aproksymacyjne sieci neuronowej z tylko jedną nieliniową warstwą ukrytą oraz warstwą wyjściową będącą pojedynczym liniowym neuronem. W praktyce używa się sieci o większej ilości warstw ukrytych. Zwykle zwiększenie ilości warstw zmniejsza sumaryczną ilość neuronów niezbędnych do osiągnięcia zadowalającego efektu aproksymacyjnego. Dodatkowo uczenie sieci wielowarstwowych jest zwykle łatwiejsze co przekłada się na lepszą estymację parametrów systemu.

Podczas identyfikacji badamy system dynamiczny, tzn. taki który posiada historię. Oznacza to, że wyjście systemu zależy nie tylko od aktualnego stanu wejść, ale również od poprzednich stanów. Istnieje kilka metod wprowadzenia zależności od stanu poprzedniego do sieci neuronowych. Dwiema podstawowymi metodami są: dodanie komórek pamięci oraz założenia sprzężenia zwrotnego pomiędzy elementami sieci. Różne metody wymuszają odmienną architekturę układu.

5. Identyfikacja robotów

Identyfikacja jest procesem dopasowania modelu matematycznego do danych otrzymanych eksperymentalnie z badanego systemu rzeczywistego. Identyfikacja składa się z kilku kluczowych etapów. Podstawowym zadaniem jest wykonanie eksperymentów sterowania systemem rzeczywistym. Otrzymuje się dostatecznie dużą ilość przebiegów, czyli trajektorii rzeczywistych zdjętych w czasie rzeczywistym aplikując przygotowane uprzednio sterowania. Z otrzymanych danych identyfikuje się strukturę systemu. Określenie struktury jest zadaniem złożonym. Ostatnim krokiem w identyfikacji jest estymacja parametrów obiektu. Gdy model systemu jest gotowy należy poddać go weryfikacji przy pomocy porównania odpowiedzi modelu oraz obiektu rzeczywistego. Zidentyfikowane modele obiektów można wykorzystać podczas tworzenia obserwatorów oraz regulatorów predykcyjnych. Znajomość parametrów obiektu pozwala na dobór regulatora adaptacyjnego, który na etapie syntezy jest dopasowywany do struktury obiektu. W przypadku identyfikacji robotów, struktura obiektu jest na ogół znana, jednak nie są znane parametry. Oznacza to, że mamy do czynienia z identyfikacją szarego pudełka (ang. grey box). Zdarzają się jednak sytuacje, gdy struktura robota również nie jest znana lub nie jest znana całkowicie. I tak w przypadku robota firmy Mitsubishi RV-2F-D, pomimo znanej konfiguracji kinematycznej robot jest sterowany przy pomocy wewnętrznego regulatora co zaburza jego dynamikę. Dlatego do identyfikacji użyto metodę tzw. czarnego pudełka (ang. black box) [61].

Rozdział składa się z kilku części. W pierwszej są przedstawione podstawowe metody identyfikacji parametrycznej oraz metody z użyciem sieci neuronowych. Następnie są prowadzone eksperymenty z robotem RV-2F-D firmy Mitsubishi. Na zakończenie porównuje się modele otrzymane przy pomocy różnych metod identyfikacji.

Identyfikacja Parametryczna

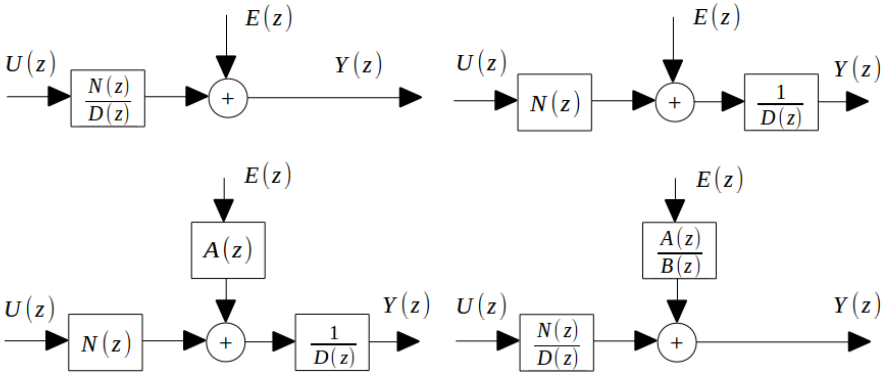
Identyfikacja parametryczna zakłada znajomość struktury obiektu definiowanej zbiorem parametrów do wyznaczenia, by spełnić zakładane kryteria jakościowe. Przykładem może być obiekt opisany wielomianem skończonego stopnia.

Struktura modelu może zostać poznana przez modelowanie, co w przypadku badań nad robotami osiąga się w dość prosty sposób. Można także podejmować próby dopasowania modeli standardowych, takich jak modele OE, FIR, ARX, ARMAX czy BJ. Kilka modeli standardowych pokazano na rysunku 5.1.

Model OE (ang. Output Error) nie uwzględnia w swojej strukturze części zakłóceniowej. Dynamika systemu oraz dynamika zakłóceń jest rozdzielona. Równanie modelu jest postaci:

$$Y(z)=\frac{N(z)}{D(z)}U(z)+E(z)$$

Jeśli system typu OE działa bez sprzężenia zwrotnego podczas zbierania próbek, pełna dynamika modelu w postaci transmitancji obiektu, może zostać wykryta, niezależnie od typu zakłócenia.



Rys. 5.1 Wybrane modele parametryczne: w lewym górnym rogu OE, w prawym górnym rogu ARX, w lewym dolnym rogu ARMAX, w prawym dolnym rogu BJ

W modelu ARX, czyli autoregresyjnym z zewnętrznym wejściem (ang. Auto-Regresive eXogenous) równanie modelu ma postać:

$$Y(z)=\frac{N(z)}{D(z)}X(z)+\frac{1}{D(z)}E(z)$$

Ta struktura jest łatwa do estymacji. Można jej dokonać przy pomocy regresji liniowej. Największą wadą modelu jest to, że mianownik część zakłóceniowej jest współdzielony z częścią od dynamiki. Może to prowadzić do sytuacji, że estymowane wielomiany osiągną wyższy stopień niż powinny, a więc do błędnego wyznaczenia dynamiki.

Model ARMAX, model autoregresyjny ze średnią ruchomą i zewnętrznym wejściem (ang. Auto-Regresive Moving Average eXogenous), jest rozszerzeniem modelu ARX. Wprowadza dodatkową elastyczność przy modelowaniu części od zakłóceń, przez wprowadzenie dynamiki zakłóceń. Z tego względu jest jednym z najczęściej spotykanych modeli identyfikacyjnych stosowanych w przemyśle.

Model Boxa-Jenkinsa jest najbardziej zaawansowany z pośród wcześniej opisanych. Dynamika zakłóceń i dynamika obiektu nie są ze sobą połączone. Model Jest trudny w estymacji, ale daje najlepsze rezultaty. Oto jego postać:

$$Y(z)=\frac{N(z)}{D(z)}X(z)+\frac{A(z)}{B(z)}E(z)$$

Identyfikacja robota przemysłowego Mitsubishi

Dostępność wszystkich sygnałów pomiarowo-sterujących obiektu jest podstawowym warunkiem do prowadzenia prac badawczych przy użyciu robota. Niestety producenci sprzętu są przede wszystkim zainteresowani przedstawieniem go jako narzędzia użytecznego do wykonywania powtarzalnych operacji z dużą dokładnością, a mniej lub wcale nie są zainteresowani robotem jako narzędziem badawczym. Szczegóły pomiarowe, sposoby zakodowania sygnałów itp. są słabo dostępne.

Podczas identyfikacji posłużono się danymi pozyskanymi bezpośrednio z enkoderów robota. Używając metod inżynierii wstecznej, autor był w stanie przeanalizować nieupubliczniony protokół enkoderowy. Jego opis zamieszczono w dodatku B. W dodatku C przedstawiono przykładowe przebiegi pomiarowe dla wymuszenia prostokątnego.

Przy pomocy algorytmu ARMAX uzyskano poniższe wartości transmitancji operatorowej dla kolejnych członów robota:

$$T_1(s)=\frac{0.0002s+0.0061}{-0.1516s^5+0.4668s^4-1.0545s^3+2.2321s^2-2.4865s+1.0000}$$

$$T_2(s)=\frac{0.0001s+0.0053}{-0.1068s^5+0.3617s^4-1.0767s^3+2.4252s^2-2.5980s+1.0000}$$

$$T_3(s)=\frac{0.0015s+0.0069}{-0.1516s^5+0.2286s^4-0.2185s^3+1.2495s^2-2.0996s+1.0000}$$

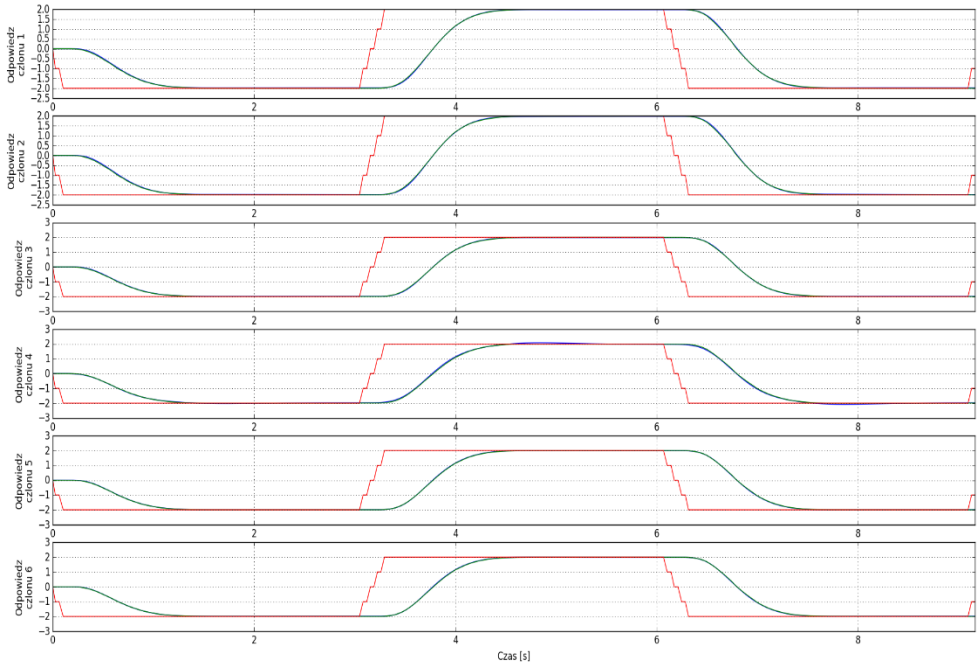
$$T_4(s)=\frac{0.0012s-0.0003}{0.1463s^5+0.0705s^4-1.9767s^3+4.1426s^2-3.3819s+1.0000}$$

$$T_5(s)=\frac{0.0037s+0.0072}{-0.3083s^5+0.7539s^4-0.7134s^3+1.2615s^2-1.9827s+1.0000}$$

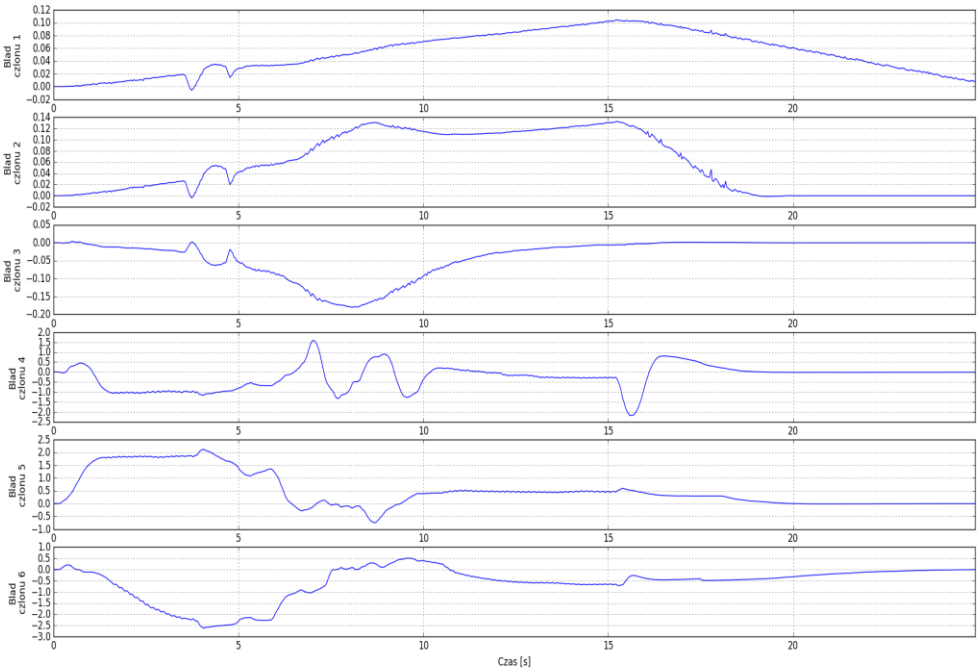
$$T_6(s)=\frac{0.0053s+0.0072}{-0.2969s^5+0.4786s^4+0.1321s^3+0.3313s^2-1.6324s+1.0000}$$

Poprawność uzyskanych wyników identyfikacji została zweryfikowana za pomocą niezależnych przebiegów, zebranych podczas pracy robota MITSUBISHI RV-2D. Na rysunku 5.2 przedstawiono porównanie odpowiedzi na wymuszanie skokowe użyte jako wejście do algorytmu identyfikującego oraz odpowiedzi modelu na tę samą wartość zadaną. Na rysunku 5.3 znajdują się wykresu błędu odpowiedzi modelu względem rzeczywistej odpowiedzi systemu.

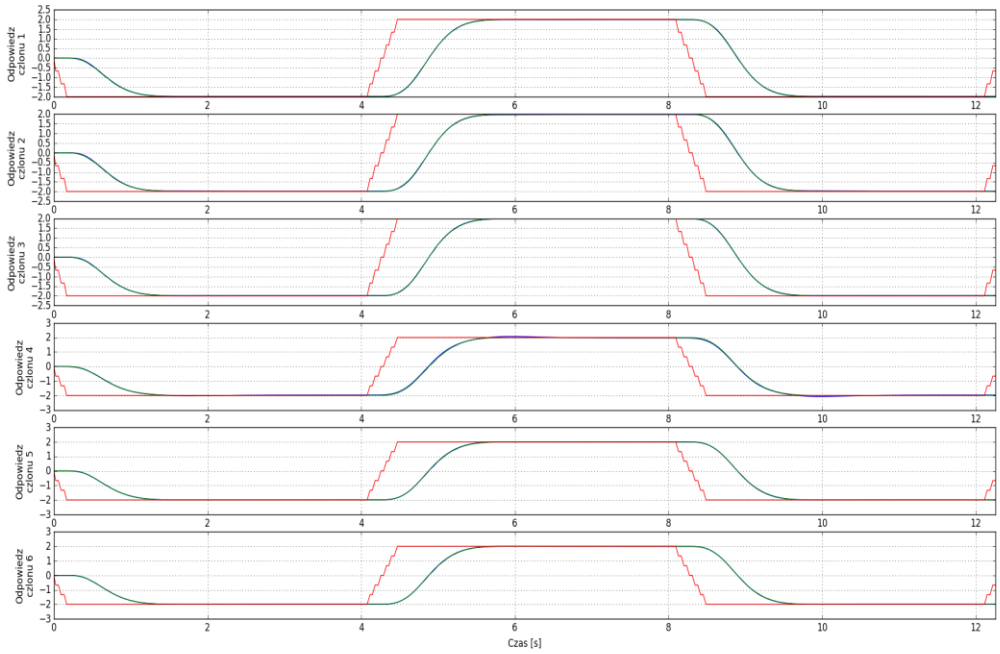
Na rysunku 5.4 przedstawiono porównanie odpowiedzi modelu uzyskanego przy pomocy algorytmu AMAX na inne niż użyte do identyfikacji wymuszenie. Tym razem amplituda skoku była wyższa niż w poprzednim eksperymencie. Na rysunku 5.5 przedstawiono wykresy błędu dla mocniejszego wymuszenia skokowego. Można zaobserwować, że pomimo innego wymuszenia model wciąż zachowuje się poprawnie, a błąd nie wzrósł w sposób znaczący.



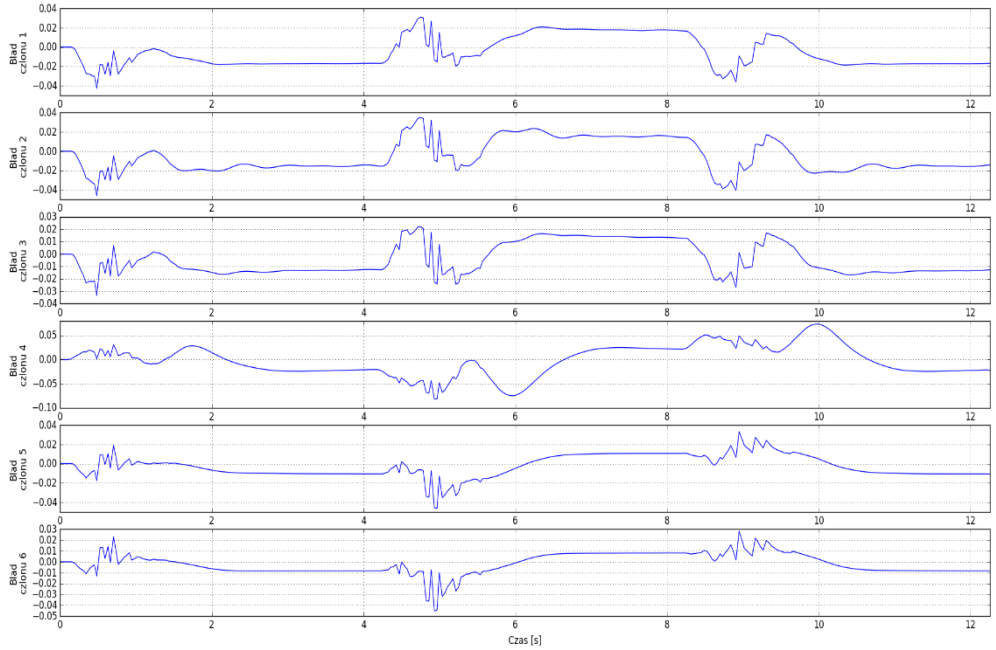
Rys. 5.2 Porównanie przebiegu zebranego z robota oraz uzyskanego za pomocą transmitancji operatorowej wygenerowanej przez algorytm ARMAX. Na czerwono zaznaczono wartość zadaną, na niebiesko przebieg oryginalny, na zielono przebieg z modelu



Rys. 5.3 Błąd pomiędzy przebiegiem zebranym z robota oraz uzyskanym za pomocą transmitancji operatorowej wygenerowanej przez algorytm ARMAX



Rys. 5.4. Porównanie przebiegu zebranego z robota oraz uzyskanego za pomocą transmitancji operatorowej wygenerowanej przez algorytm ARMAX. Na czerwono zaznaczono wartość zadaną, na niebiesko przebieg oryginalny, na zielono przebieg z modelu



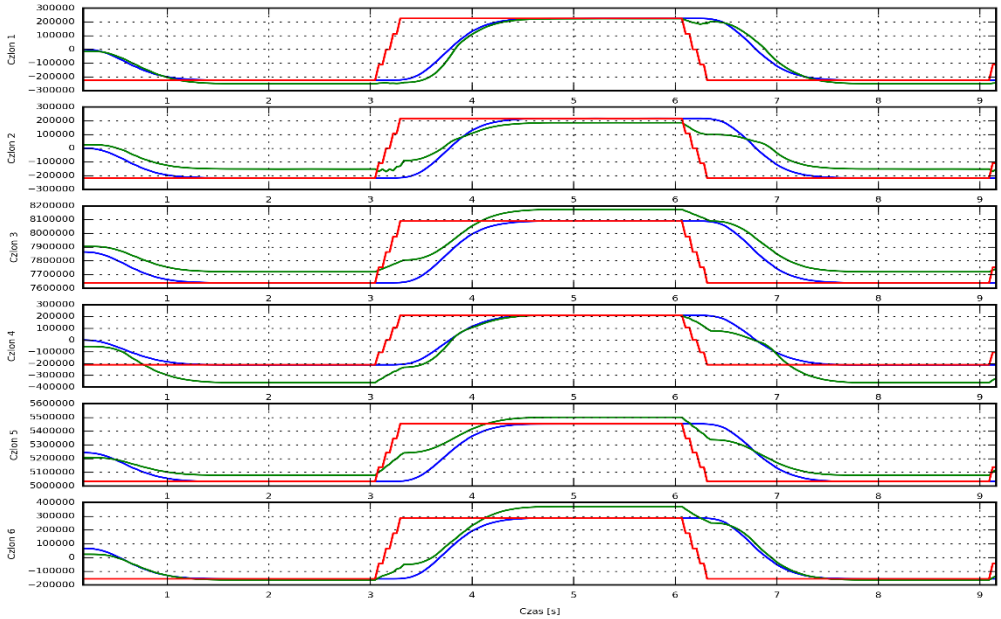
Rys. 5.5 Błąd pomiędzy przebiegiem zebranym z robota oraz uzyskanym za pomocą transmitancji operatorowej wygenerowanej przez algorytm ARMAX

Identyfikacja przy pomocy sieci neuronowej

Jak wynika z twierdzenia o uniwersalnej aproksymacji sieć neuronowa z propagacją w przód (ang. feed-forward) z jedną warstwą ukrytą, o skończonej ilości neuronów jest w stanie aproksymować dowolną ciągłą funkcję na zbiorze zawartym [15]. Bardziej odpowiednią jest aproksymacja dowolnej funkcji ciągłej ze skończonymi skokami. Dla takich przypadków udowodniono twierdzenia o poprawnej aproksymacji z użyciem sieci neuronowej. Większość obiektów mechanicznych podpada pod taki opis na skutek konieczności zamodelowania zjawiska tarcia statycznego. Przykładem jest choćby badany robot przemysłowy Mitsubishi.

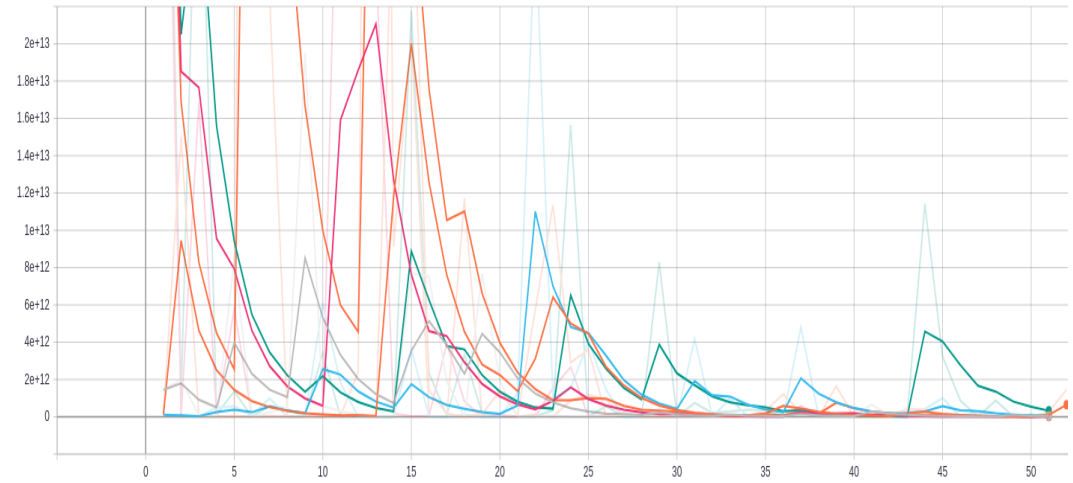
Aby nauczyć powyższy typ sieci dynamiki robota, posłużono się siecią opisaną w powyższym twierdzeniu. Sieć na wejściu przyjmuje pięć kolejnych wyjść z robota oraz dwie próbki sterowania, z okresu poprzedniego i aktualnego. Jako funkcje aktywacji użyto funkcji ReLU. Nuka sieci była przeprowadzona w sposób nadzorowany. Do przeprowadzania nauki oraz organizacji sieci neuronowej użyto frameworku do zastosowań uczenia maszynowego TensorFlow.

