Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Informatyki

Praca dyplomowa magisterska

na kierunku Informatyka
w specjalności Inżynieria Systemów Informatycznych

System robotyczny chwytający obiekty

Aleksandra Karbarczyk
Numer albumu 246404

promotor
dr inż. Tomasz Winiarski

Warszawa 2017

Streszczenie

Tytuł: *Chwytanie obiektów ramieniem robota przy siłowych algorytmach sterowania*

Celem pracy magisterskiej była implementacja algorytmu kompensującego wpływ siły grawitacji chwytanego przedmiotu o nieznanych parametrach masy i inercji w robocie sterowanym siłowo.

Wykorzystywany do badań robot Velma zbudowany jest z dwóch ramion LWR-4 osadzonych na ruchomym korpusie. Oprogramowanie robota zostało zaprojektowane przy pomocy teorii agenta upostaciowionego i zaimplementowane przy użyciu struktury ramowej FABRIC. Robot korzysta z impedancyjnego prawa sterowania i samodzielnie kompensuje siłę grawitacji ramion.

W pracy zaprezentowano trzy możliwe rozwiązania zarysowanego problemu oraz zaimplementowano jedno z rozwiązań na symulatorze robota. Przedstawiono działanie systemu po modyfikacji. Zaproponowano możliwe kierunki rozwoju w kontekscie przeprowadzonych badań.

Słowa kluczowe: *Velma, FABRIC, ROS, LWR*

Abstract

Title: *Robotic object grasping system*

The aim of t

Keywords: *grasping, localization, IRp-6 robot, ROS, DisCODe, OpenCV*

załącznik do zarządzenia nr 28/2016 r.
Rektora PW

Politechnika Warszawska

„załącznik nr 3 do zarządzenia nr 24/2016 Rektora PW

.....
miejscowość i data

.....
imię i nazwisko studenta

.....
numer albumu

.....
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta"

Spis treści

Rozdział 1

Wstęp

Współczesne roboty z powodzeniem zastępują człowieka przy wielu żmudnych i niewdzięcznych czynnościach. W zakładach produkcyjnych sprawują się znakomicie. Mimo tego podchodzi się niezwykle ostrożnie do współpracy robotów z ludzmi. Podyktowane to jest nie tylko wysokimi kosztami ale także obawami związanymi z ochroną otoczenia w którym robot ma operować.

Zakres zastosowań robotyki można znacznie poszerzyć projektując roboty bezpieczne zarówno dla otaczającego je środowiska jak i samych robotów. Jednym z fundamentów takiego bezpieczeństwa może być algorytm sterowania impedancyjnego, który znacznie ogranicza ryzyko zniszczeń.

Prawo sterowania algorytmu jest skonstruowane tak by silniki ramienia działały w sposób symulujący układ ze sprężyną i amortyzatorem. Robot sterowany w taki sposób nie minimalizuje uchybu pozycji za wszelką cenę. W momencie kontaktu z otoczeniem robot pozostaje elastyczny i ugina się. Opisana zaleta może przerodzić się w wadę. Algorytm korzysta ze zdefiniowanego modelu który nie uwzględnia chwytanych przez robota przedmiotów. Z punktu widzenia algorytmu sterowania chwycony przedmiot jest traktowany tak samo jak reszta środowiska. Zamiast skompensować siłę grawitacji chwyconego przedmiotu ramię robota ugina się pod ciężarem.

Celem pracy jest znalezienie metody kompensacji siły grawitacji przedmiotu o nieznanej masie i inercji w opisanym środowisku. W niniejszej pracy zostanie przeprowadzona dyskusja na temat możliwych sposobów radzenia sobie ze wspomnianymi wadami algorytmu sterowania impedancyjnego. Zostanie zaprezentowana praktyczna realizacja jednego z nich wraz z testami.

Rozdział 2

Podstawy teoretyczne

Dyskutowane w pracy algorytmy znane są w wielu różnych odmianach. Poniżej zaprezentowano wersje używane w trakcie dalszych badań.

Do dalszych rozważań możemy zdefiniować uchyb jako:

$$\boldsymbol{e_x} = \boldsymbol{x_d} - \boldsymbol{x} \tag{2.1}$$

gdzie:

- $\boldsymbol{x_d}$ to wektor zadanych pozycji uogólnionych
- $\boldsymbol{x}$ to wektor pozycji uogólnionych

2.1 Sterowanie impedancyjne

Prawo sterowania impedancyjnego w przestrzeni operacyjnej sprawia, że chwytak robota zachowuje się jak przytwierdzony do układu ze sprężyną i amortyzatorem. Pojawienie się nieprzewidzianych sił zewnętrznych powoduje, że uchyb pozycji nie jest minimalizowany za wszelką cenę. Przy kontakcie z otoczeniem stawy robota w pewnym stopniu sprężyste i miękkie. W konsekwencji robot ugina się przed otaczającym go środowiskiem.