Na rysunku 5.6 przedstawiono porównanie odpowiedzi skokowej uzyskanej przy pomocy sieci neuronowej oraz wyniku eksperymentalnego. Można zauważyć, że pomimo zachowania ogólnego kształtu błąd aproksymacji jest zauważalny.



Rys. 5.6 Porównanie przebiegu zebranego z robota oraz uzyskanego za pomocą sieci neuronowej. Na czerwono zaznaczono wartość zadaną, na niebiesko przebieg oryginalny, na zielono przebieg z modelu

Jako dowód na poprawność działania algorytmu uczącego na rysunku 5.7 przedstawiono funkcję błędu. Można zauważyć, że wraz z kolejnymi epokami wielkość błędu dla wszystkich osi malała.



Rys 5.7 Błąd aproksymacji podczas uczenia sieci

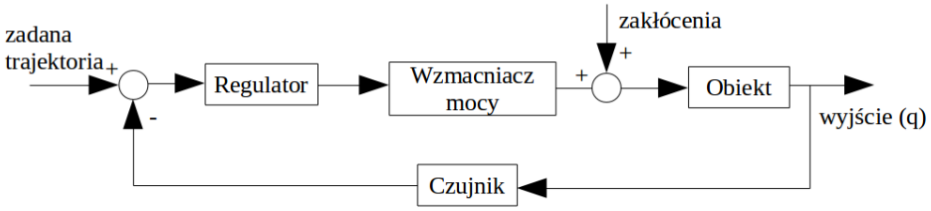
6. Sterowniki robotów o otwartym łańcuchu kinematycznym

Przedstawiono poniżej wybrane typy regulatorów używane w robotyce. Podstawowymi trzema wielkościami, przy pomocy których można przeprowadzić akcję sterowania robotów są: położenie kątowe, prędkość kątowa oraz moment obrotowy pomiędzy poszczególnymi członami. Manipulatorami-robotami można sterować na dwa podstawowe sposoby, przez sterowanie każdym członem osobno oraz wszystkimi jednocześnie korzystając z równań opisujących dynamikę całego manipulatora.

Na początek opisano sterowniki klasyczne, takie jak PID oraz PD często spotykane w innych zadaniach sterowania. Następnie opisano regulatory adaptacyjne, które stanowią dużą i ważną grupę sterowników w robotyce. Badania nad nimi pozwoliły na wygenerowanie wielu interesujących algorytmów sterowania. Następnie opisano regulację wykorzystującą sieci neuronowe. Pokazano aplikację jak sterować w przypadkach niezidentyfikowanych manipulatorów. Ostatnimi w tym rozdziale są regulatory z predykcją. Wyznaczają sterowania korzystając ze znanego modelu regulatora.

Sterowanie pojedynczymi członami

Pierwszy typ sterowania jest jednocześnie najprostszym. Sterowanie wyznaczane jest indywidualnie dla każdego przegubu, nie biorąc pod uwagę całościowej dynamiki oraz kinematyki manipulatora. Jako dane wejściowe do zaprojektowania regulatora są zwykle wybierane dynamika elementu wykonawczego danego członu oraz planowany sposób poruszania się. Przy prostych zadaniach sterowania, gdy nie jest wymagana wysoka koordynacja elementów wykonawczych, taka metoda sterowania może okazać się zadowalająca. Różniące się regulatory winny zostać zsyntetyzowane dla ruchu PTP oraz dla podążania za zadaną trajektorią. Dynamika serwomechanizmu prądu stałego została pokazana w podrozdziale Dynamika członów wykonawczych. Schemat tradycyjnego układu sterowania w jednej osi robota pokazano na rysunku 6.1.



Rys. 6.1 Tradycyjny układ sterowania dla regulatorów jednoosiowych

Sterowanie nadążne

Na ogół mamy do czynienia z regulatorami nadążnymi PD oraz PID. Mogą być realizowane zarówno w sposób analogowy [56] jak i cyfrowy. Ten rodzaj sterowania jest szczególnie skuteczny dla robotów pracujących przy niskiej prędkości. Przykładem mogą być roboty z bardzo dużymi przekładniami.

Łącząc ze sobą równania (2.3) oraz (2.12) otrzymujemy układ:

$$A_{m_k}\ddot{q}_{m_k}+B_{m_k}\dot{q}_{m_k}=K_{m_k}v_k-F_{m_k}-R_m\left(\sum_j d_{kj}\ddot{q}_j+\sum_{i,j}c_{ijk}(q)\dot{q}_i\dot{q}_j-\frac{\partial P}{\partial q_k}\right),\quad k=1,\ldots,n$$

w którym dynamika manipulatora, wraz z modelem tarcia pełnią rolę zakłócenia:

$$A_{m_k}\ddot{q}_{m_k}+B_{m_k}\dot{q}_{m_k}=u_k-d_k, \tag{6.1}$$

W tym miejscu należy zauważyć, że zarówno współczynniki A_m jak i B_m mają wartości przybliżone. Rzeczywiste wartości są zależne od wartości zmiennych przegubowych.

Pierwszym regulatorem proponowanym tutaj, a przeznaczonym do sterowania systemem (1) jest klasyczny regulator PD:

$$u(t)=K_P(q_d(t)-q(t))-K_D\dot{q}(t), \tag{6.2}$$

Transmitancja układu zamkniętej regulacji obiektu (6.1) z powyższym regulatorem wyraża się wzorem:

$$L(s)=\frac{K_p}{M_{PD}(s)}Q_d(s)-\frac{1}{M_{PD}(s)}D(s), \tag{6.3}$$

Gdzie Q_d jest transformatą Laplace'a zaplanowanej trajektorii, D jest transformatą Laplace'a zakłóceń, a M jest wielomianem charakterystycznym układu:

$$M_{PD}(s)=A_{m_k}s^2+(B_{m_k}+K_D)s+K_P$$

Kolejnym klasycznym regulatorem w układzie o rozdzielonych osiach, jest regulator PID. Sterowanie generowane przez niego zapisać można w postaci:

$$u(t)=K_P(q_d(t)-q(t))-K_D\dot{q}(t)+K_I\int q_d(t)-q(t)dt. \tag{6.4}$$

Transmitancja zamkniętego układu z regulatorem PID przyjmie zatem postać:

$$L(s)=\frac{K_Ds^2+K_Ps+K_I}{M_{PID}(s)}Q_d(s)-\frac{1}{M_{PID}(s)}D(s), \tag{6.5}$$

gdzie M_{PID}:

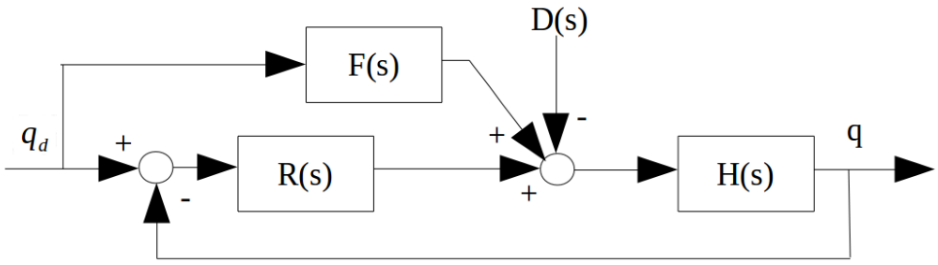
$$M_{PID}(s)=A_{m_k}s^3+(B_{m_k}+K_D)s^2+K_Ps+K_I.$$

Za pomocą twierdzenia Hurwitza możemy stwierdzić, że układ jest stabilny, gdy zachodzi poniższa nierówność:

$$K_I<\frac{(B_{m_k}+K_D)K_P}{A_{m_k}}$$

Sterowanie z uwzględnieniem sprzężenia w przód

Podążanie za trajektorią jest jednym z podstawowych i najważniejszych zadań w robotyce. Znajomość zaplanowanej wcześniej trajektorii pozwala na polepszenie sterowania przy pomocy przewidywania przyszłych sterowań, czyli inaczej stosowanie sprzężenia w przód (ang. feedforward). Schemat układu regulacji ze sprzężeniem w przód przedstawiono na rysunku 6.2.



Rys. 6.2 Układ regulacji ze sprzężeniem feedforward

Dobierając transmitancję części w przód jako odwrotność transmitancji obiektu sterowanego:

$$F(s)=A_{m_x}s^2+B_{m_x}s$$

oraz stosując regulator PD transmitancja błędu śledzenia trajektorii q_d wyraża się wzorem:

$$L_e(s)=\frac{E(s)}{D(s)}=\frac{-1}{A_{m_x}s^2+(B_{m_x}+K_D)s+K_P}. \tag{6.6}$$

Z powyższego wynika, że całkowity błąd sterowania pochodzi od zakłócenia będącego dynamiką manipulatora. Koncept sterowania ze sprzężeniem w przód można rozszerzyć przez dodanie członu obliczającego moment siły, niezbędny do wykonania zadania sterowania na podstawie założonej uprzednio trajektorii. Momenty te są znane już w chwili planowania ruchu robota. Pozwala to na poprawę jakości sterowania. Powyższa technika zostanie szerzej opisana w kolejnej sekcji.

Sterowanie wieloczłonowe

Aby w pełni wykorzystać potencjał wynikający z wiedzy o dynamice manipulatora-robota należy zastosować regulatory operujące na całym stanie robota jednocześnie. Sterowanie wieloczłonowe pozwala na efektywne wykonanie zadania sterowania robotem. Sterowanie jest wyznaczane na podstawie równania (2.4), które w uproszczonej formie jest postaci:

$$D(q)\ddot{q}+N(q,\dot{q})+\tau_d=\tau, \tag{6.7}$$

gdzie τ to moment sterujący a τ_d to zakłócenia. Ponieważ błąd regulacji, wraz z pochodnymi, można zapisać jako:

$$e(t)=q_d(t)-q(t),\quad \dot{e}(t)=\dot{q}_d(t)-\dot{q}(t),\quad \ddot{e}(t)=\ddot{q}_d(t)-\ddot{q}(t) \tag{6.8}$$

Zadanie sterowania sprowadza się do minimalizacji poniższego równania:

$$\ddot{e}(t)=\ddot{q}_d(t)-D^{-1}(N+\tau_d-\tau). \tag{6.9}$$

Linearyzując model przez sprzężenie zwrotne, podobnie jak w (2.10):

$$u=\ddot{q}_d(t)-D^{-1}(N-\tau), \tag{6.10}$$

Otrzymujemy system:

$$\frac{d}{dt}\begin{bmatrix}e\\ \dot{e}\end{bmatrix}=\begin{bmatrix}0 & I\\ 0 & 0\end{bmatrix}\begin{bmatrix}e\\ \dot{e}\end{bmatrix}+\begin{bmatrix}0\\ I\end{bmatrix}u+\begin{bmatrix}0\\ I\end{bmatrix}w, \tag{6.11}$$

Gdzie *w* stanowi część od zakłóceń układu. Rozwiązując równanie (6.10) względem momentu sterowania otrzymamy:

$$\tau = D(\ddot{q}_d - u) + N \,, \tag{6.12}$$

Równanie (6.12) nazywamy sterowaniem na podstawie prawa wyliczonego momentu (ang. computed-torque control law). Jego istotność polega na byciu pomostem, który na podstawie sterowania *u*, stabilizującego układ (6.11), pozwala manipulatorowi-robotowi na podążanie za zadaną uprzednio ścieżką.

Regulatory klasyczne

Najprostszą grupę regulatorów stabilizujących układ (6.11) stanowią regulatory klasyczne. Ich struktura jest zbliżona, dla regulatorów PDD oraz PD+, lub identyczna, dla regulatorów PID oraz PD, z tymi, które były stosowane podczas regulacji poszczególnymi członami z osobna. Pierwszym niech będzie regulator PD o prostym równaniu:

$$u = -K_D\dot{e} - K_P e \,, \tag{6.13}$$

wówczas równanie na moment siły podczas podążania po zadanej ścieżce sprowadza się do postaci:

$$\tau = D(\ddot{q}_d + K_D\dot{e} + K_P e) + N \,, \tag{6.14}$$

natomiast część zakłóceniowa sprowadza się do postaci:

$$\ddot{e} + K_D\dot{e} + K_P e = w$$

A równania stanu do macierzowego równania:

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} w \,, \tag{6.15}$$

gdzie:

$$K_D = diag(K_{D_i}), \quad K_P = diag(K_{P_i})$$

Przy powyższych założeniach wielomian charakterystyczny układu (6.14) wyraża się wzorem:

$$W_c = \prod_{i=1}^{n} (s^2 + k_{D_i}s + k_{P_i})$$

Układ błędu nadążnego jest asymptotycznie stabilny, gdy wszystkie współczynniki macierzy K_D i K_P są dodatnie. Ponadto, ponieważ błąd *w* jest ograniczony, to rozwiązania systemu (6.15) są również ograniczone.

Dla bardziej ogólnego regulatora PID mamy:

$$\dot{\epsilon} = e$$
$$\tau_c = D(q)(\ddot{q}_d + K_v\dot{e} + K_p e + K_i\epsilon) + C(q,\dot{q})$$

Macierzowe równania stanu dla regulatora PID jest postaci:

$$\frac{d}{dt}\begin{bmatrix}\epsilon\\ e\\ \dot{e}\end{bmatrix}=\begin{bmatrix}0 & I & 0\\ 0 & 0 & I\\ -K_I & -K_P & -K_D\end{bmatrix}\begin{bmatrix}\epsilon\\ e\\ \dot{e}\end{bmatrix}+\begin{bmatrix}0\\ 0\\ I\end{bmatrix}w \quad (6.16)$$

gdzie:

$$K_D=diag\left[K_{D_i}\right],\quad K_P=diag\left[K_{P_i}\right],\quad K_I=diag\left[K_{I_i}\right]$$

A wielomian charakterystyczny przedstawia się następująco:

$$W_c=\prod_{i=1}^{n}\left(s^3+k_{D_i}s^2+k_{P_i}s+k_{I_i}\right) \quad (6.17)$$

Z powyższego można stwierdzić, że dla współczynników spełniających nierówność:

$$k_{I_i}<k_{D_i}k_{P_i}$$

System śledzenia błędu dla regulatora PID jest stabilny.

Wariacją na temat tej grupy regulatorów są rozbudowane o znajomość części dynamiki manipulatora sterowniki takie jak PD+, gdzie do standardowego członu proporcjonalno-różniczkującego dodano informacje na temat zachowania się mas manipulatora od grawitacji. Używając oznaczeń z (6.7) formuła dla regulatora typu PD+ przyjmuje postać:

$$\tau_c=K_v\dot{e}+K_pe+g(q) \quad (6.18)$$

Aby pokazać stabilność w sensie Lapunowa w stanie ustalonym powyższego układu posłużymy się poniższym funkcjonałem:

$$V=\frac{1}{2}\left(\dot{q}^TD\dot{q}+e^TK_pe\right),\ e=q_d-q \quad (6.19)$$

Różniczkując po czasie oraz zauważając, że $\dot{q}=\dot{e}$ otrzymujemy:

$$\dot{V}=\dot{q}^T\left(D\ddot{q}+\frac{1}{2}\dot{D}\dot{q}-K_pe\right) \quad (6.20)$$

Podstawiając (7) do powyższego równania otrzymujemy:

$$\dot{V}=\dot{q}^T\left(\frac{1}{2}\dot{D}-N-K_v\right)\dot{q}$$

Korzystając z symetryczności skośnej zachodzącej między macierzami *D* oraz *N* otrzymujemy:

$$\dot{V}=-\dot{q}^TK_v\dot{q}$$

Ponieważ *V* jest dodatnio określone a *V'* ujemnie półokreślone układ (6.7) sterowany przy pomocy regulatora (18) jest stabilny w sensie Lapunowa.

Aby wykazać stabilność asymptotyczną należy posłużyć się twierdzeniem LaSalle-a. Ponieważ funkcjonał *V* posiada kres dolny będący zerem a *V'* jest niedodatnie, można udowodnić, że *V* jest ograniczone, co można zapisać jako:

$$-\int_0^\infty \dot{V}\,dt=const$$

Aby pokazać, że *V'* dąży do zera użyjemy lematu Barbalata. Najpierw pokażemy, że *V'* jest ciągła jednostajnie. Można zauważyć, że *V''* jest funkcją ciągłą:

$$\ddot{V} = -2\dot{q}^T K_v \ddot{q} .$$

Korzystając ze wcześniejszego dowodu na ograniczoność *q* oraz na podstawie (6.7) oraz ograniczoności $M^{-1}(q)$ możemy stwierdzić, że *q''* jest ograniczone. Z tego względu *V''* jest ograniczone, a *V* jest jednostajnie ciągłe. Na tej podstawie, korzystając z lematu Barbalata, *V'* dąży do zera.

Na podstawie powyższego możemy stwierdzić, że *q'* dąży do zera. Podstawiając zatem $\dot{q} = 0$ do (2.4) z regulatorem opisanym przez (6.18) uzyskujemy zależność pomiędzy drugą pochodną zmiennych przegubowych oraz błędem śledzenia:

$$\ddot{q} = M^{-1} K_p e .$$

Z powyższego równania wynika, że niezerowy błąd skutkuje niezerowym przyspieszeniem, a to oznacza $\dot{q} \neq 0$. Ponieważ taka sytuacja jest możliwa tylko dla $q(t) = 0$, możemy wywnioskować, że zarówno $e(t)$ jak i $q'(t)$ dążą do zera.

Regulator liniowo kwadratowy

Regulatory liniowo-kwadratowe (LQR, ang. Linear-Quadratic Regulator) posiada sprzężenie zwrotne z dostępem do stanu obiektu. Regulator ten jest stosowany w zadaniach sterowania, gdy system jest układem liniowych równań różniczkowych, a koszt sterowania opisany jest formą kwadratową. Regulator ma za zadanie minimalizować zadana formę kwadratową, jest zatem regulatorem optymalnym.

Korzystając z równań (6.11), (6.12) oraz podstawiając regulator *PD* (6.13), możemy przedstawić problem sterowania robotem jako problem LQ:

$$u = -Kx = -\begin{bmatrix} K_p & K_v \end{bmatrix} x = -K_p e - K_v \dot{e} , \tag{6.21}$$

a jako funkcjonał kosztu *J* przyjąć:

$$J = \frac{1}{2}\int_0^\infty \left(x^T Q x + u^T R u \right) dt , \tag{6.22}$$

gdzie macierz *Q* jest symetryczna i dodatnio półokreślona, a macierz *R* jest symetryczna i dodatnio określona. Aby wyznaczyć macierze wzmocnień K_P oraz K_v należy użyć poniższego równania:

$$K = R^{-1} B^T P , \tag{6.23}$$

gdzie *P* jest symetryczną macierzą stanowiącą rozwiązanie równania Riccatiego:

$$A^T P + PA + Q - PBR^{-1} B^T P = 0 . \tag{6.24}$$

Przyjmując macierz *Q* w postaci:

$$Q = diag\left[Q_p, Q_v \right], \quad Q_p, Q_v \in R^{n \times n}$$

I z uwagi na prostą strukturę (6.11), rozwiązanie problemu (6.24) wyznacza poniższe sterowanie:

$$K_p=\sqrt{Q_p R^{-1}}, \quad K_v=\sqrt{2K_p+Q_v R^{-1}}$$

Regulatory adaptacyjne

Najważniejszą cechą regulatorów adaptacyjnych jest ich odporność na nieznane parametry manipulatora. Podczas strojenia tego typu regulatora wystarczy tylko ogólna znajomość dynamiki obiektu, by zapewnić stabilność sterowania. Wewnętrzna budowa regulatorów pozwala im na elastyczne dostosowanie się do rzeczywistego obiektu mimo niedoskonałości wzorca. Tego typu regulatory w sposób ciągły obserwują wyjście oraz wejście obiektu, aby na tej podstawie dokonać modyfikacji parametrów wbudowanego modelu. Z upływem czasu, wartości współczynników modelu stają się zbliżone do wartości parametrów rzeczywistego manipulatora. Poniżej podano dwie metody wykorzystania algorytmów adaptacyjnych w robotyce.

Algorytm sterowania manipulatorem przy użyciu wyznaczonego momentu siły wymaga znajomości dokładnego modelu obiektu dla generacji sterowania. Na ogół nie da się uzyskać perfekcyjnego modelu, dlatego algorytm sterowania staje się niedokładny już w chwili jego konstruowania. Pierwszym poważnym problemem jest, że ze względu na błędy pomiaru, nigdy nie możemy dostarczyć idealnych parametrów obiektu do modelu. Kolejnym jest zmienność niektórych czynników w czasie. Dotyczy to przede wszystkim współczynników tarcia oraz masy ładunku podlegającego manipulacjom. Zwykle masa ładunku, którym operuje manipulator jest znana tylko w przybliżeniu i może zmieniać się w ograniczonym zakresie. Dzięki zastosowaniu algorytmu adaptacyjnej zmiany parametrów modelu jest możliwe ciągłe śledzenie zmian parametrów modelu i poprawianie parametrów na bieżąco.

Załóżmy, że równanie przedstawia model robota o nieznanych parametrach:

$$\tau=\hat{D}(q)\ddot{q}+\hat{C}(q,\dot{q})\dot{q}+\hat{F}(\dot{q})+\hat{g}(q), \qquad (6.25)$$

gdzie macierze oznaczone daszkiem posiadają nieznane lub przybliżone parametry. Korzystając z liniowości parametrów robota równanie (2.4) można zapisać w formie:

$$\tau=D(q)\ddot{q}+C(q,\dot{q})\dot{q}+F(\dot{q})+g(q)=W(q,\dot{q},\ddot{q})\varphi, \qquad (6.26)$$

gdzie W jest nazywana macierzą regresji, a wektor φ wektorem parametrów. Dodając do (19) sterowanie otrzymujemy:

$$\tau=\hat{D}(q)(\ddot{q}_d-u)+\hat{C}(q,\dot{q})\dot{q}+\hat{F}(\dot{q})+\hat{g}(q), \qquad (6.27)$$

Po wstawieniu (6.8) do (6.21) otrzymujemy:

$$\tau=\hat{D}(q)(\ddot{e}-u)+\hat{D}(q)\ddot{q}+\hat{C}(q,\dot{q})\dot{q}+\hat{F}(\dot{q})+\hat{g}(q)=\hat{D}(q)(\ddot{e}-u)+W(q,\dot{q},\ddot{q})\hat{\varphi}, \qquad (6.28)$$

gdzie φ z daszkiem jest zmienną w czasie estymatą nieznanych parametrów. Równanie (6.22) przekształca się do postaci wiążącej ze sobą błąd sterowania z błędem estymacji. Podstawiając (6.20) do (6.22) otrzymuje się następującą postać:

$$W(q,\dot{q},\ddot{q})\varphi=\hat{D}(q)(\ddot{e}-u)+W(q,\dot{q},\ddot{q})\hat{\varphi}$$

Skąd ostatecznie mamy:

$$\hat{D}^{-1}(q)W(q,\dot{q},\ddot{q})(\varphi-\hat{\varphi})=\ddot{e}-u \,. \tag{6.29}$$

Uzyskujemy układ w klasycznej formie równań stanu:

$$\dot{\boldsymbol{e}}=Ae+B\hat{D}^{-1}(q)W(q,\dot{q},\ddot{q})\tilde{\varphi} \,, \tag{6.30}$$

gdzie:

$$\boldsymbol{e}=\begin{bmatrix} e \\ \dot{e} \end{bmatrix}, \quad B=\begin{bmatrix} \mathbf{0} \\ I \end{bmatrix}, \quad A=\begin{bmatrix} \mathbf{0} & I \\ -K_P & -K_D \end{bmatrix}, \quad \tilde{\varphi}=\varphi-\hat{\varphi} \,.$$

Kluczowym dla równania (6.30) jest sprawdzenie jego asymptotycznej stabilności względem $\tilde{\varphi}$. Aby to uzyskać posłużono się poniższym funkcjonałem Lapunowa:

$$V=\boldsymbol{e}^{\boldsymbol{T}}P\,\boldsymbol{e}+\tilde{\varphi}^{T}\Gamma^{-1}\tilde{\varphi} \,, \tag{6.31}$$

gdzie *P* spełnia Równanie Lapunowa, a *Γ* jest macierzą diagonalną o nieujemnych elementach:

$$A^{T}P+PA=-Q$$
$$\Gamma=diag\left[\gamma_1,\gamma_2,\ldots,\gamma_r\right] \,.$$

Po zróżniczkowaniu (6.31) po czasie i podstawieniu (6.30) otrzymujemy równanie wyrażające prędkość zmiany niepewnych współczynników, nazywane zasadą aktualizacji adaptacyjnej:

$$\dot{\hat{\varphi}}=\Gamma W^{T}(q,\dot{q},\ddot{q})\hat{D}^{-1}(q)B^{T}Pe \,. \tag{6.32}$$

Drugim sposobem regulacji adaptacyjnej jest obserwacja macierzy bezwładności. W celu wyprowadzenia równań współczynników niepewnych oraz prędkości ich zmiany posłużono się równaniem Lapunowa:

$$V=\frac{1}{2}r^{T}M(q)r+\frac{1}{2}\tilde{\varphi}^{T}\Gamma^{-1}\tilde{\varphi} \,, \tag{6.33}$$

gdzie *r* jest funkcją błędu regulacji:

$$r=\dot{e}+\Lambda\, e \,, \tag{6.34}$$

gdzie *Λ* jest symetryczną dodatnio określoną macierzą. W celu wykazania stabilności, zgodnie z twierdzeniem Lapunowa zróżniczkowano *V* względem czasu:

$$\dot{V}=r^{T}M(q)\dot{r}+\frac{1}{2}r^{T}\dot{M}(q)r+\frac{1}{2}\tilde{\varphi}^{T}\Gamma^{-1}\dot{\tilde{\varphi}} \,. \tag{6.35}$$

Rozwiązując równanie dynamiki robota przy założeniu podążania znaną trajektorią i podstawiając (6.28), a następnie rozwiązując ze względu na $\dot{r}$ otrzymujemy:

$$M(q)\dot{r}=Y(\cdot)\varphi-\tau-V_m(q,\dot{q})r \,, \tag{6.36}$$

gdzie:

$$Y(\cdot)\varphi=M(q)(\ddot{q}_d+\Lambda\dot{e})+V_m(q,\dot{q})(\dot{q}_d+\Lambda e)+G(q)+F(\dot{q}) \,.$$

Podstawiając (6.36) do (6.35) otrzymujemy:

$$\dot{V}=r^T(Y(\cdot)\varphi-\tau)+r^T(\frac{1}{2}\dot{M}(q)-V_m(q,\dot{q}))r+\tilde{\varphi}^T\Gamma^{-1}\dot{\tilde{\varphi}}$$

Po zastosowaniu zasad skośnej symetryczności macierzy M oraz V_m powyższe równanie upraszcza się do postaci:

$$\dot{V}=r^T(Y(\cdot)\varphi-\tau)+\tilde{\varphi}^T\Gamma^{-1}\dot{\tilde{\varphi}}, \qquad (6.37)$$

Aby zapewnić stabilność regulatora dobieramy moment sterujący o postaci:

$$\tau=Y(\cdot)\hat{\varphi}+K_v r$$

co sprowadza (6.37) do wyrażenia:

$$\dot{V}=-r^T K_v r+\tilde{\varphi}^T(\Gamma^{-1}\dot{\tilde{\varphi}}+Y(\cdot)r)$$

Wybierając:

$$\dot{\hat{\varphi}}=-\dot{\tilde{\varphi}}=\Gamma Y^T(\cdot)r$$

Jako równanie zmiany współczynników niepewności pochodna równania Lapunowa sprowadza się do prostej postaci:

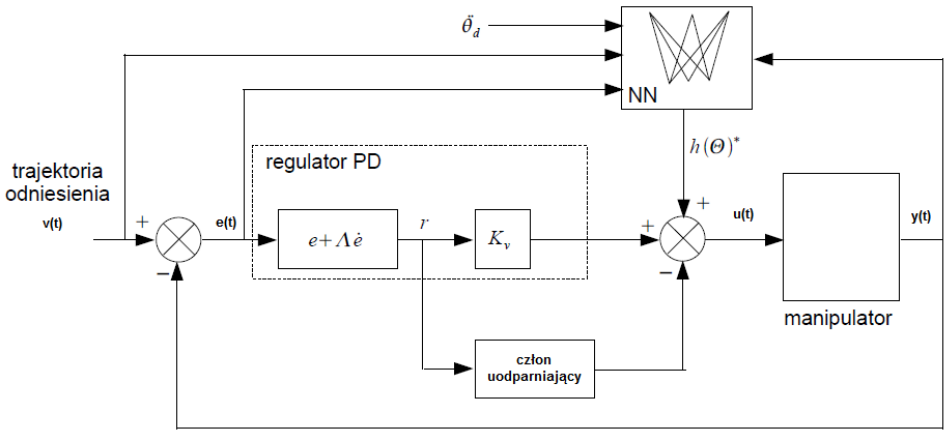
$$\dot{V}=-r^T K_v r \qquad (6.38)$$

Ponieważ K_v może być macierzą dodatnio określoną, równanie (6.38) jest ujemnie półokreślone co zapewnia stabilność układu z regulatorem. Istotną cechą przedstawionego podejścia jest brak wymogu znajomości drugiej pochodnej zmiennych przegubowych robota. Upraszcza to znacząco stosowanie powyższej metody, jako że usuwa ona konieczność pomiaru lub estymacji drugiej pochodnej zmiennych przegubowych robota.

Regulatory neuronowe

W przypadku, gdy model obiektu nie jest znany oraz jest trudny w identyfikacji, sieci neuronowe stanowią interesującą alternatywę dla regulatorów syntetyzowanych klasycznie. Jak wiadomo klasyczne podejście wymaga znajomości pełnego modelu. [38], [43], [69]

Przy znajomości pełnego stanu obiektu, statyczna sieć neuronowa ze sprzężeniem w przód może generować sterowania nadążające za trajektorią. Zwykle jednak sieć neuronowa jest używana jako człon linearyzujący lub generujący dodatkowe wymuszenia. Standardowym podejściem jest użycie regulatora PD jako głównego członu minimalizujący błąd sterowania. Na rysunku 6.3 przedstawiono przykładową strukturę regulatora PD wspomaganego siecią neuronową. [47]



Rys. 6.3 Schemat ideowy regulatora PD z dodaną siecią neuronową

Podobnie jak w przypadku regulatora PD zapisujemy błąd śledzenia jako:

$$\begin{aligned} e(t) &= q_d(t) - q(t) \\ r &= \dot{e} + \Lambda\, e \end{aligned} \qquad (6.39)$$

gdzie Λ jest symetryczną dodatnio określoną macierzą. Zwykle wybieraną jako macierz przekątna. Pomiędzy błędem e oraz błędem filtrowanym r zachodzą zależności:

$$\begin{aligned} \|e\| &\le \|r\| / \lambda_{min}(\Lambda) \\ \|\dot{e}\| &\le \|r\| \end{aligned}$$

Podstawiając (6.39) do równania dynamiki robota otrzymujemy:

$$M\dot{r} = -V_m r - \tau + f + \tau_d$$

gdzie część nieliniowa zapisana jest jako:

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + G(q) + F(\dot{q})$$

Ponieważ zawiera ona w sobie nieznane elementy dotyczące mas, momentów bezwładności oraz współczynników tarcia, zostanie zastąpiona przez aproksymację generowaną przez sieć neuronową. Wówczas, przy pomocy metody wyznaczonego momentu możemy określić sterowanie robota jako:

$$\tau = \hat{f} + K_v r - v$$

Niech v będzie sterowaniem pochodzącym od członu uodparniającego, Jest ono niezbędne w wypadku wystąpienia nieoczekiwanych, niezamodelowanych zakłóceń. Może ono zostać dobrane w różnoraki sposób. Równanie dynamiki zamkniętego układu regulacji z siecią neuronową w roli aproksymatora oraz regulatorem PD wygląda następująco:

$$M\dot{r} = -(K_v + V_m)r + \hat{f} + \tau_d + v$$

Zależnie od przyjętego typu sieci neuronowej sterowanie robota może przyjąć postać:

$$\tau = \hat{W}^T \phi(x) + K_v r - v$$

dla sieci LIP lub:

$$\tau = \hat{W}^T \sigma\left(\hat{V}^T x\right) + K_v r - v$$

dla sieci NLIP. W zależności od przyjętego typu sieci neuronowej oraz od typu użytej funkcji wyjściowej sieć neuronowa powinna być uczona w odpowiedni sposób. Bardzo ważnym elementem algorytmów uczących jest to, że nie muszą one działać na wcześniej przygotowanych danych. Mogą one modyfikować wagi sieci w trakcie działania algorytmu sterującego, stopniowo poprawiając jakość regulacji. [47], [58]

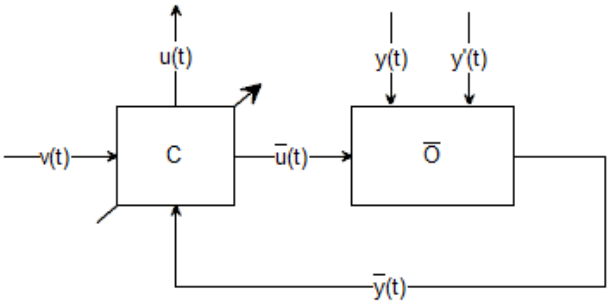
7. Nieliniowe regulatory predykcyjne

Sterowanie predykcyjne (ang. Model Predictive Control) jest zaawansowaną metodą sterowania. Polega ono na generowaniu sterowania w oparciu o cykliczne rozwiązywane zadanie optymalizacji, które wymaga ciągłego przewidywania odpowiedzi obiektu na podstawie jego modelu. W swoim założeniu są zatem w stanie prognozować i reagować na błąd sterowania jeszcze przed jego faktycznym zaistnieniem. Z tej przyczyny algorytmy predykcyjne winny być szeroko stosowane w automatyce przemysłowej.

Sterowanie predykcyjne można podzielić na dwie podstawowe grupy: sterowanie predykcyjne liniowe oraz nieliniowe (ang. Nonlinear Model Predictive Control). Sterowanie liniowe ma wiele zastosowań [62], jednak ze względu na nieliniowy charakter obiektów manipulatorów-robotów, w pracy szerzej zostanie opisany wariant nieliniowy. W szczególności zostaną opisane metody numeryczne niezbędne do wyznaczania przyszłego stanu obiektu.

Nieliniowe MPC

Na rysunku 7.1 przedstawiono schemat nieliniowego algorytmu predykcyjnego. Jako *v*(*t*) oznaczono zaplanowaną trajektorię, jako *C* oznaczono regulator (ang. Controller), który wykonuje predykcję na zadanym horyzoncie poprzez wielokrotne próbowanie sterowania $\bar{v}(t)$ na model obiektu *O*, wynik działania predykcyjnego oznaczono jako $\bar{y}(t)$. Przez *y(t)* oraz *y'(t)* oznaczono rzeczywiste wyjście z obiektu.



Rys. 7.1 Schemat ideowy regulatora predykcyjnego

Powyższy kontroler wykonuje kolejne kroki algorytmu:

1. Pomiar zmiennych przegubowych
2. Predykcja na zadanym horyzoncie czasowym przy użyciu modelu oraz całkowania numerycznego
3. Wyznaczenie sterowania
4. Wykonanie zaplanowanego sterowania
5. Powrót do kroku pierwszego

W dalszej części rozdziału opisano szczegółowo kolejne bloki regulatora.

Predykcja

Blok predyktora jest zasadniczo modelem sterowanego obiektu. Danymi niezbędnymi do działania bloku predyktora są próbkowane sterowanie oraz dane pomiarowe zmiennych stanu z obiektu rzeczywistego. Pomiary są niezbędne do poprawnego wystartowania modelu. Predykcja trajektorii obiektu jest dokonywana przy pomocy standardowego modelu matematycznego robota:

$$\ddot{q}_i = D^{-1}(q_i)\left(\bar{u}_i - C(q_i,\dot{q}_i)\dot{q}_i - g(q_i) - f(q_i,\dot{q}_i)\right)$$

Obliczenia są prowadzone na zadanym uprzednio horyzoncie.

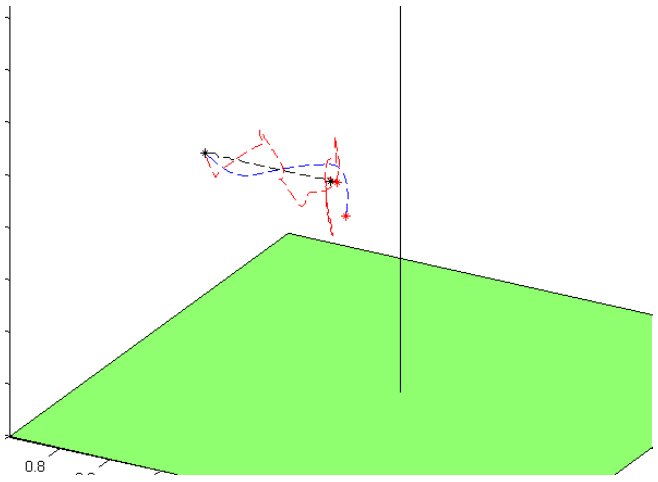
Predykcja numeryczna

Równanie jest dwukrotnie numerycznie całkowane. Dla algorytmu Eulera otrzymuje się:

$$q_{i+1} = q_i + \left(\dot{q}_i + \ddot{q}_i \cdot d_{sim}\right) \cdot d_{sim}, \qquad (7.1)$$

gdzie $q_1 = y(t), \dot{q}_1 = \dot{y}(t)$ i $i \in 1 \ldots H$.

Eksperymenty dotyczą robota-manipulatora typu stanfordzkiego pokazanego na rysunku 7.2. Na rysunku poniżej pokazano, jak przewidywalne jest działanie dwóch klasycznych regulatorów. Mają one za zadanie podążać za aproksymowaną trajektorią prostoliniową:

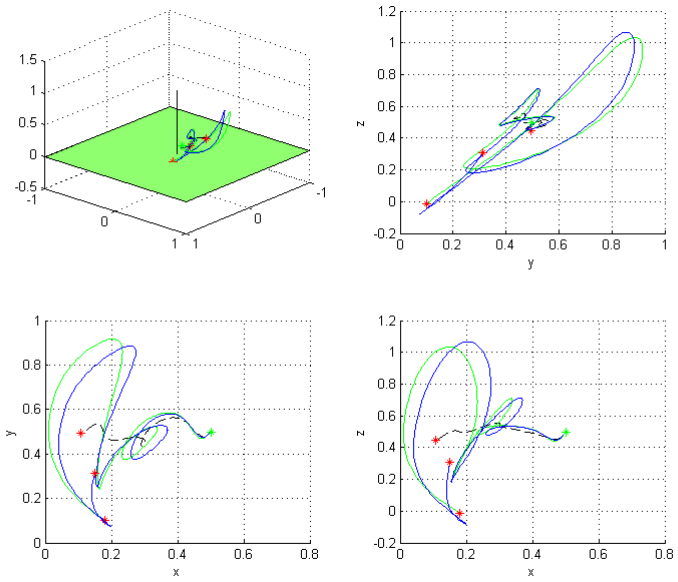


Rys. 7.2 Porównanie predykcji dla różnych metod całkowania numerycznego

Czarną przerywaną linią zaznaczono wygenerowaną zadaną trajektorię, na niebiesko ruch końcówki manipulatora sterowanego przy pomocy regulatora PD+, na czerwono narysowano ruch robota sterowanego regulatorem PD+NN. Powyższe trajektorie zostały wyrysowane przy pomocy stało krokowej metody całkowania numerycznego Dormanda-Prince'a (ode5) o długości kroku 0,001. Dobór metody całkowania numerycznego jest bardzo istotny z uwagi na wysoce nieliniowy charakter badanego modelu.

Na rysunku 7.3 można zaobserwować dla modelu matematycznego ramienia stanfordzkiego jak bardzo rozwiązania numeryczne mogą różnić się od siebie. Czarną linią oznaczono trajektorię końcówki obliczoną przy pomocy metody Runge-Kutta piątego rzędu (ode5) z krokiem 0,001. Na niebiesko oznaczono trajektorię obliczoną przy pomocy metody Runge-Kutta 4 (ode4), zaś na zielono trajektorię wykreśloną metodą Bogacki-Shampine (ode3)

o długości kroku 0,01. Horyzontem czasowym całkowania numerycznego były 4 sekundy, czyli czas krótszy niż wymagany do osiągnięcia końca zadanej trajektorii liniowej. Metoda ode5 była traktowana jako referencyjna, z tego powodu jej krok był dziesięciokrotnie krótszy od kroku pozostałych metod.



Rys. 7.3 Porównanie predykcji dla różnych metod całkowania numerycznego

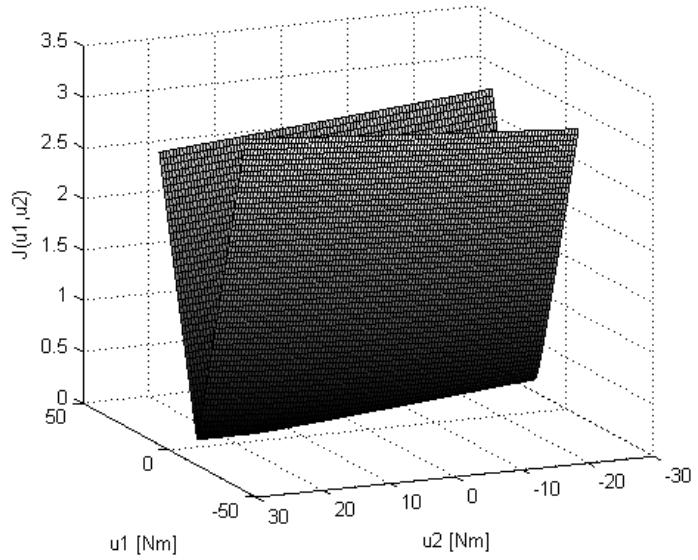
Różne metody całkowania numerycznego są badane pod kątem późniejszego zastosowania w algorytmach predykcyjnych [5].

Optymalizacja

Dokonywana jest na podstawie predykcji. Do jej przeprowadzania niezbędne jest określenie wskaźnika jakości, który odzwierciedla cel sterowania. Takim wskaźnikiem przy braku ograniczeń na sterowanie może być błąd średniokwadratowy na horyzoncie *h*:

$$J(u_1, u_2, h) = \int_{t=0}^{h} |q(t) - v(t)|^2 dt \qquad (7.2)$$

W celach ilustracyjnych na rysunku 7.4 pokazano wykres wskaźnika jakości dla mechanizmu dwuczłonowego. Mechanizm ten wybrano ze względu na możliwość narysowania funkcji celu w trzech wymiarach.



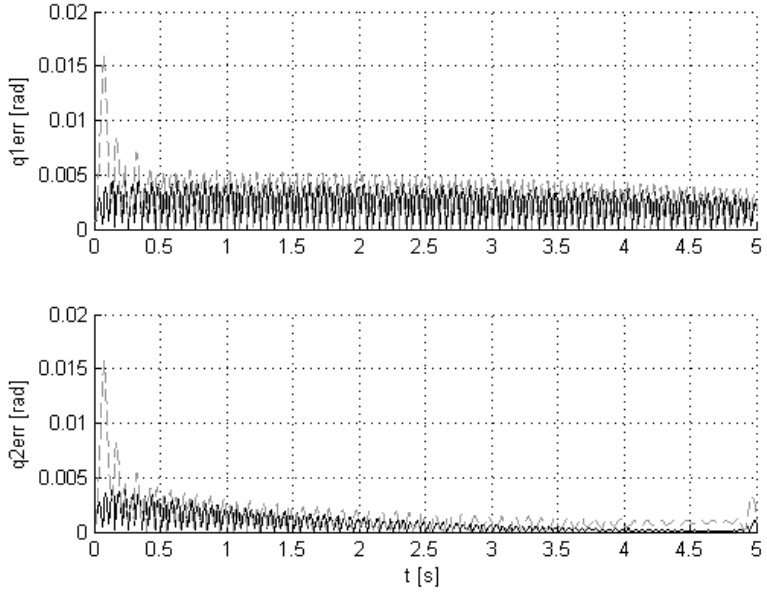
Rys. 7.4 Dolina formowana przez wskaźnik jakości dla mechanizmu dwuczłonowego

Aby znaleźć optymalne sterowanie $u(t)$ w sensie zaproponowanego wskaźnika jakości, posłużono się drugim wariantem metody Powella. Metoda Powella jest bezgradientową metodą poszukiwania minimum w zbiorze rozwiązań. Opisał ja w 1964 Michael Powell w pracy [57]. Korzystając z wniosków z pracy [4], możemy przyjąć, że znalezienie nawet niedokładnego minimum funkcji celu może prowadzić do wyznaczenia skutecznego sterowania.

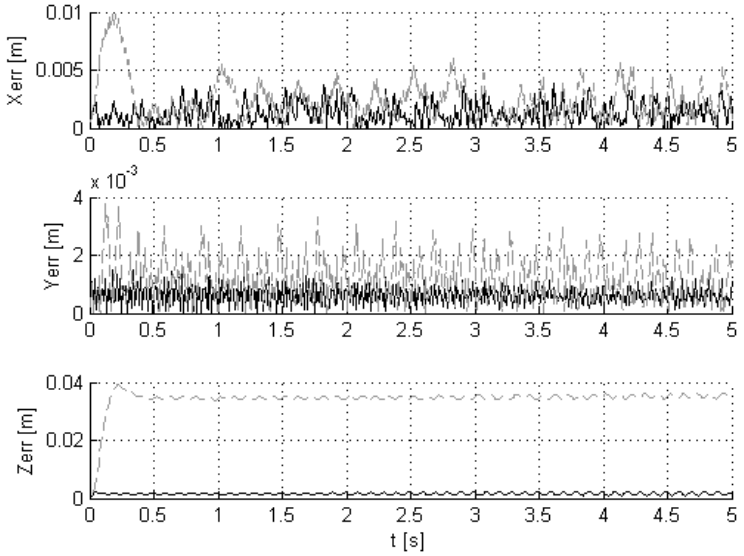
Inna przydatną cechą metody Powella jest jej iteracyjność. Pozwala ona na przerwanie poszukiwania minimum w dowolnym momencie działania, a nie tylko w wypadku znalezienia minimum. Jako że metoda ta jest iteracyjna, każdy kolejny jej krok przybliża algorytm do znalezienia lepszego rozwiązania. Jednak w wypadku, gdy poszukiwania trwają zbyt długo, tzn. model w bloku predyktora został wywołany zbyt wiele razy, algorytm może zostać zakończony w miejscu suboptymalnego rozwiązania. Umożliwia to ograniczenie czasu wykonywania pojedynczego kroku sterowania, co jest niezwykle istotne w systemach sterowania czasu rzeczywistego.

Jako wejście do metody służy zaplanowana ścieżka oraz przewidywana droga, którą porusza się robot. Poszukiwanie minimum na prostej odbywa się metodą złotego podziału. Podejście powyższe zostało opisane między innymi w pracy [82], gdzie przedstawione zostały wyniki eksperymentów symulacyjnych dla mechanizmu dwuczłonowego oraz ramienia stanfordzkiego. Rezultaty działania dla ramienia stanfordzkiego zostały przedstawione na rysunkach 7.5 oraz 7.6.

Na rysunkach można zauważyć, że horyzont, na jakim dokonuje się predykcji, nie pozostaje obojętny na wyniki sterowania. W obu wypadkach akcja sterowania była osłabiona, co skutkowało większym błędem na początku eksperymentu. W wypadku ramienia stanfordzkiego wydłużenie horyzontu predykcji skutkowało wysokim błędem ustalonym wzdłuż osi Z.



Rys. 7.5. Wykres błędu nadążnego dla mechanizmu dwuczłonowego w przestrzeni zmiennych przegubowych. Linią ciągłą zaznaczono ruch z predykcją na horyzoncie 20 ms i kroku 5 ms, linią przerywaną zaznaczono ruch z horyzontem 90 ms i krokiem 30 ms



Rys. 7.6 Wykres błędu nadążnego dla ramienia Stanfordzkiego w przestrzeni zmiennych kartezjańskich. Linią ciągłą zaznaczono ruch z predykcją na horyzoncie 20 ms i kroku 5 ms, linią przerywaną zaznaczono ruch z horyzontem 90 ms i krokiem 30 ms

Generacja funkcji kary z ominięciem przeszkody

Jednym z ciekawych efektów ubocznych sterowania przy pomocy minimalizacji funkcji kary jest możliwość dodania dodatkowych współczynników lub modyfikacja samej funkcji, podczas trwania ruchu. Pozwala to na omijania przeszkód. Zagadnieniu temu, wraz z doborem

odpowiednich metod wprowadzania przeszkód do funkcji kary, poświęcono wiele prac [2], [12], [17], [22], [23], [25], [26], [34], [35], [36], [37], [44], [83].

Zaproponowany w (7.2) wskaźnik jakości uwzględniał jedynie karę uzależnioną od położenia końcówki robota. W zastosowaniach praktycznych istotnymi elementami są również kary uzależnione od prędkości poszczególnych członów manipulatora oraz od przecięcia położenia końcówki lub dowolnego członu robota z obszarem zajętym przez inny obiekt, inaczej nazwany przeszkodą. Aby uwzględnić powyższe oryginalne równanie, można poszerzyć do następującej formuły:

$$f_p(t)=f_q(x(t))+f_R(x(t))+f_o(x(t)), \tag{7.3}$$

gdzie $f_q(x(t))$ jest częścią kary od położenia, $f_R(x(t))$ jest częścią kary od ograniczeń prędkościowych, $f_o(x(t))$ jest częścią wyznaczaną na podstawie wzajemnego położenia robota oraz przeszkód.

Funkcja kary od położenia może być wyrażona tak jak w (7.2), jeśli steruje się prędkością lub momentem. Jeśli zakładamy, że sterowanie jest uproszczone do sterowania PTP, wówczas funkcja przyjmuje prostszą formę zależną jedynie od przewidywanego położenia końcowego:

$$f_q=\Theta\left(\bar{x}(t_n)-x(t_n)\right)^2, \tag{7.4}$$

gdzie t_n jest oczekiwanym czasem zakończenia n-tego kroku sterowania, $x(t_n)$ jest planowanym, wynikającym z zaplanowanej trajektorii robota położeniem w czasie t_n, a $x(t_n)$ jest osiągniętym położeniem w t_n.

Funkcja kary od ograniczeń robota zawiera w sobie karę za przekroczenie maksymalnych prędkości dopuszczalnych dla manipulatora. Podobnie jak dla $\bar{x}$ tak dla $f_R(x(t))$ można zastosować dwie równoważne reguły wyznaczania kary. Dla sterowania PTP reguła może przyjąć następującą postać:

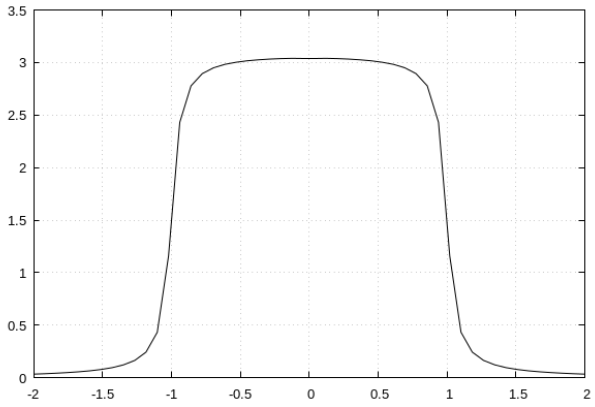
$$f_R(x(t))=\xi\left(\frac{x(t_n)-x(t_{n-1})}{\rho_v}\right)^4, \tag{7.5}$$

kara wyliczana jest na podstawie ostatecznego położenia członów pod koniec pojedynczego cyklu sterowania, gdzie położenie początkowe oznaczono jako $x(t_{n-1})$, a końcowe jako $x(t_n)$. Całość jest skalowana przez współczynnik ρ_v zależny od prędkości maksymalnych dla każdej osi robota.

Funkcja kary dla przeszkód powinna stanowić dla algorytmu minimalizującego trudną do przekroczenia barierę. Jej charakterystycznymi cechami są strome zbocza oraz niska wartość, najlepiej zerowa, w przestrzeniach nie zajętych przez przeszkody. Przykładem takiej funkcji może być suma dwóch arcus tangensów przedstawiona poniżej:

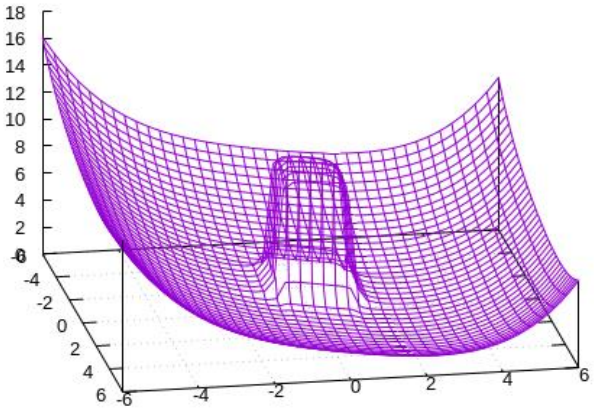
$$f_O(x(t))=\sum_{n=0}^{N}\prod_{i=1}^{dim(x)}\arctan\left(\xi_{in}\left(x(t)-(\bar{x}_{iO}-\bar{r}_{iO})\right)\right)-\arctan\left(\xi_{in}\left(x(t)-(\bar{x}_{iO}+\bar{r}_{iO})\right)\right), \tag{7.6}$$

przykładowy przebieg funkcji (7.6) przedstawiono na rysunku 7.7. W celu uproszczenia przyjęto, że przestrzeń jest jednowymiarowa.



Rys. 7.7 Przykładowy przebieg funkcji sumy dwóch arcus tangensów da parametrów $N=1, \xi=20, x_O=0, r_O=1$

Na rysunku 7.8 przedstawiono płaszczyznę funkcji kary dla hipotetycznego mechanizmu dwuczłonowego. Można na niej wyróżnić kilka kluczowych elementów. Po środku znajduje się hipotetyczna przeszkoda. Jest to nagle występująca nierówność. W jej prawym dolnym rogu można zaobserwować minimum błędu, w miejscu, gdzie powinien się znaleźć punkt końcowy tego segmentu ruchu. Wokół niego można zaobserwować koliste poziomice pochodzące od części kary za błąd ruchu. W końcu na brzegach wykresu widoczny jest gwałtowny wzrost funkcji kary pochodzący od ograniczeń robota na maksymalną prędkość.



Rys. 7.8 Przykładowa płaszczyzna generowana przez funkcję kary dla mechanizmu dwuczłonowego

Algorytm unikania kolizji

Unikaniu kolizji poświęcono wiele prac, m.in. [63] czy [68]. Zwykle problemowi omijania przeszkód towarzyszy analityczne podejście. Jednakowoż, ze względu na charakter kontrolera wykorzystującego dynamiczną optymalizację numeryczną, możliwy jest unik przed przeszkodą bez konieczności wyznaczania uprzednio trajektorii. Zadanie unikania kolizji jest uproszczone do podstawowego zadania minimalizacji wskaźnika jakości (7.2) Przez proste

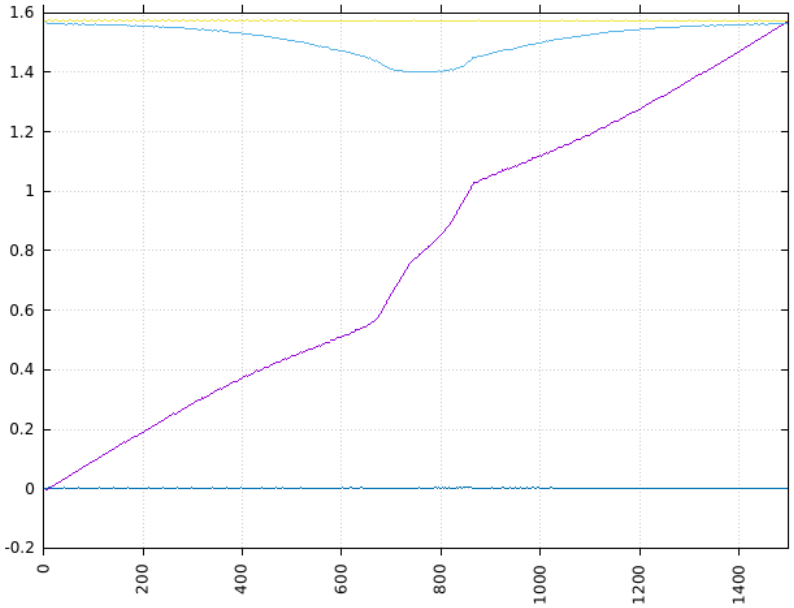
wstawienie wykrytej funkcji kary do obiektu możliwe jest stworzenie algorytmu reagującego w trybie on-line na zmiany środowiskowe.

Algorytm składa się z następujących kroków:

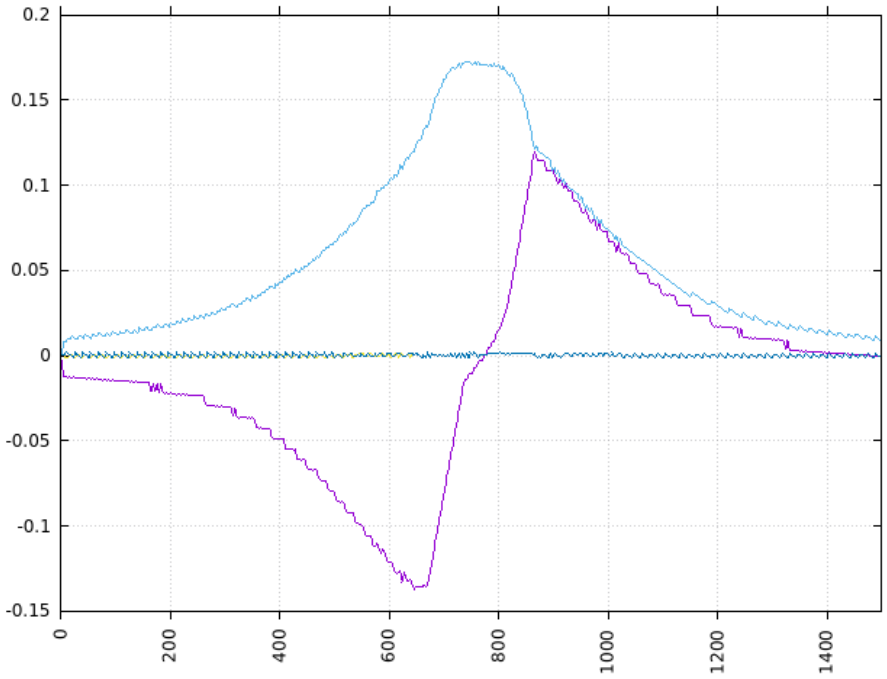
1. Pobranie aktualnej pozycji manipulatora $\varphi(t)$
2. Obliczenie następnego punktu planowanej trajektorii $\varphi_d(t+1)$
3. Znalezienie punktu z najniższą możliwą karą przy użyciu metody Powella
4. Generowanie sterowania.

Ilustracją zadania są dwa przykłady. W pierwszym zadana trajektoria to monotonicznie zmieniane kątowe położenie pierwszego przegubu od 0 do π/2 radianów w ciągu 1500 ms. Dwie przeszkody o środkach w: $o_1:\left\{\frac{\pi}{4},0,\frac{\pi}{1.9},0,\frac{\pi}{2.0},0\right\}$, $o_2:\left\{\frac{\pi}{4},0,\frac{\pi}{1.8},0,\frac{\pi}{2.0},0\right\}$, o promieniu: $\left\{\frac{1}{15}\pi,\frac{1}{15}\pi,\frac{1}{18}\pi,\frac{1}{18}\pi,\frac{1}{18}\pi,\frac{1}{18.0}\pi\right\}$, są włączone w ścieżkę ruchu.

Na poniższych rysunkach przedstawiono położenia oraz błąd powstający w trakcie manewrów wymijania. Na rysunkach 7.9 oraz 7.10 przedstawiono położenie oraz błąd towarzyszące pierwszej trajektorii.



Rys. 7.9 Zmienne przegubowe robota Mitsubishi RV-2F-D podczas manewru omijania przeszkód. Ruch w jednym przegubie

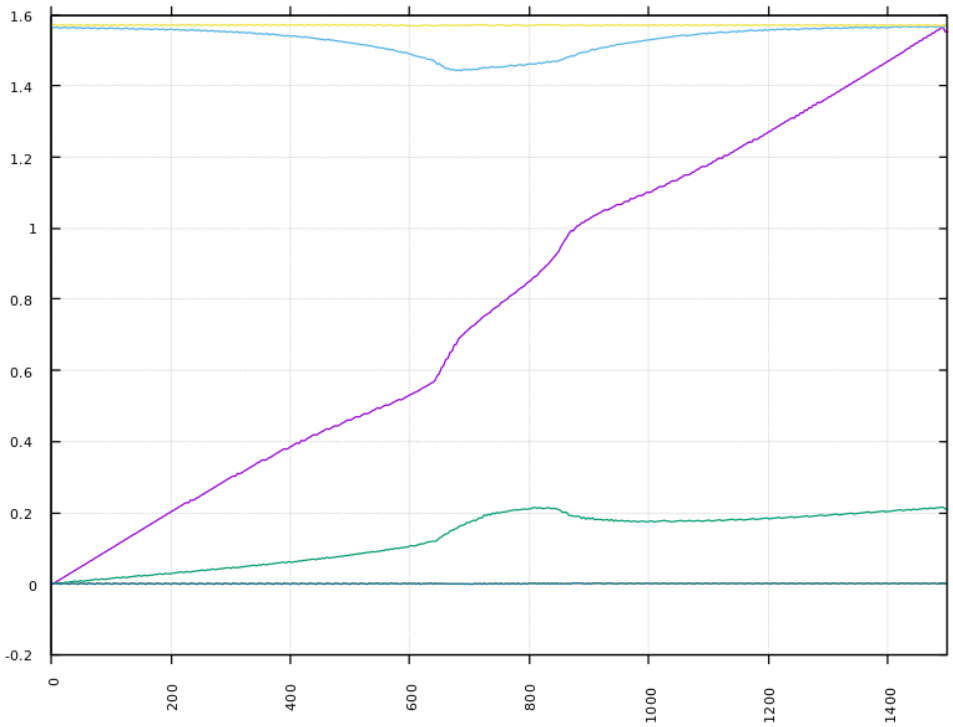


Rys. 7.10 Błąd na poszczególnych przegubach robota podczas manewru wymijania. Ruch w jednym przegubie

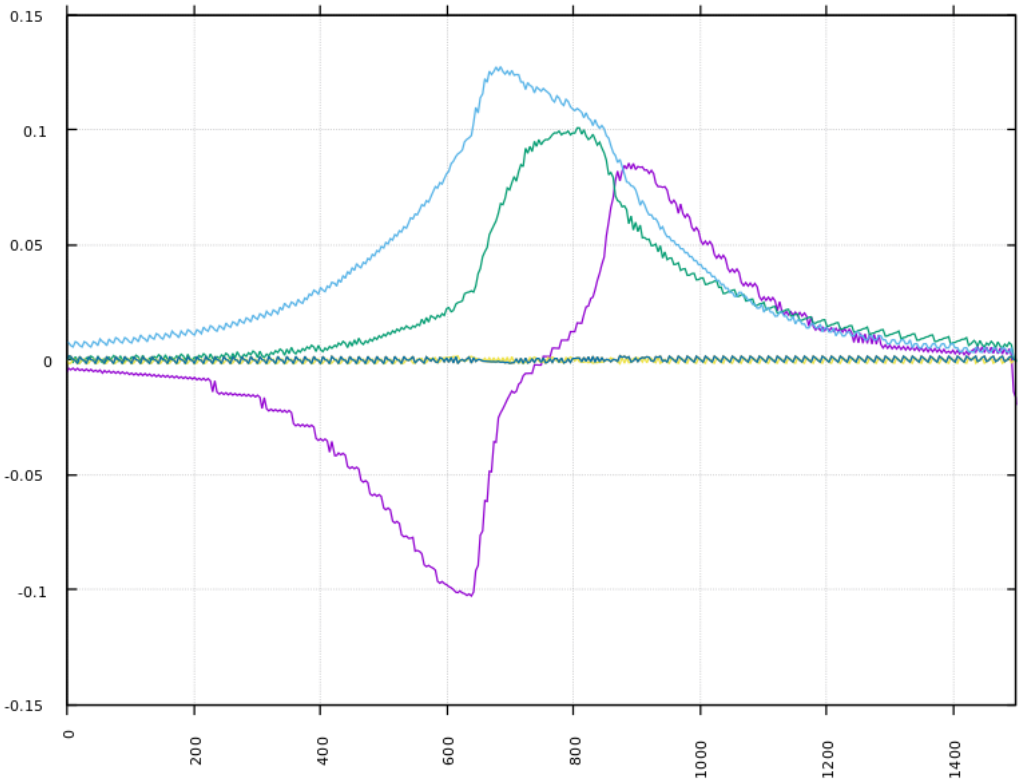
Można zaobserwować, że regulator podążał za zadaną trajektorią, ale błąd narastał, do momentu mijania szczytu przeszkody. Po ominięciu przeszkody robot był w stanie powrócić do śledzenia zadanej trajektorii.

Druga trajektoria jest wzmocnioną zmianą monotoniczną drugiego przegubu pozycji od 0 do π/15. Te same przeszkody są obecne na ścieżce ruchu jak w pierwszym przykładzie.

Na rysunku 7.11 można zaobserwować, że pomimo braku zmiany w algorytmie robot był w stanie ominąć przeszkodę przy pomocy modyfikacji położenia kolejnej osi. Dodatkowo na wykresie 7.12 możemy zaobserwować, że podczas manewru wymijania błąd na pozostałych osiach był mniejszy niż w poprzednim przypadku. Pozwala to wysnuć wniosek, że algorytm poprawnie zminimalizował koszt sterowania i ominął przeszkodę przy minimalnym błędzie.

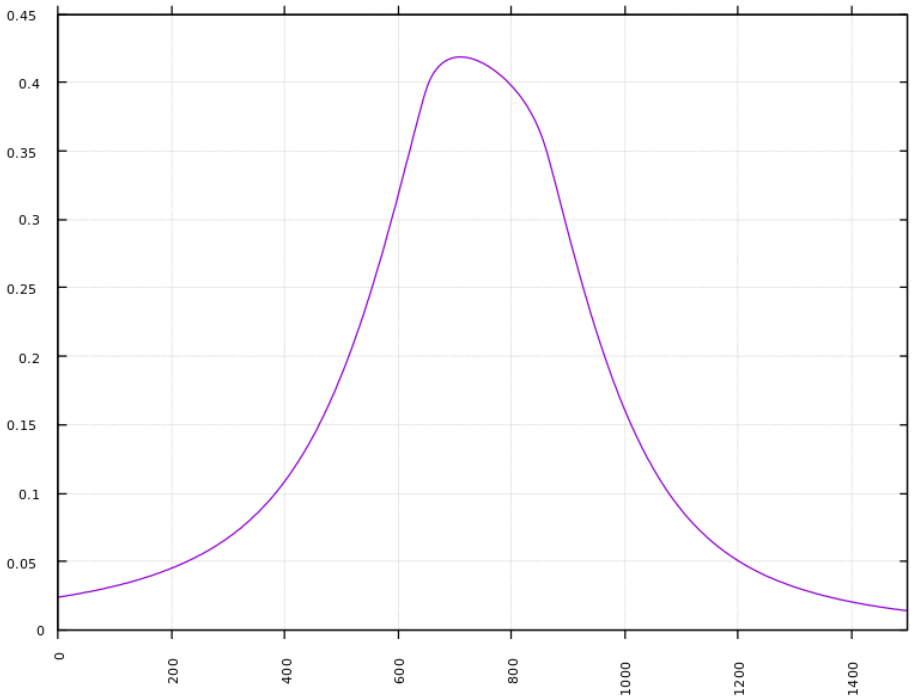


Rys. 7.11 Zmienne przegubowe robota Mitsubishi RV-2F-D podczas manewru omijania przeszkód. Ruch w dwóch przegubach



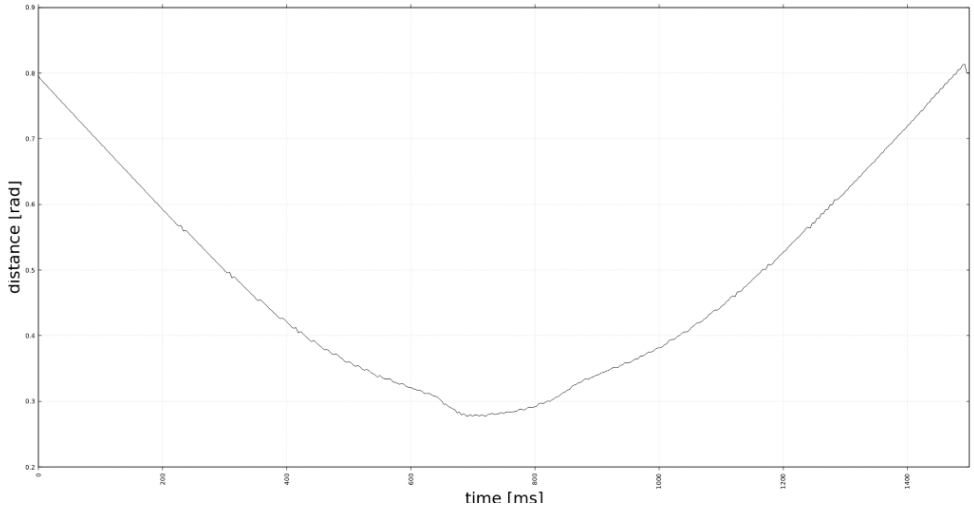
Rys. 7.12 Błąd na poszczególnych przegubach robota podczas manewru wymijania. Ruch w dwóch przegubach

Na rysunku 7.13 przedstawiono, w jaki sposób zachowuje się funkcja kary podczas manewru ominięcia przeszkód. Można zaobserwować, że kara zaczyna rosnąć jeszcze przed zbliżeniem do przeszkody, po ominięciu kara maleje, aby dojść do niskiego poziomu na końcu ruchu. Jej zachowanie koresponduje z kształtem funkcji błędu na poszczególnych osiach. Można zaobserwować, że jej maksimum przypada tam, gdzie błąd kątowy był największy, czyli w momencie ominięcia przeszkody.



Rys. 7.13 Wykres funkcji kary podczas manewru ominięcia przeszkody

Jednocześnie przedstawiając wykres odległości od przeszkody 7.14, można zaobserwować, że końcówka zawsze zachowuje bezpieczną odległość od przeszkód.

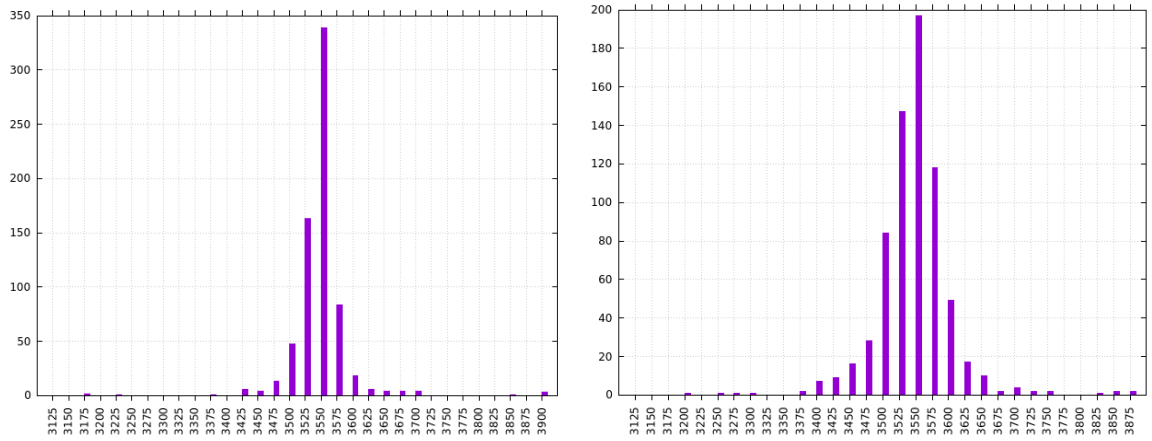


Rys. 7.14 Odległość końcówki robota od przeszkody.

Kod źródłowy poszukiwania minimum funkcji kary oraz jej przedstawienie w języku C++ zamieszczono w dodatku D.

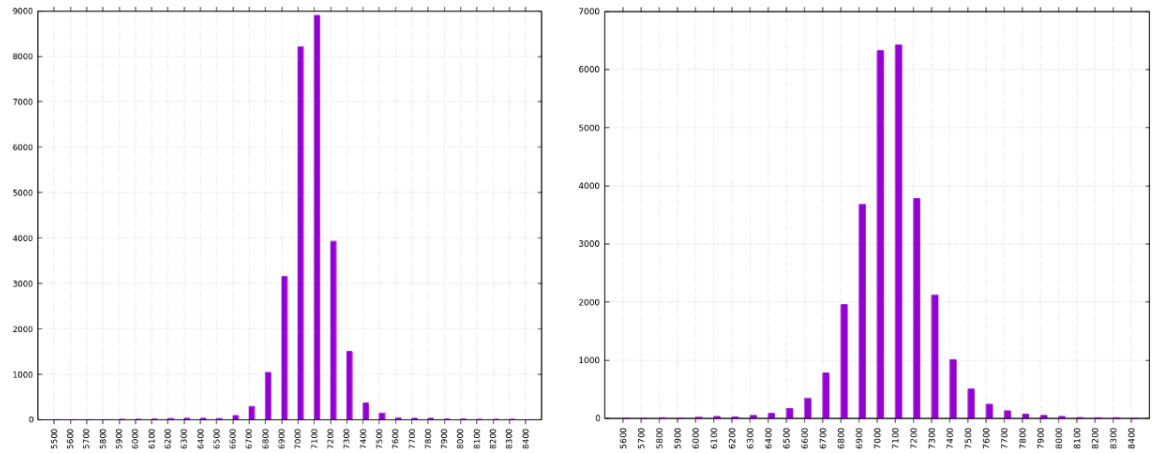
Analiza czasowa

Bardzo ważnym aspektem algorytmów sterujących pracą robota jest, aby te były wykonywalne w stabilny czasowo sposób. W poniższym rozdziale przedstawiona zostanie analiza czasowa działania algorytmu. Sterowanie było realizowane na dwóch różnych platformach sprzętowych. Na komputerze klasy PC wyposażonym w Intel Core i7-6700HQ taktowany 2.6 GHz oraz 16 GB pamięci operacyjnej RAM. Systemem operacyjnym był Linux o jądrze 4.15.0-47 oraz na mikrokomputerze Raspberry PI B w wersji 3 pod kontrolą systemu Linux 4.4.50-rt63. W wypadku mikrokomputera sterowanie było realizowane w cyklu 7 ms, zaś w wypadku komputera klasy PC w cyklu 3,5 ms. Wykresy na rysunkach 7.15 przedstawiają jitter procesu sterującego oraz procesu nadzorującego robota na komputerze PC.



Rys. 7.15 Jitter procesu sterującego dla komputera klasy PC przy jednoczesnym przetwarzaniu obrazów z dwóch kamer w celu uzyskania położenia przeszkody

Na rysunkach 7.16 przedstawiono jittery procesów na mikrokomputerze Raspbery PI.



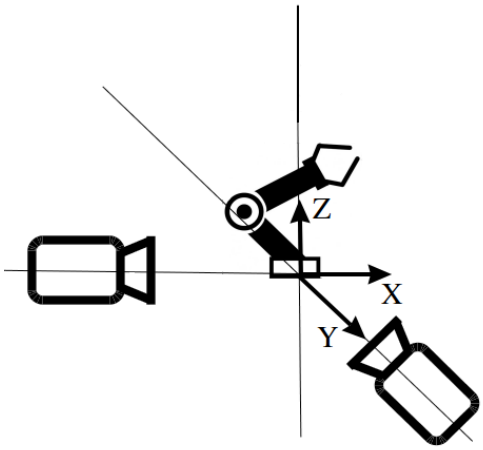
Rys. 7.16 Jitter procesu sterującego dla mikrokomputera Raspberry PI przy jednoczesnym przetwarzaniu obrazów z dwóch kamer w celu uzyskania położenia przeszkody

W obu przypadkach można zauważyć, że maksymalne opóźnienie próbki sterowania nie przekracza 400 us. Jest to spowodowane iteracyjnością metody Powella, co było opisane w rozdziale o poszukiwania rozwiązania optymalnego. Ta cecha pozwala na ograniczenie czasu wykonywania pojedynczego kroku sterowania.

8. Systemy wizyjne

Rozpoznawanie położenia obiektów w przestrzeni roboczej stanowi jeden z najbardziej interesujących i jednocześnie jeden z trudniejszych aspektów badawczych w robotyce. Dzięki rozpoznawaniu położenia robota oraz obiektów trzecich generowana jest reakcja na zdarzenia niezależne od systemu robota, na przykład reakcja na pojawienie się przeszkód wewnątrz pola pracy lub reakcja na interwencję operatora.

Obserwacja przestrzeni roboczej odbywa się wieloetapowo. Najpierw przy pomocy krawędzi oraz barw wyznaczone są położenia obiektów. Następnie dla wszystkich kamer wyznaczane są położenia elementów wspólnych. Przy ich pomocy tworzona jest przestrzenna reprezentacja pola roboczego i przeszkód w nim obecnych. Na końcu wyznaczana jest funkcja kary dla regulatora, który ma za zadanie przeprowadzić robota przez zmieniające się otoczenie.



Rys. 8.1 Dwie kamery na dwóch osiach układu odniesienia

Budowa systemu wizyjnego

Proponowany w pracy system wizyjny składa się z trzech kamer o osiach prostopadłych do siebie. Kamery podczas eksperymentów działały w dwóch trybach pracy:

1. Rozdzielczości: 320x240 w trybie 50 klatek na sekundę (jedna klatka co 20 ms)
2. Rozdzielczości: 160x120 w trybie 100 klatek na sekundę (jedna klatka co 10 ms)

Te dwa tryby zostały wybrane jako odpowiednio: najwyższa możliwa jakość obrazu, najkrótszy czas odpowiedzi kamery.

Trzy kamery ustawione w powyższy sposób umożliwiają obserwację całego otoczenia roboczego robota. Pozwala to na wyznaczanie położenia obiektu w przestrzeni trójwymiarowej. Dołożenie kolejnej kamery może poprawić wyniki pracy algorytmu ze względu na redukcję ilości martwych stref pola widzenia. Położenie kamer zostało przedstawione na rysunku 8.1. Kamery są umieszone odpowiednio: nad, za oraz z prawej strony przestrzeni roboczej.

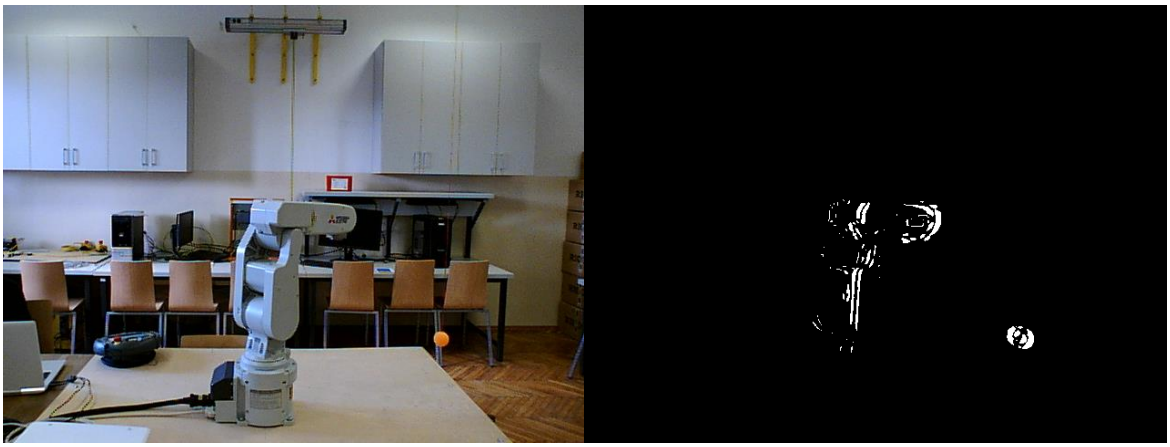
Detekcja obiektów

W celu zbudowania trójwymiarowego modelu sceny konieczne jest zbudowanie hierarchii obiektów znajdujących się w obrazach zarejestrowanych przez kamery. Pierwszym krokiem jest wykrycie nieruchomego tła oraz zmniejszenie jego istotność w dalszym przetwarzaniu. Kolejnym zadaniem jest wyodrębnienie obiektów znanych i łatwych w detekcji, takich jak stół czy ramię robota. Ostatecznie można przejść do wykrywania obiektów będących w ruchu.

Usuwanie tła

Usunięcie niezmiennego tła z obrazu pozwala na zmniejszenie ilości obliczeń w późniejszych krokach zaawansowanych algorytmów przetwarzania obrazu, które muszą klasyfikować oraz identyfikować wykryte punkty charakterystyczne. Zmniejszenie ilości przetwarzanych detali pozwala na redukcję czasu wykonywania niektórych algorytmów nawet o kilka rzędów wielkości.

Najprostszym sposobem na usunięcie elementów otoczenia jest usunięcie wszystkich nieporuszających się obiektów z obrazu. Jedną z możliwości przeprowadzenia takiej operacji jest odjęcie od siebie kolejnych klatek obrazu. Na rysunku 8.2 przedstawiono wynik takiej operacji. Po lewej przedstawiono statyczny oraz zaś po prawej wynik odjęcia między dwiema następującymi po sobie klatkami. Metoda ta była przedstawiona w [75].



Rys. 8.2 Usunięcie tła przy pomocy prostego odejmowania kolejnych klatek

Metoda ta jest prosta w implementacji oraz stosunkowo szybka. Ma również szereg wad, takich jak wrażliwość na nagłe zmiany oświetlenia oraz wrażliwość głównie na kontury obiektów poruszających się. Sposoby minimalizacji powyższych problemów opisano m.in. w [40].

Innymi sposobami usuwania tła są metody oparte o filtrowanie. Ich działanie polega na cyklicznym obliczaniu średniej z koloru pikseli, a następnie usuwanie tych, które się nie zmieniają. Dotykają ich jednak podobne problemy, jak poprzednio opisany algorytm.

Najbardziej zawansowane metody polegają na uproszczonym modelowaniu zachowania tła. Interesujący jest opisany w [33] algorytm oparty o probabilistyczny Gaussian Mixture Model opisujący zachowanie pikseli tła obrazu.

Punkty charakterystyczne

Jako wejście do algorytmu detekcji obiektów kamera dostarcza dwuwymiarowy barwny obraz sceny. Obraz zawiera w sobie wiele nadmiarowych informacji, które spowalniają oraz utrudniają wykrywanie obiektów na scenie. W celu minimalizacji danych podczas przetwarzania w obrazie wyszukuje się tzw. punkty charakterystyczne (ang. keypoints) oraz ich łatwo opisywalne cechy (ang. features). Większość algorytmów wykrywania cech obrazów bazuje na trzech podstawowych grupach:

1. Narożnikach (ang. corners)
2. Krawędziach (ang. edges)
3. Kleksach (ang. blobs)

Detekcja narożników

W literaturze pojawia się wiele algorytmów dedykowanych wykrywaniu narożników w obrazie. W pracy rozpatrzone zostaną trzy algorytmy: Harrisa, SUSAN (ang. smallest univalue segment assimilating nucleus) oraz FAST (ang. features from accelerated segment test) należące do grupy algorytmów często używanych.

Algorytm Harrisa (będący rozszerzeniem algorytmu Moraveca) bada różnicę jasności danego piksela w stosunku do pikseli w jego sąsiedztwie, które określono w oknie analizy. Powyższe sprowadza się do obliczenia wyniku równania (8.1) dla każdego piksela w obrazie oraz porównanie tej wartości do zera.

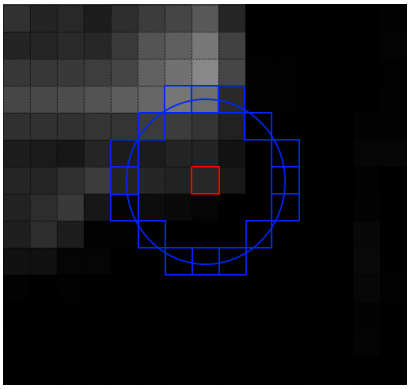
$$S(x,y)=\sum_u \sum_v w(u,v),\left(I(u+x,v+y)-I(u,v)\right)^2$$
$$I(u+x,v+y)\approx I(u,v)+I_x(u,v)x+I_y(u,v)y \qquad (8.1)$$

Równania te po podstawieniu można zapisać w postaci macierzowej:

$$A=\sum_u \sum_v w(u,v)\begin{bmatrix} I_x(u,v)^2 & I_x(u,v)I_y(u,v) \\ I_x(u,v)I_y(u,v) & I_y(u,v)^2 \end{bmatrix}$$

Forma macierzowa pozwala na stwierdzenie, czy punkt jest narożnikiem na podstawie wartości własnych macierzy stworzonej przez pochodne na kierunkach obrazu, bez konieczności przeprowadzania sumowań.

Algorytm FAST wykorzystuje szybkie progowanie na kręgu wokół piksela rozpatrywanego jako narożnik. Algorytm zakłada, że wokół narożnika musi znajdywać się co najmniej *n* pikseli o jasności oddalonej od badanego piksela o zadany próg. Ponieważ badanych jest 16 pikseli, a zakłada się, że co najmniej 13 musi być jednorodnych do odrzucenia hipotezy o byciu narożnikiem wystarczy zbadać jedynie 4 piksele, co znacząco przyspiesza działanie algorytmu. Na rysunku 8.3 przedstawiono przykładową analizę dla piksela przedstawiającego końcówkę robota Mitsubishi.



Rys. 8.3 Przykładowy okrąg testowy dla algorytmu FAST

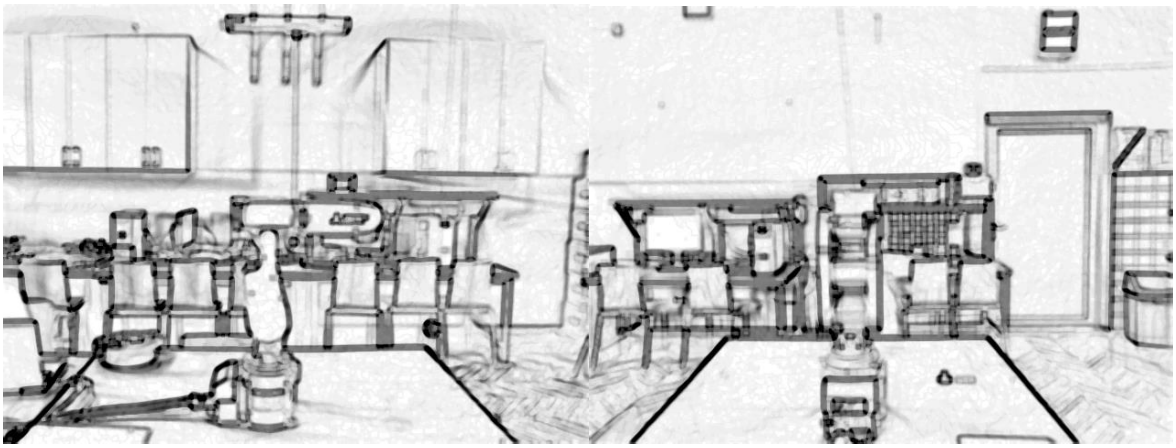
Detekcja krawędzi

Wykrywanie krawędzi obywa się przez znajdywanie nagłych zmian, nieciągłości, w strukturze obrazu. Na ogół są one spowodowane: różnicami obrazu w głębi sceny, nieciągłościami w strukturze materiału oraz zmianami naświetlenia pod wpływem kąta padania światła. W literaturze można znaleźć wiele dobrze opisanych metod służących wyznaczaniu krawędzi na obrazie. Wśród najbardziej popularnych można wskazać: operator Sobela, operator Scharra oraz algorytm Cannego [10]. W pracy użyto: operatora Scharra oraz algorytmu Cannego. Pierwszy użyto do detekcji poruszających się obiektów, drugi do tworzenia hierarchii elementów na obrazie.

Operator Scharra został użyty w dwóch wersjach, odpowiednio do wykrycia linii poziomych oraz pionowych:

$$S_1=\begin{bmatrix}-1 & 0 & +1\\ -2 & 0 & +2\\ -1 & 0 & +1\end{bmatrix}, \quad S_3=\begin{bmatrix}+1 & +2 & +1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

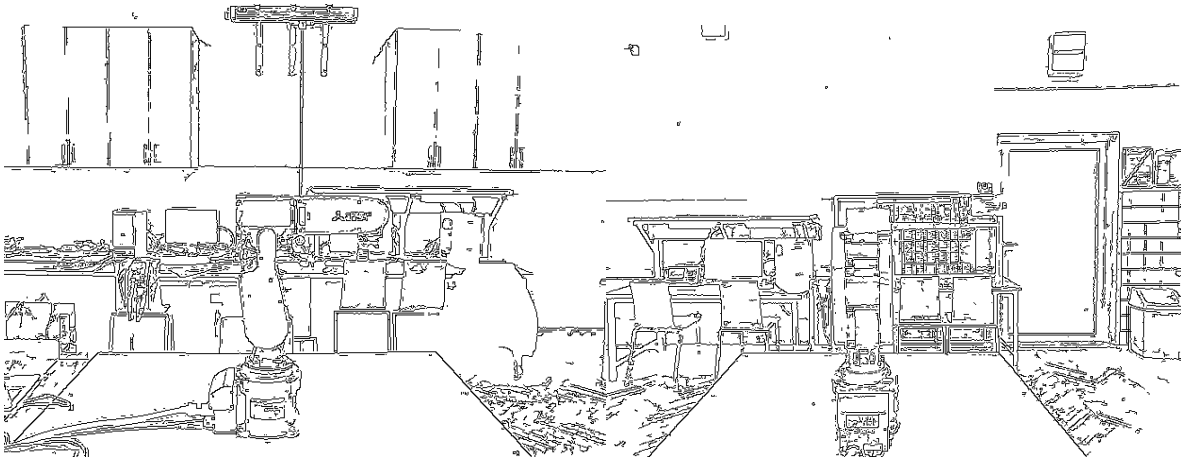
Przekształcenia są wykonywane równolegle, następnie wynikowe obrazy są sumowane z wagami 0,5. Wyniki działania algorytmu przedstawiono na rysunku 8.4.



Widok z prawej strony robota Widok zza robota

Rys. 8.4 Krawędzie uzyskane przy pomocy operatora Scharra

Rezultat działania algorytmu Cannego został zobrazowany na rysunkach 8.5.



Widok z prawej strony robota Widok zza robota

Rys. 8.5 Krawędzie uzyskane przy pomocy operatora Cannego

Można zaobserwować, że rezultatem algorytmu Cannego jest dużo "czystszy" obraz krawędzi. Jest to skutek dodatkowych kroków przetwarzania, które usuwają szum oraz redukują grubość krawędzi do minimum. Dodatkowe kroki przetwarzania powodują, że opisywany algorytm wykonuje się dłużej niż prosty, oparty na splocie operator Sobela.

Detekcja kleksów

Kleksem (ang. blob) nazywamy grupę położonych przy sobie punktów o wspólnej wyróżniającej w stosunku do otoczenia cesze. Cechą taką może być kolor lub nasycenie. Jednym z najlepszych narzędzi do wykrywania kleksów w obrazie jest operator LoG [8] (ang. Laplacian of Gaussian), który opisany jest wzorem:

$$LoG_{\sigma} = -\frac{1}{\pi\sigma^4}\left[1-\frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{(x^2+y^2)}{2\sigma^2}}.$$

Dla różnych wartości parametru σ maska użyta do splotu przyjmować będzie różne postaci. Przykładowa maska LoG dla $\sigma=1{,}4$:

$$LoG_{\sigma=1.4} = \begin{bmatrix} 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \\ 0 & 2 & 3 & 6 & 6 & 6 & 3 & 2 & 0 \\ 3 & 3 & 6 & 3 & 0 & 3 & 6 & 3 & 3 \\ 2 & 6 & 3 & -12 & -23 & -12 & 3 & 6 & 2 \\ 2 & 6 & 0 & -23 & -40 & -23 & 0 & 6 & 2 \\ 2 & 6 & 3 & -12 & -23 & -12 & 3 & 6 & 2 \\ 3 & 3 & 6 & 3 & 0 & 3 & 6 & 3 & 3 \\ 0 & 2 & 3 & 6 & 6 & 6 & 3 & 2 & 0 \\ 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \end{bmatrix}.$$

Po użyciu powyższej maski na obrazie uzyskane minima oraz maksima można interpretować jako kleks. Należy wyszukać maksima oraz minima odpowiedzi filtru. W miejscu maksimum

znajduje się ciemny kleks na jasnej powierzchni, a w minimum jasny kleks na ciemnej powierzchni.

Inną metodą, obecne niezwykle intensywnie badaną, jest metoda MSER (ang. Maximally stable extremal regions) opisana w [49]. Polega ona na wydzielaniu regionów poprzez wielokrotne progowanie względem jasności.

Algorytmy cechujące

Aby w chmurze uzyskanych punktów charakterystycznych wykryć punkty należące do poszukiwanego obiektu, należy przeprowadzić identyfikację cech i je opisać. Do najpopularniejszych metod opisu punktów charakterystycznych obiektów zaliczyć można: SIFT (ang. Scale-invariant feature transform), SURF (ang. Speded up robust features) czy HOG (ang. Histogram of Oriented Gradients).

Uproszczony algorytm detekcji robota

Kolejne etapy przetwarzania sceny przedstawiają się następująco:

1. Wykrycie stołu przy pomocy barw
2. Detekcja obiektów
    a. Wykrycie podstawy oraz członów robota
    b. Wykrycie elementów przesuniętych
3. Fuzja danych

Pierwszy etap nazwany “Wykryciem stołu przy pomocy barw” jest dokonywany przez ustalenie bezpośredniego odniesienia do zakresu przestrzeni roboczej. Detekcja stołu ma miejsce na podstawie koloru. W obszarze zdjęcia znajdywane są największe obszary zgodne z kolorem stołu, a następnie są one nanoszone na znalezione wcześniej kontury metodą Cannego tak, by został wyszukany kontur otaczający. Następnym krokiem jest wyznaczenie prostych, stycznych do znalezionego obszaru. Na rysunku 8.6 przedstawiono wynik wykrywania stołu dla dwóch kamer. Rysunek 8.7 przedstawia proste otaczające znalezioną przestrzeń roboczą. W celu uodpornienia algorytmu na zmianę oświetlenia, wszystkie operacje na barwach są przeprowadzane w przestrzeni HSV (ang. Hue, Saturation, Value).

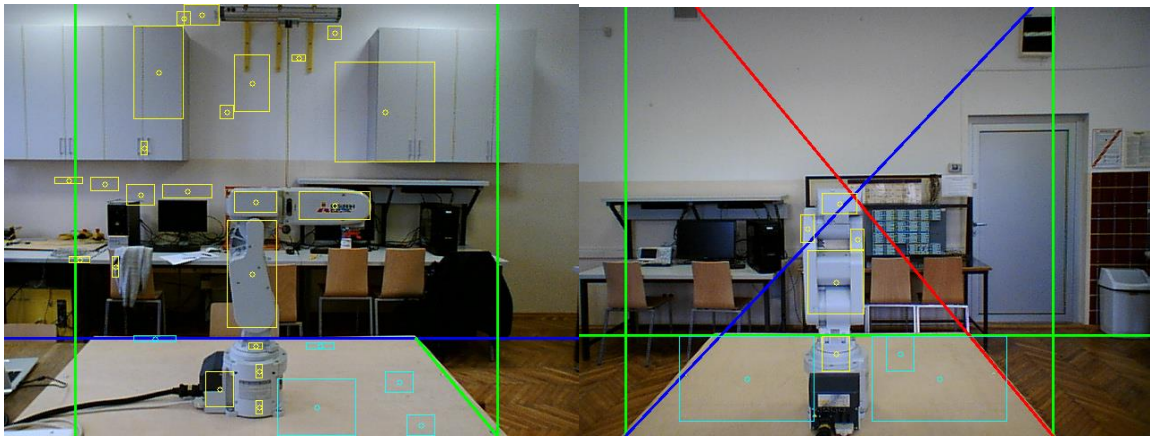
Drugi etap nazwany “Detekcją obiektów” składa się z dwóch możliwych do zrównoleglenia operacji. Pierwsza jest dokonywana na podstawie maski uzyskanej w pierwszym kroku. Przy pomocy prostych otaczających przestrzeń roboczą szacuje się wstępnie położenia ramienia robota, a co za tym idzie uzyskuje się bieżącą informację o kolorze i kształcie manipulatora.

Obiekty poruszające się są znajdywane przy pomocy różnicy aktualnej klatki obrazu z poprzednią. Używana jest niepełna informacja o kolorze, a jedynie informacja o krawędziach uzyskana przy pomocy operatora Scharra. Prowadzi to do zminimalizowania informacji o przesuwających się przedmiotach w kadrze.

Widok z prawej strony robota Widok zza robota

Rys. 8.6 Maski pokrywające przestrzeń stołu

Ostatnim etapem nazwanym “Fuzją danych” jest łącznie informacji z wielu kamer w jedną spójną trójwymiarową scenę. W tym celu znajdywane są na obrazach predefiniowane punkty charakterystyczne, które są następnie kojarzone i użyte jako odniesienia dla pozostałych obiektów.



Widok z prawej strony robota Widok zza robota

Rys. 8.7 Wykryte obszary na zdjęciach wraz z liniami odniesienia

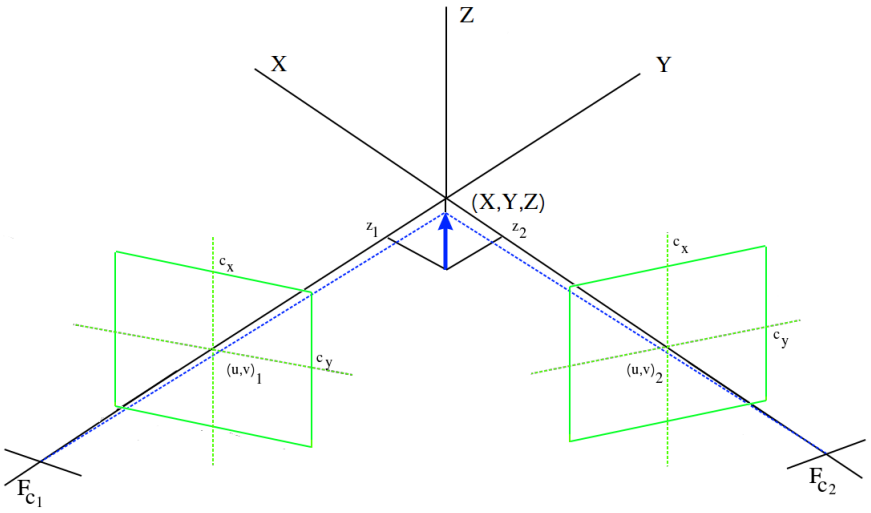
Odtworzenie położenia punktu w przestrzeni kartezjańskiej

W celu uzyskania estymowanego położenia obiektu w przestrzeni kartezjańskiej dokonuje się fuzji danych uzyskanych z co najmniej dwóch kamer. Na podstawie równań (8.2) wyznacza się prostą przechodzącą przez ognisko obiektywu oraz punkt będący projekcją obiektu na płaską matrycę kamery.

$$\begin{aligned} x' &= \frac{x}{y}, y' = \frac{y}{z} \\ u &= f_x x' + c_x \\ v &= f_y y' + c_y \end{aligned} \quad , \tag{8.2}$$

gdzie *x*, *y*, z są współrzędnymi punktu w układzie współrzędnych powiązanych z kamerą. Parametry f_x i f_y są długościami ogniskowej wyrażonymi w pikselach. Współrzędne (*u*, *v*) oznaczają pozycje punktu projekcji na matrycę, podobnie jak ogniskowe wyrażone w pikselach. Punkt o współrzędnych (c_x, c_y) oznacza arbitralnie ustalony punkt na obrazie, zwykle stanowiący początek układu współrzędnych dwuwymiarowych. Uproszczony schemat układu optycznego pokazano na rysunku 8.8.

Optyczne niedoskonałości obiektywu zostały w powyższych wzorach pominięte, ponieważ kamery zostały uprzednio skalibrowane. Odległość obiektu od obiektywu *z* nie może być ustalona na podstawie pojedynczego zdjęcia



Rys. 8.8 Uproszczony widok systemu optycznego

W celu estymacji położenia punktu w przestrzeni trójwymiarowej wyznacza się miejsca przecięcia co najmniej dwóch prostych przechodzących przez ogniskową kamery oraz poszukiwany punkt. W rzeczywistości otrzymanie dwóch prostych krzyżujących się ze sobą wyznaczonych na podstawie dwóch przesuniętych obrazów jest niezwykle trudne, a często niemożliwe. Rozwiązaniem łatwiejszym jest wyznaczenie na prostych punktów o najmniejszej odległości względem siebie. Te dwa punkty tworzą odcinek będący najkrótszą drogą między prostymi. Położenie poszukiwanego punktu w przestrzeni kartezjańskiej może być estymowane przez punkt środkowy tego odcinka.

$$\vec{f} = F_{C_1}\bar{F}_{C_2}$$

$$X_1 = F_{C_1} + \vec{a}_1 \frac{-(\vec{a}_1\cdot\vec{a}_2)(\vec{a}_2\cdot\vec{f}) + (\vec{a}_1\cdot\vec{f})(\vec{a}_2\cdot\vec{a}_2)}{(\vec{a}_1\cdot\vec{a}_1)(\vec{a}_2\cdot\vec{a}_2) - (\vec{a}_1\cdot\vec{a}_2)(\vec{a}_1\cdot\vec{a}_2)}$$

$$X_2 = F_{C_2} + \vec{a}_2 \frac{-(\vec{a}_1\cdot\vec{a}_2)(\vec{a}_1\cdot\vec{f}) + (\vec{a}_2\cdot\vec{f})(\vec{a}_1\cdot\vec{a}_1)}{(\vec{a}_1\cdot\vec{a}_1)(\vec{a}_2\cdot\vec{a}_2) - (\vec{a}_1\cdot\vec{a}_2)(\vec{a}_1\cdot\vec{a}_2)} \qquad (8.3)$$

$$X_{est} = \frac{X_1 + X_2}{2}$$

Podobną procedurę można zaaplikować podczas obliczania położenia punktu dla obrazu z trzech kamer. W taki wypadku zestaw równań (8.3) musi zostać użyte trzykrotnie, aby wyznaczyć odległości między każdą parą prostych. Uzyskane punkty w idealnym przypadku powinny się pokrywać. Jednak ze względu na wiele czynników zakłócających jest to skrajnie rzadki przypadek. Zwykle punkty stworzą trójkąt, wewnątrz którego znajduje się rzeczywisty punkt. Jego położenie można estymować poprzez wyznaczenie środka tego trójkąta.

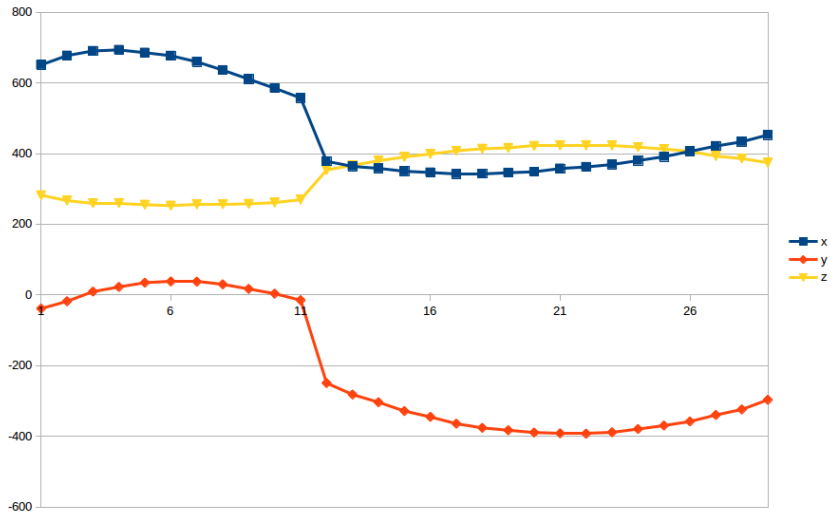
Użycie większej liczby kamer niż trzy jest możliwe, jednak znacząco wydłuża to czas obliczeń. Posługując się symbolem Newtona, by wyznaczyć ilość niezbędnych obliczeń, uzyskujemy wzór:

$$N=\frac{n(n-1)}{2}.$$

Pewną trudność implementacyjną przedstawia zagadnienie synchronizacji przetwarzania obrazów z dwóch niezależnie działających kamer. Aby móc uzyskać estymatę położenia punktu, niezbędne jest, by dane z kamer pochodziły z tego samego lub niezwykle zbliżonego momentu. W tym celu podczas implementacji użyto jednoelementowych buforów, które były wypełniane najświeższym obrazem z kamery, a następnie były opróżniane, gdy obraz został wykorzystany do obliczeń. Obliczenia były wyzwalane, gdy obrazy z obu kamer zostały odświeżone.

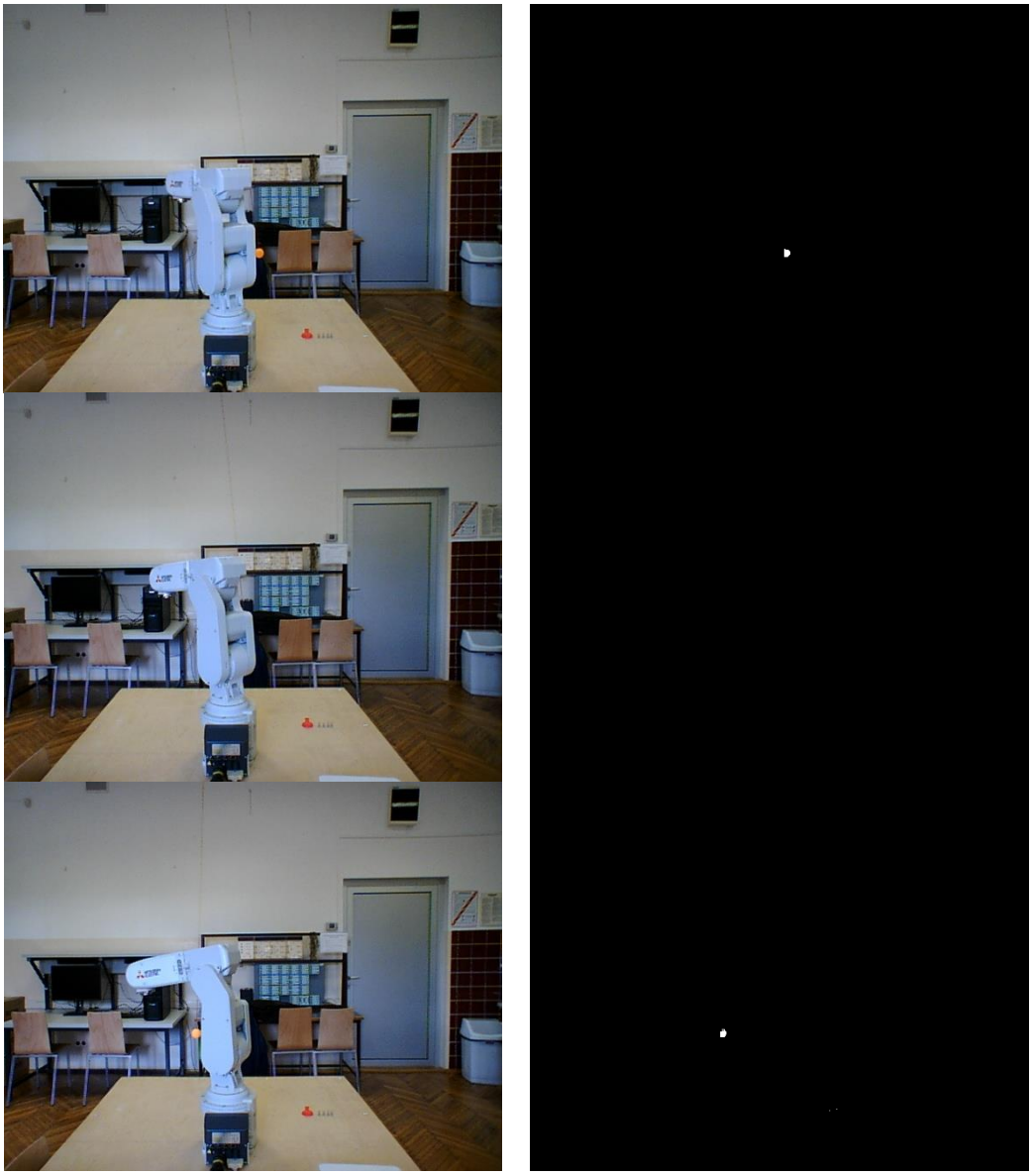
W wypadku użycia większej liczb kamer należałoby dodatkowo weryfikować, czy obrazy posiadają istotne dla przetwarzania detale. Jeśli obiekt śledzony zniknął z pola widzenia którejś kamery, nie oznacza to, że nie jest widoczny dla innych. Gdy użyte są tylko dwie kamery, problem ten nie występuje, ponieważ brak obiektu w którymś z obrazów powoduje automatycznie niemożność wyznaczenia położenia.

Odtworzone pozycje wykrytego obiektu pokazano na rysunku 8.9.



Rys. 8.9 Odtworzone położenie punktu w przestrzeni trójwymiarowej

Można na nim zaobserwować moment, w którym wykryta piłeczka była zakryta przez robota na jednym z obrazów kamer. Moment zakrycia obiektu został uchwycony na rysunkach 8.10. Przedstawiono na nim trzy fazy zakrycia obiektu. Pierwsza, gdy obiekt znika z prawej strony, druga, gdy obiekt w całości jest zakryty oraz trzecia, gdy obiekt znów staje się widoczny dla kamery. Po lewej stronie przedstawiono kolejne klatki z kamery monitorującej ruch, z prawej przedstawiono maskę, na której przedstawiono wykryty obiekt.

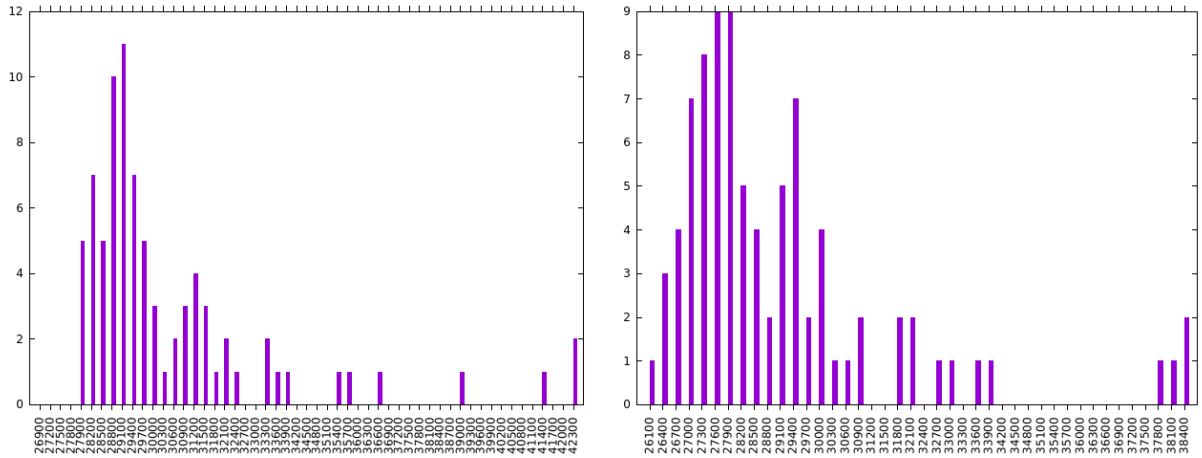


Rys. 8.10 Kolaż przedstawiający moment zakrywania przez robota wykrytego obiektu

Analiza czasowa

Istotnym elementem każdego algorytmu, który ma działać na rzeczywistym systemie, jest czas jego wykonania. Opisane wcześniej algorytmy zostały zaimplementowane przy użyciu języka C++ oraz były uruchomiane na komputerze klasy PC wyposażonym w Intel Core i7-6700HQ taktowany 2.6 GHz oraz 16 GB pamięci operacyjnej RAM. Systemem operacyjnym był Linux o jądrze 4.15.0-47. W sposób równoległy działy dwa watki, które przetwarzały symultanicznie

obrazy z dwóch kamer. Każde odpalenie algorytmu synchronizowane jest poprzez przerwanie od kamery wystawiane, gdy dostępna jest kolejna klatka do przetwarzania. Wykresy jittera tasku odpowiedzialnego za przetwarzanie obrazów przedstawiono na rysunku 8.11.



Rys. 8.11 Jitter procesu przetwarzania obrazów dla mikrokomputera klasy PC przy przetwarzaniu dwóch obrazów jednocześnie

Można zaobserwować, że niektóre próbki były przetwarzane ponad dwukrotnie dłużej niż inne. Wynikało to zwykle z wykrycia dodatkowych obiektów o kolorze lub kształcie zbliżonym do obiektu poszukiwanego, co wydłużało klasyfikację i w rezultacie wydłużało również czas działania algorytmu. Dodatkową obserwacją jest również to, że nie zdarzało się, by niektóre klatki były przetwarzane niewspółmiernie długo do pozostałych.

Podsumowanie

W pracy prześledzono konstrukcję algorytmów wysokiej dokładności śledzenia trajektorii robota przemysłowego. W opinii autora potwierdziły się tezy pracy, a jej cel ściśle związany z tytułem w dużym zakresie został osiągnięty.

Obszerny fragment pracy stanowi przedstawienie zagadnienia budowy algorytmów upraszczających zapis trajektorii do postaci wielomianowej. Dzięki zastosowaniu interpolacji wielomianowej możliwe było zredukowanie stopnia skomplikowania zapisu trajektorii. Pokazano, że nawet skomplikowane formy geometryczne, takie jak helisa, można interpolować bez szkody dla odwzorowania oryginalnego ruchu. Dodatkowo zauważono, że wystarczy użycie wielomianu trzeciego stopnia, gdy zastosowano interpolację klasycznymi wielomianami sklejanymi oraz drugiego stopnia w przypadku wielomianu B sklejanego.

Kolejnym istotnym punktem pracy było zaprezentowanie rozlicznych algorytmów generacji sterowania nadążnego. Przedstawiono m.in. algorytmy PD, PID, adaptacyjne, wspierane siecią neuronową oraz predykcyjne. Zastosowanie ich pozwala na minimalizację błędu, co z kolei przekłada się na wysoka dokładność odwzorowania zamierzonej trajektorii. Aby było to możliwe, w pracy umieszczono wprowadzenie do dynamiki oraz identyfikacji robotów. Przeprowadzono również serię eksperymentów identyfikacyjnych, które pozwoliły na poznanie modelu badanego manipulatora RV-2D. Przy pomocy uzyskanych równań było możliwe zsyntezowanie regulatora predykcyjnego opartego o funkcję kary. Jej specjalna forma wprowadza dodatkowo ograniczenia wynikające z przestrzeni operacyjnej samego robota.

Dodatkowo, w celu zobrazowania elastyczności algorytmów predykcyjnych o działaniu opartym na minimalizacji funkcji kary, do pracy wprowadzono elementy przetwarzania obrazów. Przy użyciu autorskiego systemu wizyjnego możliwe było śledzenie obiektu spoza układu robota. Jego położenie było monitorowane przez kamery, a następnie zostało odtworzone w układzie kartezjańskim związanym z manipulatorem. Obiekt wprowadzono następnie do funkcji kary jako dodatkowe zakłócenie w celu pokazania, że algorytm predykcyjny jest w stanie uniknąć kolizji, pomimo nieznanego początkowego położenia przeszkody. Opis eksperymentu można znaleźć w pracy [65] oraz rozdziale 7.

Oryginalne wyniki autora zawarto w rozdziałach 3, 5, 8 i 7. By zachować kolejność historyczną powstawania tych wyników, przestawiono rozdziały ósmy z siódmym. Ukoronowaniem badań jest praca [76] prezentowana przez autora podczas warsztatów “Control Applications of Optimization” (CAO) w Jekaterinburgu w Rosji w dniach 15-19 października 2018 roku. Dotyczy ona aplikacji nieliniowego regulatora predykcyjnego wykorzystującego funkcję kary, tak pomyślaną, by robot był zdolny do omijania przeszkód nieruchomych, a także przeszkód będących w ruchu.

Spis Rysunków

Bibliografia

(1) Ahlberg J.H., Nilson E.N., and Walsh J.L., 1967, *The Theory of Splines and their Applications*. Academic Press

(2) Akishita S., Kawamura S., 1990, and Hayashi K. *Laplace potential for moving obstacle avoidance and approach of a mobile robot*. Japan-USA Symposium on Flexible Automation, A Pacific Rim Conference, pp.139–142

(3) Arkin R. C., 2005, *Principles of Robot Motion Theory, Algorithms and Implementation*. MIT Press, Boston

(4) Bania P., 2008, *Algorytmy sterowania optymalnego w nieliniowej regulacji predykcyjnej*, PhD thesis, Kraków

(5) Bania P., 2006, *Czy sterowanie predykcyjne wymaga dokładnej optymalizacji?*, Automatyka, vol. 10

(6) Behal A., Dixon W., Dawson D. M., and Xian B., 2010, *Lyapunov-Based Control of Robotic Systems*. CRC Press

(7) Bin Yao, Masayoshi Tomizuka, 1996, *Smooth robust adaptive sliding mode control of manipulators with guaranteed transient performance*.

(8) Burt P., Adelson E.,1983, *The Laplacian Pyramid as a Compact Image Code*, IEEE Transactions on Communications, vol. 31(4) , pp. 532-540

(9) Campion G., Bastin G., and D'Andrea-Novel B., 1996, Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. IEEE Transactions on Robotics and Automation, 12(1), pp. 47– 62

(10) Canny J., 1986, *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8, pp. 679-714.

(11) Chuan-Chih Hsiung, 1981, *A first course in Differential Geometry*. John Wiley & Sons, 1981.

(12) Connolly C. I., Burns J. B., and Weiss R., 1990, *Path planning using Laplace's equation*. In Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2102–2106

(13) Craig J., 1995, *Wprowadzenie do robotyki*, WNT, Warszawa

(14) Crampin M., Guifo R., and Read G.A., 1985, Linear approximation of curves with bounded curvature and a data reduction algorithm. Computer Aided Design, vol. 17(6), pp.257-261

(15) Cybenko G., 1989, *Approximation by Superposition of Sigmoidal Functions*, Mathematical Control SignalsSystems, Vol. 2. pp. 303-314

(16) Danevit, Hatenberg, 1955, *A kinematic notation of lower-pair mechanisms based on matrices*, Journal of Applied Mechanics

(17) Dulęba I., 1998, *Algorithms of motion planning for nonholonomic robots*. Publishing House of Wrocław University of Technology

(18) Dutkiewicz P., Kozłowski K. R., Experimental identification of robot and load dynamic parameters, Control Applications, Second IEEE Conference on, (1993)

(19) Farin G., 1996, *Curves and Surfaces for CAGD*. Academic Press

(20) Farin G., 1999, *NURB Curves and Surfaces*. A. K. Peters

(21) Faux I.D. and Pratt M.J., 1979, *Computational Geometry for Design and Manufacture*. Ellis Horwood
(22) Feder H. J. S. and Slotine J. J. E., 1997, *Real-time path planning using harminic potentials in dynamic environments*. In Proceedings of the IEEE International Conference on Robotics and Automation, pp. 874– 881
(23) Fox, D., Burgard, W., and Thrun S., 1997, *The dynamic window approach to collision avoidance,* IEEE Robotics & Automation Magazine, vol. 4.1, p. 23-33.
(24) Funahashi K. I., 1989, *On the Approximate Realization of Continuous Mappings by Neural Networks*, Neural Networks, Vol. 2. No. 3. pp. 183-192.
(25) Gilbert, E., and Johnson, D., 1985, *Distance functions and their application to robot path planning in the presence of obstacles.* IEEE Journal on Robotics and Automation, vol. 1.1, pp. 21-30.
(26) Girau B., and Boumaza A., 2007, Embedded harmonic control for dynamic trajectory planning on fpga. In Proceedings of the 25th conference on Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications, pp. 244–249, Anaheim, CA, USA
(27) Graetz G. and Michaels G., 2018, Robots at Work. IZA Discussion Paper No. 8938. Available at SSRN: https://ssrn.com/abstract=2589780
(28) Gregory J. A. and Delbourgo R., 1983, *C2 rational quadratic spline interpolation for monotonic data*. IMA Journal of Numerical Analysis, vol. 3, pp. 141-152
(29) Gregory J. A. and Delbourgo R., 1982, *Piecewise rational quadratic interpolation for monotonic data*. IMA Journal of Numerical Analysis, pp. 123-130
(30) Gronowicz A., 2003, *Podstawy analizy układów kinematycznych*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław
(31) Holzle G.E., 1983, *Knot placement for piecewise polynomial approximation of curves*. Computer Aided Design, vol. 15(5), pp.295-296
(32) Hornik K., Stinchcombe M. and White H., 1989, *Multilayer Feed-forward Networks are Universal Approximators*, Neural Networks Vol. 2. pp. 359-366.
(33) KaewTraKulPong P., Bowden R., 2002, *An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection*. In: Remagnino P., Jones G.A., Paragios N., Regazzoni C.S. (eds) Video-Based Surveillance Systems
(34) Khalil, Kleinfinger, 1986, A new geometric notation for open and closed-loop robots, IEEE
(35) Khatib O., 1986. *Real-time obstacle avoidance for manipulators and mobile robots*. In Autonomous robot vehicles. pp. 396-404. Springer, New York, NY.
(36) Kim J. and Khosla P., 1992, *Real-time obstacle avoidance using harmonic potential functions*. IEEE Transactions on Robotics and Automation, pp. 338–349
(37) Koditschek D., 1987, *Exact robot navigation by means of potential functions: Some topological considerations*. In Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1–6.
(38) Koker R., 2005, *Design and performance of an intelligent predictive controller for a six-degree-of-freedom robot using the Elman network*
(39) Kozlowski K. R. 2012 *Modelling and Identification in Robotics*, Advances in Industrial Control

(40) Kryjak T., 2011, *Analysis and evaluation of background subtraction algorithms for video survelliance systems*, Automatyka, vol. 15(3), pp. 177–196
(41) LaValle S. M., 2006, *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., Available at http://planning.cs.uiuc.edu/.
(42) Leshno M., Lin V. Y., Pinkus A. and Schocken S., 1993, *Multilayer Feed-forward Networks With a Nonpolynomial Activation Function Can Approximate Any Function*, Neural Networks, Vol. 6. pp. 861-67
(43) Lewis F. L., Dawson D.M., Abdallah Ch.T., 2003, *Robot Manipulator Control, Theory and Practice*. CRC Press
(44) Louste C. and Liégeois A., 2002, *Path planning for nonholonomic vehicles: a potential viscous fluid field method*. Robotica, vol. 20, pp. 291–298
(45) Ma W. and Kruth J.P., 1994, *Mathematical modelling of free-form curves and surfaces from discrete points with nurbs*. In P. J. Laurent, A. Le Mehaute, and L. Schumaker, editors, Curves and Surfaces II. A. K. Peters
(46) Mao, Z., and Hsia, T. C., 1997, *Obstacle avoidance inverse kinematics solution of redundant robots by neural networks*. Robotica, 15(1), pp. 3-10.
(47) Marchewka D., 2006, Algorytmy sterowania robotem za pomocą silników niskoobrotowych o dużym momencie obrotowym, Rozprawa doktorska, Wydział Elektrotechniki, Automatyki Informatyki i Elektroniki, AGH
(48) Mastellone S., Stipanovic D. M., Graunke C. R., Intlekofer K. A., and Spong M. W., 2008, *Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments*. The International Journal of Robotics Research, vol. 27(1), pp. 107–126
(49) Matas J., Chum O., Urban M., and Pajdla T., 2002, *Robust wide baseline stereo from maximally stable extremal regions*. Proc. of British Machine Vision Conference, pp. 384-396
(50) Milne-Thomson L.M., 1996, *Theoretical hydrodynamics*. Dover Publications.
(51) Morecki A., Knapczyk J., 1994, *Podstawy robotyki: teoria i elementy manipulatorów i robotów*, WNT, W-wa
(52) Nubiola, A., and Bonev I., 2014, *Geometric approach to solving the inverse displacement problem of calibrated decoupled 6R serial robots*. Transaction of the Canadian Society for Mechanical Engineering, vol. 38.1, p. 31-44.
(53) Panagou D., Tanner H. G., and Kyriakopoulos K. J., 2010, *Dipole-like fields for stabilization of systems with pfaffian constraints*. In Proceedings of the IEEE International Conference on Robotics and Automation, pp. 4499–4504, Anchorage, Alaska, USA
(54) Piazzi A., Lo Bianco C. G., 2001, *A semi-infinite optimization approach to optimal spline trajectory planning of mechanical maniplator*, Semi-Infinite Programming, pp. 271-297
(55) Piazzi A., Visioli A., 2000, *Global Minimum-Jerk Trajectory Planning of Robot Manipulators*, IEEE Transactions on Industrial Electronics, vol. 47, no. 1
(56) Piłat A., 2014, *Embedding a PD controller into an Analogue Signal Processor using a bilinear filter analogue module*, Aktualne problemy automatyki i robotyki: praca

zbiorowa pod red. Krzysztofa Malinowskiego, Jerzego Józefczyka, Jerzego Świątka. pp. 477–486
(57) Powell M. J. D., 1964, *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, Computer Journal. 7, pp. 155–162.
(58) Pozniak S., Sanchez E. N., Yu W., Camastra F., 2002, *Differential Neural Networks for Robust Nonlinear Control*, IEEE Transactions on Neural Networks
(59) Pratt M., Goult R., and He L., 1993, *On rational parametric curve approximation*. Computer Aided Geometric Design, vol. 10(3/4), pp. 363-377
(60) Rahman M. H., Saad M., Kenné J. P., Archambault P. S., 2009, *Modeling and Control of a 7DOF Exoskeleton Robot for Arm Movements*, Proceedings of the 2009 IEEE, International Conference on Robotics and Biomimetics
(61) Reyes F., Kelly R., 1997, *Experimental evaluation of identification schemes on a direct drive robot*, Robotica vol. 15, pp. 563-571
(62) Rossiter J. A., 2003, *Model based predictive control : a practical approach*, CRC Press
(63) Roussos G. P., Dimarogonas D. V., and Kyriakopoulos K. J., 2008, *3d navigation and collision avoidance for a non-holonomic vehicle*. In Proceedings of American Control Conference, pp. 3512–3517, Seattle, WA, USA
(64) Saramago, S. F., and Junior V. S., 2000, *Optimal trajectory planning of robot manipulators in the presence of moving obstacles*. Mechanism and Machine Theory, vol. 35(8) pp. 1079-1094
(65) Schneider F. J. Interpolation, 1993, *Approximation and Konverterung mit rationalen B-Splines*. PhD thesis, TH Darmstadt, Germany
(66) Sheth, Uicker, 1971, *A generalized symbolic notation for mechanisms*, Journal of Engineering for Industry, vol. 93(1)
(67) Spong M. W., Hutchinson S., Vidyasagar M., 2006, *Robots modeling and control*
(68) Szulczyński P., Pazderski D., Kozłowski K., 2011, *Real-time obstacle avoidance using harmonic potential functions*, Journal of Automation Mobile Robotics and Intelligent Systems, Vol. 5, No. 3, pp. 59-66
(69) Tadeusiewicz R., 1993, *Sieci neuronowe*, Akademicka Oficyna Wydawnicza, W-wa
(70) Takefuji Y., 1992, *Neutral network parallel computing,* Kluwer Academic Publishers Norwell, MA, USA
(71) Tchon K., Muszynski R., 1998, *Singular Inverse Kinematic Problem for Robotic Manipulators: A Normal Form Approach,* IEEE Trans. Robotics & Automat., Vol. 14, No.1, pp. 93-104.
(72) Tchon K., Muszynski R., 1997, *Singularities of Nonredundant Robot Kinematics,* Int. J. Robotic Res., vol. 16, pp. 60-76.
(73) Tchon K., 1998, *Quadratic normal forms of redundant robot kinematics with application to singularity avoidance*. IEEE Trans. Robotics & Automat., vol. 14, No.5, pp. 834-837.
(74) Tchon K., 1997, *Singularity avoidance in redundant robot kinematics: a dynamical system approach*. Appl. Math. and Comp. Sci., vol. 7, no.2, pp. 401-412.

(75) Turnau A., 2002, *Target control of nonlinear systems in real time – intelligent and time-optimal algorithms*, Uczelniane Wydawnictwa Naukowo-Dydaktyczne, Rozprawy Monografie, vol. 102

(76) Turnau, A., and Zwonarz W., 2017, *Real-time control of the Mitsubishi RV-2F robot*. In Kosiuczenko et al (eds.), Software engineering research for the practice, p. 123-131. Polish Information Processing Society, Warszawa.

(77) Ude A., Atkeson C. G., Riley M., 2000, *Planning of Joint Trajectories for Humanoid Robots Using B-Spline Wavelets*

(78) Waydo S. and Murray R. M., 2003, *Vehicle motion planning using stream functions*. In Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2484–2491

(79) Wolters H., 1994, *Rational Geometric Curve Approximation*. PhD thesis, Arizona State University

(80) Zwonarz W., 2013, *Time stability of computer generated control in real-time*. Automatyka vol. 17 no. 2, pp. 271–277.

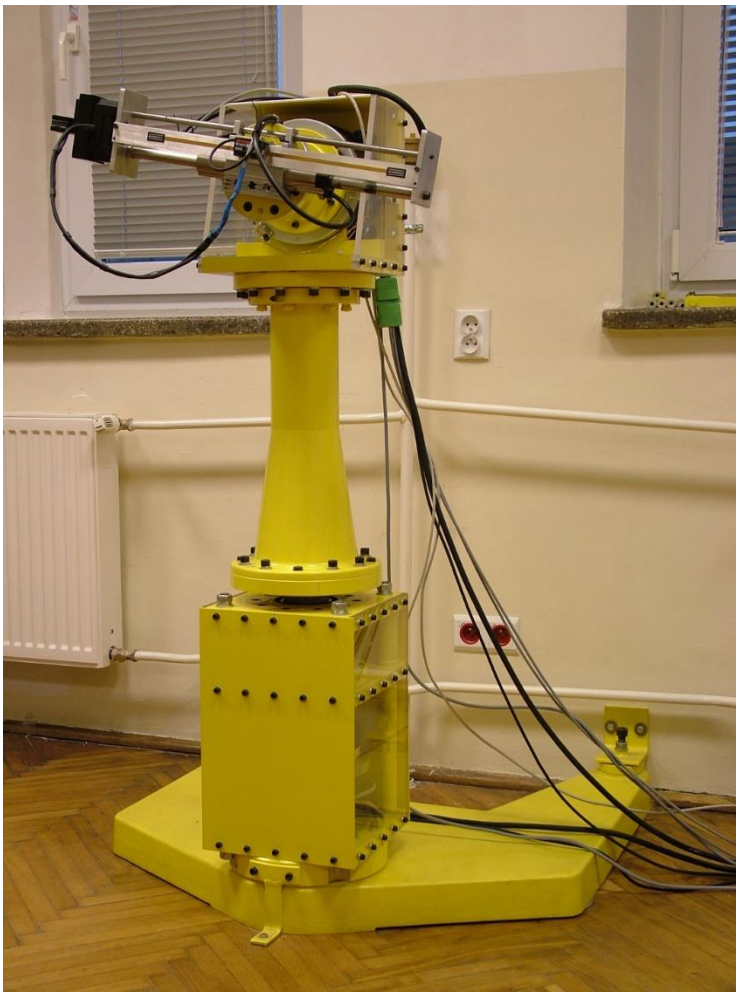
(81) Zwonarz W., 2013, *Trajectory planning and path tracking in real time*. In Trybus L., and Mastalerz M. W. (eds.), Design, development and implementation of real-time systems pp. 191-196

(82) Zwonarz W., 2014, *Predictive control based on robotic arm model*. In 19th International Conference On Methods and Models in Automation and Robotics (MMAR), pp. 544-548

(83) Zwonarz W., Turnau A., 2018, *Dynamically modified penalty function for control of Mitsubishi robotic arm in real time.* IFAC-PapersOnLine vol. 51 iss. 32, pp. 418-432

Dodatek A Kinematyki robotów

1. Robot stanfordzki z niskoobrotowymi o wysokim momencie silnikami dedykowanymi robotom firmy japońskiej NSK



Rys.A.1 Robot laboratoryjny ramię Stanfordzkie

Kinematyka prosta

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -0{,}2 \cdot \sin\varphi - \Theta \cdot \sin\alpha \cdot \cos\varphi \\ 0{,}2 \cdot \cos\varphi - \Theta \cdot \sin\alpha \cdot \sin\varphi \\ 0{,}9 - \Theta \cdot \cos\alpha \end{bmatrix}$$

Kinematyka odwrotna

$$\begin{bmatrix} \varphi \\ \alpha \\ \Theta \end{bmatrix} = \begin{bmatrix} atan2(y,x) - \frac{\pi}{2} - acos\left(\frac{0{,}2}{\sqrt{x^2+y^2}}\right) \\ atan2(r,-(z-0{,}9)) \\ \sqrt{(z-0{,}9)^2+r^2} \end{bmatrix}$$

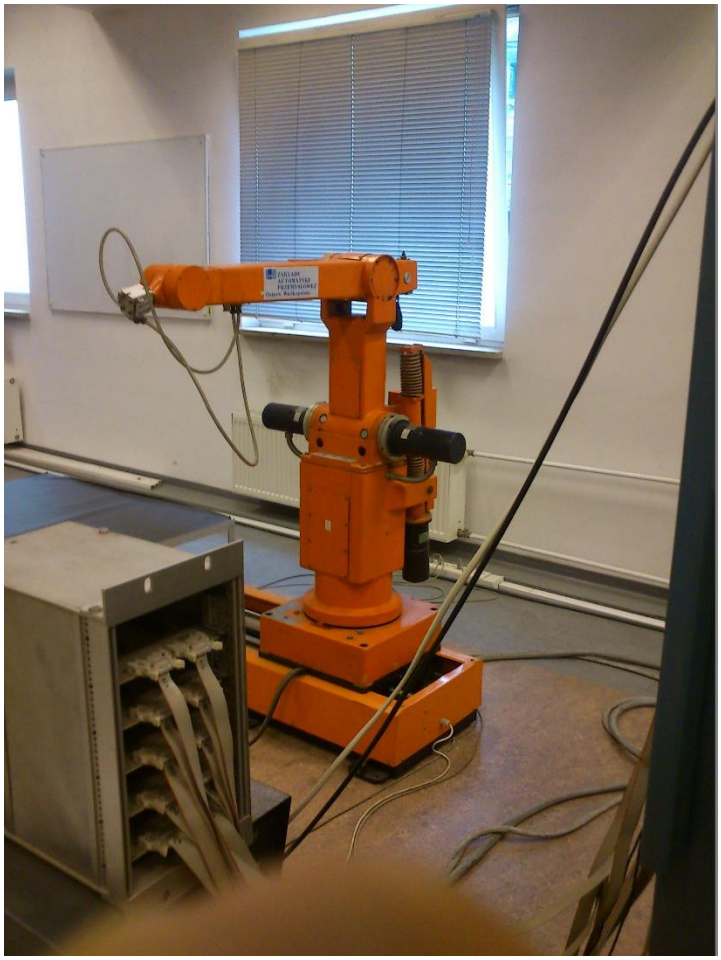
$$r=\sqrt{x^2+y^2-0{,}2^2}$$

Jakobian

$$J_v=\begin{bmatrix} -0{,}2\cos\varphi+\Theta\sin\varphi\sin\alpha & -\Theta\cos\varphi\cos\alpha & -\cos\varphi\sin\alpha \\ -0{,}2\sin\varphi+\Theta\cos\varphi\sin\alpha & -\Theta\sin\varphi\cos\alpha & -\sin\varphi\sin\alpha \\ 0 & \Theta\sin\alpha & -\cos\alpha \end{bmatrix}$$

$$J_\omega=\begin{bmatrix} 0 & -\sin\varphi & 0 \\ 0 & \cos\varphi & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

2. Robot przemysłowy IRp-6 z silnikami prądu stałego z pomiarem kątów obrotu ramion przy użyciu resolverów i pomiarem tachometrycznym ich prędkości



Rys.A.2 Robot przemysłowy IRp-6

Kinematyka prosta

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos\varphi\,(0{,}67\cos(\alpha+\Theta)+0{,}45\sin\Theta) \\ \sin\varphi\,(0{,}67\cos(\alpha+\Theta)+0{,}45\sin\Theta) \\ 0{,}45\cos\Theta+0{,}67\sin(\alpha+\Theta)+0{,}7 \end{bmatrix}$$

$$\varphi \in \left\langle -\frac{8}{9}\Pi, \frac{8}{9}\Pi \right\rangle$$

$$\Theta \in \left\langle -\frac{2}{9}\Pi, \frac{2}{9}\Pi \right\rangle$$

$$\varphi \in \left\langle -\frac{25}{180}\Pi, \frac{2}{9}\Pi \right\rangle$$

$$-\frac{2}{9}\Pi \le \Theta - \alpha \le \frac{2}{9}\Pi$$

Kinematyka odwrotna

$$\begin{bmatrix} \varphi \\ \Theta \\ \alpha \end{bmatrix} = \begin{bmatrix} atan2(y,x) \\ \frac{\Pi}{2} - atan\left(\frac{z-0{,}7}{\sqrt{x^2+y^2}}\right) - acos\left(\frac{R^2+0{,}45^2-0{,}67^2}{2\cdot 0{,}45\cdot R}\right) \\ asin\left(\frac{0{,}45\cos\Theta - z + 0{,}7}{0{,}67}\right) \end{bmatrix}$$

$$r = \sqrt{x^2+y^2}$$

$$R = \sqrt{x^2+y^2+(z-0{,}7)^2}$$

Jakobian

$$J_v = \begin{bmatrix} -A\sin\varphi & B\cos\varphi & -0{,}67\cos\varphi\cos\alpha \\ A\cos\varphi & B\sin\varphi & -0{,}67\sin\varphi\cos\alpha \\ 0 & -A & -0{,}67\cos\alpha \end{bmatrix}$$

$$A = 0{,}67\cos\alpha + 0{,}45\sin\varphi$$

$$B = 0{,}45\cos\Theta - 0{,}67\cos\alpha$$

$$J_\omega = \begin{bmatrix} 0 & -\sin\varphi & -\sin\varphi \\ 0 & \cos\varphi & \cos\varphi \\ 1 & 0 & 0 \end{bmatrix}$$

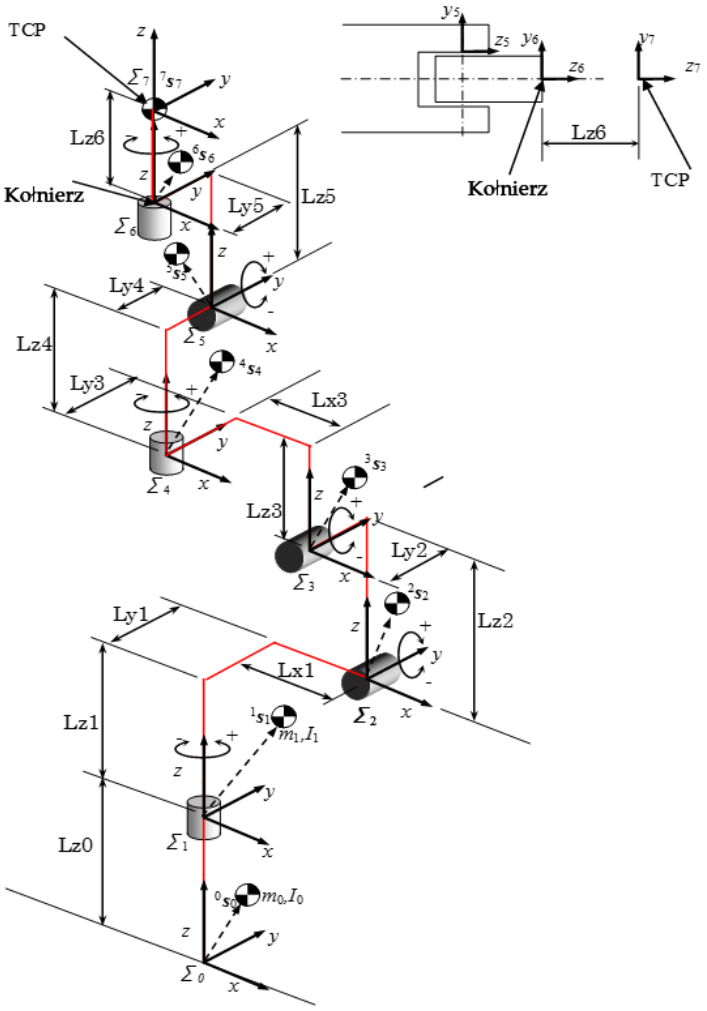
3. Robot przemysłowy Mitsubishi RV-2F używany na ogół do **operacji przenoszenia, pozycjonowania i montażu.**

Rys. A.3 Robot przemysłowy Mitsubishi RV-2F

Kinematyka prosta

	O_1	O_2	O_3	O_4	O_5	O_6	O_7
$L_{x\ i-1}$ [mm]	0	0	0	-50	0	0	100
$L_{y\ i-1}$ [mm]	0	34	0	-34	34	-34	0
$L_{z\ i-1}$ [mm]	170,5	124,5	230	66	204	70	100

Tab. A.1 Parametry kinematyki prostej robota przemysłowego Mitsubishi RV-2F



Rys. A.4 Kinematyka robota Mitsubishi RV-2F

$$A_6^0=\begin{bmatrix} a & b & c & p \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$a=\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix},\ b=\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix},\ c=\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix},\ p=\begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

$$\begin{aligned} a_1 &= -\sin(q_1)(\cos(q_4)\sin(q_6)+\sin(q_4)\cos(q_5)\cos(q_6)) \\ &\quad -\cos(q_2+q_3)\cos(q_1)(\sin(q_4)\sin(q_6)-\cos(q_4)\cos(q_5)\cos(q_6)) \\ &\quad -\sin(q_2+q_3)\cos(q_1)\cos(q_6)\sin(q_5) \\ a_2 &= \cos(q_1)(\cos(q_4)\sin(q_6)+\sin(q_4)\cos(q_5)\cos(q_6)) \\ &\quad -\cos(q_2+q_3)\sin(q_1)(\sin(q_4)\sin(q_6)-\cos(q_4)\cos(q_5)\cos(q_6)) \\ &\quad -\sin(q_2+q_3)\sin(q_1)\cos(q_6)\sin(q_5) \\ a_3 &= \sin(q_2+q_3)(\sin(q_4)\sin(q_6)-\cos(q_4)\cos(q_5)\cos(q_6))-\cos(q_2+q_3)\cos(q_6)\sin(q_5) \end{aligned}$$

$$
\begin{aligned}
b_1 &= \sin(q_1)(\cos(q_4)\cos(q_6)+\sin(q_4)\cos(q_5)\sin(q_6)) \\
&\quad +\cos(q_2+q_3)\cos(q_1)(\sin(q_4)\cos(q_6)+\cos(q_4)\cos(q_5)\sin(q_6)) \\
&\quad -\sin(q_2+q_3)\cos(q_1)\sin(q_6)\sin(q_5) \\
b_2 &= -\cos(q_1)(\cos(q_4)\cos(q_6)-\sin(q_4)\cos(q_5)\sin(q_6)) \\
&\quad +\cos(q_2+q_3)\sin(q_1)(\sin(q_4)\cos(q_6)+\cos(q_4)\cos(q_5)\sin(q_6)) \\
&\quad -\sin(q_2+q_3)\sin(q_1)\sin(q_6)\sin(q_5) \\
b_3 &= -\sin(q_2+q_3)(\sin(q_4)\cos(q_6)-\cos(q_4)\cos(q_5)\sin(q_6))-\cos(q_2+q_3)\sin(q_6)\sin(q_5)
\end{aligned}
$$

$$
\begin{aligned}
c_1 &= \sin(q_1)\sin(q_4)\sin(q_5)-\sin(q_2+q_3)\cos(q_1)\cos(q_5)-\cos(q_2+q_3)\cos(q_1)\cos(q_4)\sin(q_5) \\
c_2 &= -\cos(q_1)\sin(q_4)\sin(q_5)-\sin(q_2+q_3)\sin(q_1)\cos(q_5)-\cos(q_2+q_3)\sin(q_1)\cos(q_4)\sin(q_5) \\
c_3 &= \sin(q_2+q_3)\cos(q_4)\sin(q_5)-\cos(q_2+q_3)\cos(q_5)
\end{aligned}
$$

$$
\begin{aligned}
p_1 &= \cos(q_1)(a_1+a_3\cos(q_2+q_3)+a_2\cos(q_2)) \\
&\quad +\sin(q_2+q_3)\cos(q_1)(d_4+d_6\cos(q_5))-d_6\sin(q_1)\sin(q_4)\sin(q_5) \\
&\quad +d_6\cos(q_2+q_3)\cos(q_1)\cos(q_4)\sin(q_5) \\
p_2 &= \sin(q_1)(a_1+a_3\cos(q_2+q_3)+a_2\cos(q_2)) \\
&\quad +\sin(q_2+q_3)\sin(q_1)(d_4+d_6\cos(q_5))+d_6\cos(q_1)\sin(q_4)\sin(q_5) \\
&\quad +d_6\cos(q_2+q_3)\sin(q_1)\cos(q_4)\sin(q_5) \\
p_3 &= d_1+\cos(q_2+q_3)(d_4+d_6\cos(q_5))-a_3\sin(q_2+q_3)-a_2\sin(q_2)-d_6\sin(q_2+q_3)\cos(q_4)\sin(q_5)
\end{aligned}
$$

Elementy dynamiki

W tabelach A.2, A.3 oraz A.4 umieszczono wybrane parametry manipulatora udostępnione przez producenta. Parametry pochodzą z modelu mechaniki stworzonego w oprogramowaniu CAD/CAM.

	m0	m1	m2	m3	m4	m5	m6
kg	7.6	3.54	4.71	1.75	2.96	0.7	0.2

Tab. A.2 Masy członów robota przemysłowego Mitsubishi RV-2F

	s0	s1	s2	s3	s4	s5	s6
X [mm]	-3.98	1.07	-13.24	-40.85	0.42	-0.41	0.00
y [mm]	-0.14	3.42	-7.65	-45.36	-1.95	-27.34	0.00
z [mm]	143.48	34.90	105.64	-1.58	87.12	21.37	3.59

Tab. A.3 Przesunięcia środka masy względem początków członów w robocie przemysłowym Mitsubishi RV-2F

	I0			I1			I2		
xx,xy,xz gmm^2	0.00E+00	0.00E+00	0.00E+00	2.33E+07	-1.42E+04	-9.49E+04	3.63E+07	-6.06E+05	8.03E+05
yx,yy,yz gmm^2	0.00E+00	0.00E+00	0.00E+00	-1.42E+04	2.33E+07	-1.91E+06	-6.06E+05	3.05E+07	-4.23E+05
zx,zy,zz gmm^2	0.00E+00	0.00E+00	0.00E+00	-9.49E+04	-1.91E+06	4.94E+06	8.03E+05	-4.23E+05	1.27E+07
	I3			I4			I5		
xx,xy,xz gmm^2	5.60E+06	-1.96E+05	-6.43E+04	2.02E+07	1.20E+04	-2.42E+05	8.09E+05	-2.25E+03	-4.50E+03
yx,yy,yz gmm^2	-1.96E+05	5.98E+06	-5.66E+05	1.20E+04	2.02E+07	2.28E+05	-2.25E+03	7.73E+05	9.09E+04
zx,zy,zz gmm^2	-6.43E+04	-5.66E+05	4.70E+06	-2.42E+05	2.28E+05	4.19E+06	-4.50E+03	9.09E+04	5.12E+05
	I6			I7					
xx,xy,xz gmm^2	3.13E+04	3.34E+00	-1.63E+02	3.20E+07	0.00E+00	0.00E+00			
yx,yy,yz gmm^2	3.34E+00	3.10E+04	0.00E+00	0.00E+00	3.20E+07	0.00E+00			
zx,zy,zz gmm^2	-1.63E+02	0.00E+00	4.56E+04	0.00E+00	0.00E+00	7.00E+06			

Tab. A.4 Macierze bezwładności poszczególnych ramion robota Mitsubishi RV-2F

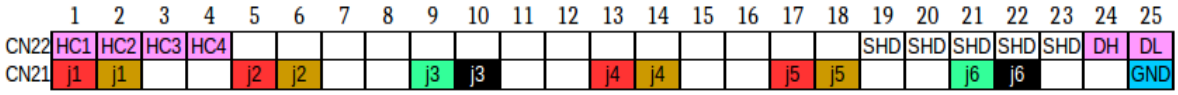
Dodatek B Enkodery w robocie Mitsubishi RV-2F-D

Warstwa fizyczna

Komunikacja z enkoderami dla sterownika CR750 odbywa się przez 6 równoległych szyn komunikacyjnych w standardzie RS485 (jedna szyna na enkoder) z następującą konfiguracją: 2 500 000 b/s, 8/N/1.

Konwersja między sygnałem różnicowym a asymetrycznym może być przeprowadzona przy pomocy czipu SN75176BM chip.

Fizyczne wyjścia umieszczone są na dolnej stronie wtyczki CN2 (wewnętrzna taśma danych CN21), rysunek B.1.



Rys. B.1 Wtyczka CN2 wraz z oznaczeniami różnicowych szynk danych j1...6, kolory kratek pasują do kolorów okablowania wewnątrz robota.

Komunikacja

Komunikacja odbywa się w systemie Master-Slave, gdzie sterownik przyjmuje rolę mastera. Transmisja danych jest zorganizowana w następujący sposób:

1. Sterownik wysyła komendę do enkodera
2. Enkoder odpowiada komendą i treścią

Komendy są wysyłane do robota około cztery razy na milisekundę, gdy sterownik jest w trybie normalnej pracy (wysyła zapytania o położenie) transfer danych wynosi około 480 kbs.

Komendy

Komendy są to jednobajtowe wartości. Poniżej przedstawiono ramki odpowiedzi dla różnych komend, które są wysyłane podczas normalnej pracy robota. Można zauważyć, że w każdej ramce odpowiedzi istnieją pola o ustalonej funkcji (oznaczone na pomarańczowo): pierwszy bajt jest zawsze numerem komendy, a drugi jest zawsze wartością 0x01. Znaczenie bajtów oznaczonych na żółto jest nieznane.

Wyślij swoje ID 0x92

Odpowiedzią jest numer identyfikacyjny enkodera, jest to wartość dwubajtowa. Ta komenda wysyłana jest podczas startu sterownika robota. Faza ta trwa zwykle około 15s zob. rysunek B.2.

0x92	0x01	D0	D1

Rys.B.2 Numer identyfikacyjny enkodera

Wyślij sumę impulsów 0x1A

Odpowiedzią jest ilość impulsów wraz z dodatkowymi danymi diagnostycznymi zob. rysunek B.3

0x1A	0x01	D0	D1	D2	D3	D4	0x00	Timer	Counter+4	CRC

Rys.B.3 Suma impulsów

Ilość impulsów można odczytać po zastosowaniu poniższej formuły:

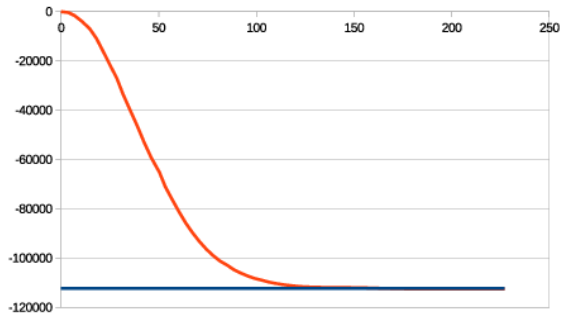
$$IC = (D3 << 20 \mid (D2 \& 0xF) << 16 \mid D1 << 8 \mid D0) >> 2$$

Aby obliczyć pozycję absolutną, należy znać ilość impulsów w zerze enkodera. Następnie od otrzymanej z wcześniejszego wzoru ilości należy odjąć ilość impulsów w zerze:

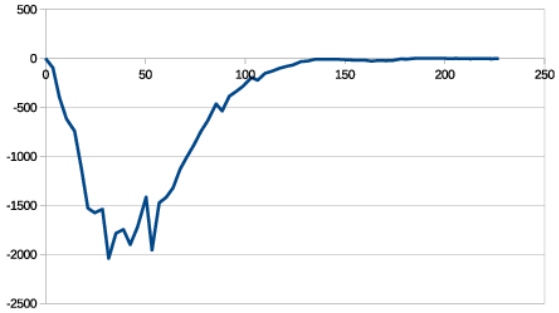
$$IC_{absolute} = IC - IC_{base}$$

Dodatek C Przebiegi eksperymentalne robota Mitsubishi RV-2F-D

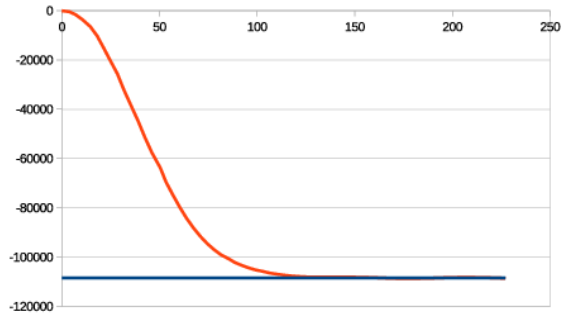
Na rysunku C.1 przedstawiono przykładowy przebieg pomiarowy dla wymuszenia skokowego. Na rysunkach zaznaczono zarówno położenia jak i prędkość poszczególnych członów robota.



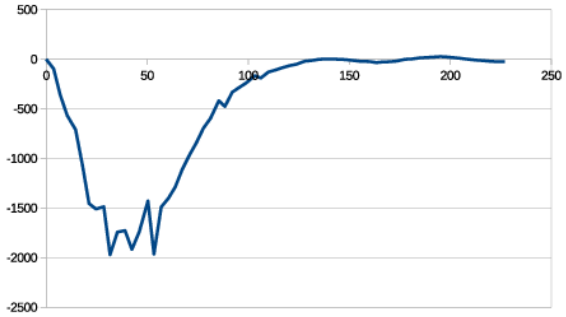
oś 1 położenie



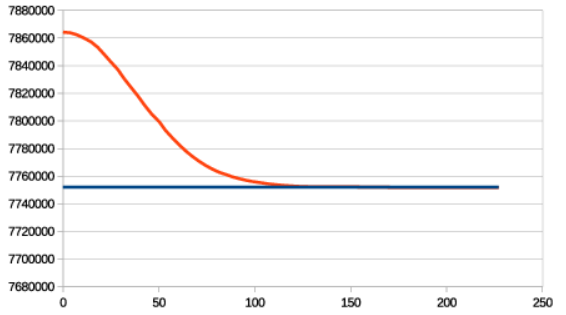
oś 1 prędkość



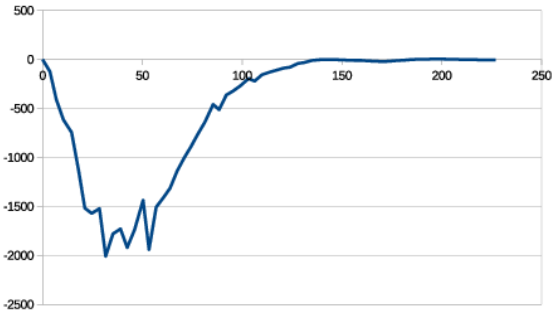
oś 2 położenie



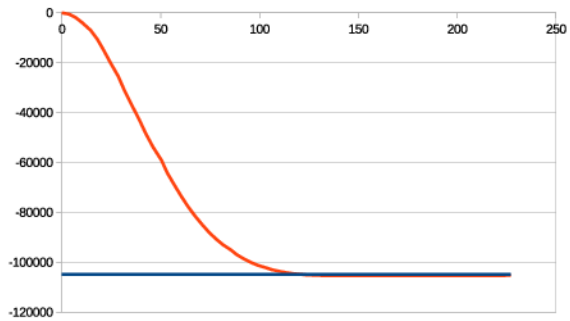
oś 2 prędkość



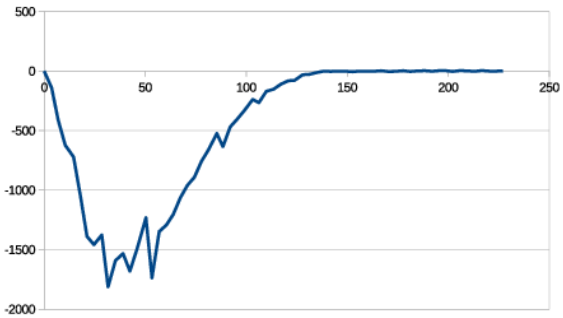
oś 3 położenie



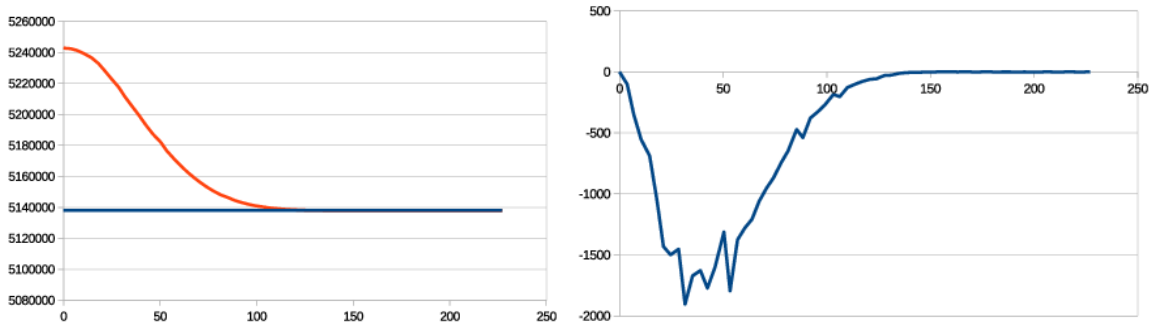
oś 3 prędkość



oś 4 położenie

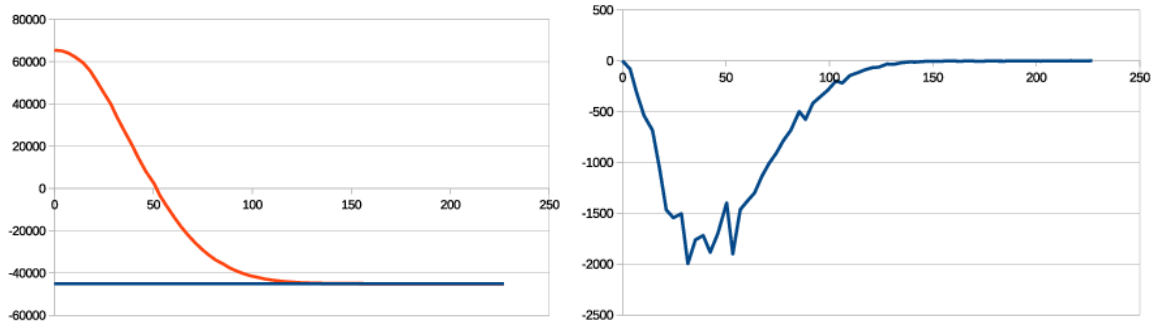


oś 4 prędkość



oś 5 położenie

oś 5 prędkość

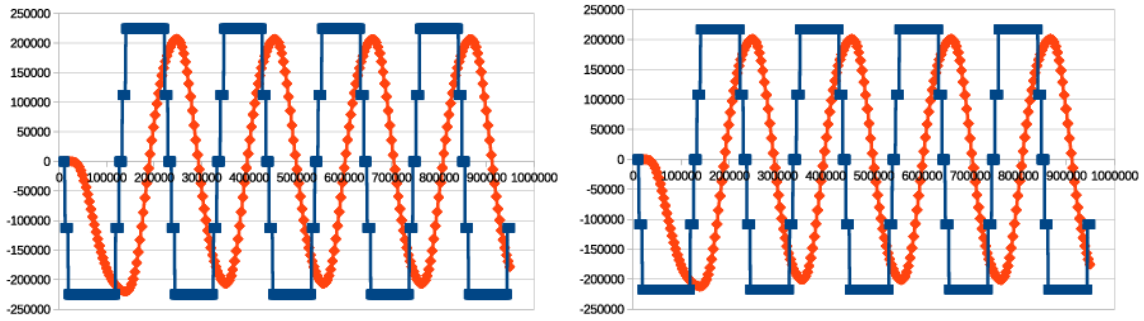


oś 6 położenie

oś 6 prędkość

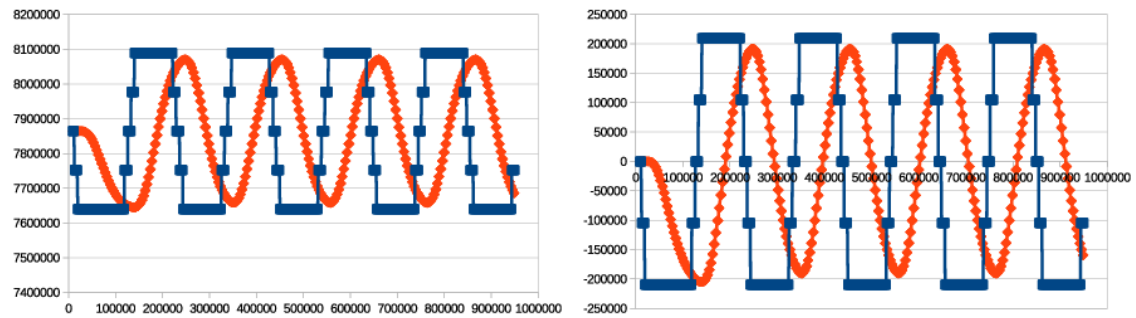
Rys. C.1 Odczyty enkoderowe dla wymuszenia skokowego, położenie oraz prędkość na poszczególnych osiach

Na rysunku C.2 przedstawiono przykładowy przebieg pomiarowy dla wymuszenia prostokątnego. Można na nim zaobserwować, że skok wartości zadanej realizowany jest stopniowo. Wynika to z konstrukcji sterownika robota.

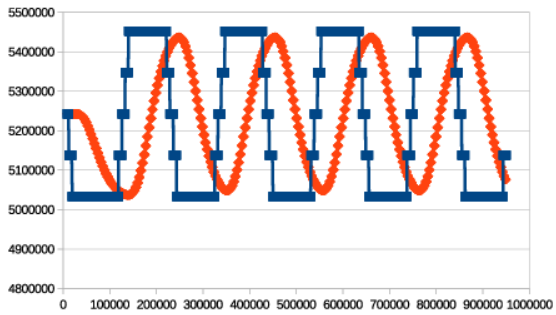


oś 1

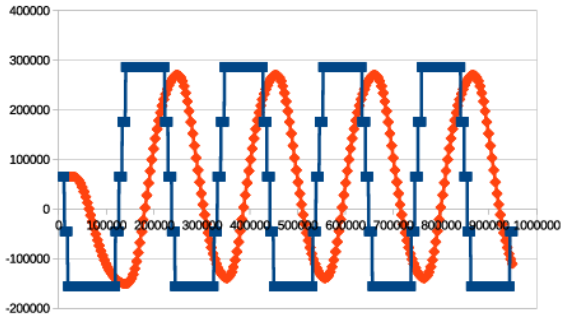
oś 2



oś 3



oś 4



oś 5

oś 6

Rys. C 2: Odczyty enkoderowe dla wymuszenia prostokątnego

Dzięki pomiarom zamieszczonym powyżej, można dokonać obserwacji, że robot pomimo trudności w sterowaniu, może zostać wprowadzony w oscylacje na każdej osi. Dodatkowo można zaobserwować pewne podobieństwa przebiegów do przebiegów obiektów liniowych. Na tej podstawie autor wysunął tezę, że sterownik dokonuje linearyzcji w pętli sprzężenia zwrotnego, co pozwala na traktowanie manipulatora wraz ze sterownikiem jako obiekt liniowy.

Dodatek D Implementacja obliczania funkcji kary

Do implementacji użyto języka C++14 oraz biblioteki Armadillo.

```
double penalty(arma::vec const& point, std::array<float,8> const& start, arma::vec const& end)
{

 double penalty = 0.0;

 // velocity penalty

 for(int i=0; i < 6; i++)
  penalty += 10.0*std::pow((start[i]-point[i])*0.03,4);

 // endpoint penalty

 for(int i=0; i < 6; i++)
  penalty += 8.0*std::pow((end[i]-point[i]),2);

 // obstacle penalty

 for(Obstacle const &obs : globalObstacles)
 {
  double obstaclePen = 1.0;
  double const static obsEdgeCoeff = 10.0;

  for(int j=0; j < 6; j++)
   obstaclePen *=
      std::atan(obsEdgeCoeff*(point[j]-(obs.center[j]-obs.radious[j]))) +
      - std::atan(obsEdgeCoeff*(point[j]-(obs.center[j]+obs.radious[j])));

  penalty += 0.005*obstaclePen;
 }

 return penalty;
}
```

Implementacja poszukiwania na kierunku metodą złotego podziału

```
double penaltyDirection(double pos,
                        arma::vec const& currentResult,
                        arma::vec const& direction,
                        std::array<float,8> const& start,
                        arma::vec const& end)
{
  return penalty(currentResult+direction*pos, start, end);
}

template<typename F>
double GoldenRatioMethod(double a, double b , F f)
{
 const static double k = ( sqrt( 5 ) - 1 ) / 2;
```

```
  double xL = b - k * ( b - a );
  double xR = a + k * ( b - a );

  while ( ( b - a ) > 0.01 )
  {
   if ( f( xL ) < f( xR ) )
   {
    b = xR;
    xR = xL;

    xL = b - k * ( b - a );
   }
   else
   {
     a = xL;
     xL = xR;

     xR = a + k * ( b - a );
   }
  }

  return ( a + b ) / 2;
}

arma::vec const powellMethod(MonitorData const& feedback, arma::vec const& point)
{

  arma::vec result = feedback.jnt;

  arma::mat dir(8,8);
  dir.eye();

  using namespace std::placeholders;

  for(int iter=0; iter < 6; iter++)
  {
   arma::vec oldResult = result;

   for(int i=0; i < 6; i++)
   {
    auto currentDirection = dir.col(i);

    auto fun = std::bind(penaltyDirection,
                         _1,
                         std::cref(result),
                         std::cref(currentDirection),
                         std::cref(feedback.jnt),
                         std::cref(point));

    double goldenResult = GoldenRatioMethod(-0.05, +0.05, fun);

    result = result + currentDirection*goldenResult;
   }
```

```
    double delta = arma::norm(result - oldResult);

    if(delta < 0.01)
      break;

    dir.col(iter%6) = (result - oldResult)/delta;
  }

  return result;
}
```

Struktury danych

```
using Cartesian = std::array<float, 8>
using Joint = std::array<float, 8>
using Encoder = std::array<int32_t, 8>
using Current = std::array<int32_t, 8>

union MonitorData {
  uint8_t raw[40]; //raw data from the robot
  Cartesian pos;     //XYZ type [mm/rad]
  Joint jnt;         //Joint type [rad]
  Encoder pls;       //Pulse type [pls]
  Current cur;       //Integer type [% / non-unit]

  MonitorData() { memset( this, 0, sizeof( MonitorData ) ); }
};

struct Obstacle
{
    arma::vec center;
    arma::vec radious;
};

using Obstacles = std::vector<Obstacle>;
```

Oświadczenie autora

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».”, oświadczam, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i że nie korzystałem ze źródeł innych niż wymienione w pracy.