2.1.1 Przypadek prosty

W najprostszym przypadku możemy więc opisać takie prawo jako układ:

$$F = kx + d\dot{x} + F_{ext} \tag{2.2}$$

gdzie:

- F to siła wynikowa

- k to parametr sztywności
- d to parametr tłumienia
- F_{ext} to nieznana siła zewnętrzna działająca na układ

2.1.2 Prawo sterowania

Można sformułować prawo sterowania jako wektor siły uogólnionej:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{K_x e_x} + \boldsymbol{D_x \dot{e}_x} \tag{2.3}$$

gdzie:

- $\boldsymbol{K_x}$ to diagonalna macierz sprężystości
- $\boldsymbol{D_x}$ to diagonalna macierz tłumienia
- $\boldsymbol{\mathcal{F}_{ext}}$ to wektor nieznanych uogólnionych sił zewnętrznych

2.1.3 Sterowanie w przestrzeni konfiguracyjnej

W rzeczywistym ramieniu robotycznym zadajemy momenty na poszczególne stawy robota. Opisane w podrozdziale 2.1.2 prawo sterowania opisuje przestrzeń operacyjną $\boldsymbol{\mathcal{F}}$. Można uzyskać porządane wartości wektora momentów $\boldsymbol{\tau}$ które zadajemy silnikom w stawach stawie wyliczamy z jakobianu $\boldsymbol{J}$:

$$\boldsymbol{\tau} = \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{\mathcal{F}} \tag{2.4}$$

2.2 Sterowanie PID

PID jest bardzo popularnym algorytmem sterowania automatycznego. Podstawowym celem algorytmu jest minimalizacja uchybu. Uchyb statyczny jest minimalizowany za wszelką cenę nawet jeśli miałoby to spowodować uszkodzenia. Uchyby dynamiczne nie są już tak dobrze kompensowane.

Człon proporcjonalny algorytmu pozwala na wzmocnienie uchybu i w ten sposób odjęcie go od sygnału sterującego. Człon całkujący algorytmu sumuje przeszłe błędy i odejmuje od sterowania ich sumę. Człon różniczkujący wzmacnia sygnał sterujący w gdy wartość błędu zmienia się w celu przyspieszenia regulacji.

2.2.1 Przypadek prosty

W jednowymiarowym przypadku prawo sterowania jest postaci:

$$F = Pe + I\int_0^t edt + D\frac{de}{dt} \tag{2.5}$$

gdzie:

- e to uchyb
- P to parametr członu proporcjonalnego
- I to parametr członu całkującego
- D to parametr członu różniczkującego

2.2.2 Przypadek wielowymiarowy

Rozpatrując wektor siły uogólnionej możemy założyć że prawo sterowania rozpatruje każdą z wartości wektora niezależnie. Prawo sterowania można zapisać w postaci:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{P}\boldsymbol{e}_{\boldsymbol{x}} + \int_0^t \boldsymbol{I}\boldsymbol{e}_{\boldsymbol{x}}dt + \boldsymbol{D}\dot{\boldsymbol{e}_{\boldsymbol{x}}} \tag{2.6}$$

gdzie:

- $\boldsymbol{P_x}$ to diagonalna macierz proporcjonalności
- $\boldsymbol{I_x}$ to diagonalna macierz członu całkowania
- $\boldsymbol{D_x}$ to diagonalna macierz członu różniczkującego

2.3 Estymacja siły uogólnionej w końcówce

Dla ramienia robotycznego możemy opisać siły występujące w samym ramieniu zgodnie ze wzorem:

$$\boldsymbol{\mathcal{F}_m}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}, \boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{\Lambda}(\boldsymbol{q})\ddot{\boldsymbol{x}} + \boldsymbol{\mu}(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \boldsymbol{\gamma}(\boldsymbol{q}) + \boldsymbol{\eta}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\mathcal{F}_{ext}} \tag{2.7}$$

gdzie:

- $\boldsymbol{\mathcal{F}}$ to wektor sił wynikowych
- $\boldsymbol{\Lambda}$ to dodatnio określona macierz inercji w przestrzeni zadań
- $\boldsymbol{\mu}$ to macierz sił Coriolisa i sił odśrodkowych

- $\boldsymbol{\gamma}$ to wektor sił grawitacji
- $\boldsymbol{\eta}$ to macierz sił tarcia oraz nieuwzględnionych sił
- $\boldsymbol{q}$ to wektor położeń stawów w przestrzeni konfiguracyjnej
- $\boldsymbol{x}$ to wektor położeń końcówki w przestrzeni zadań
- $\boldsymbol{\mathcal{F}_{ext}}$ to nieznany wektor sił zewnętrznych działających na układ

Przy wyliczaniu estymowaniu sił działających na końcówkę należy pamiętać, że w końcówce występują siły wygenerowane przez prawo sterowania oraz rzeczywiste siły występujące w układzie. Wzór estymujący rzeczywistą wartość siły uogólnionej w końcówce można zapisać jako:

$$\hat{\boldsymbol{\mathcal{F}}} = \boldsymbol{\mathcal{F}} + \boldsymbol{\mathcal{F}_m}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}, \boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{2.8}$$

gdzie $\boldsymbol{\mathcal{F}}$ to wektor sił uogólnionych wyliczony prawem sterowania.
QUESTION: czy rozwijac ta mysl skoro teog nie implementuje?

2.4 Ocena jakości algorytmów sterowania

2.4.1 Jakość sterowania

Prostym sposobem oceny jakości algorytmu sterowania jest konfrontacja rzeczywistych pozycji ramienia robota z zadanymi. W pracy przyjęto metrykę APE (ang. Absolute Trajectory Error). Metryka jest popularnym wskaźnikiem testowania algorytmów SLAM (ang. Simultaneous Localization and Mapping) ale może być też użyta do porównania trajektorii zadanej przez interpolator i rzeczywistej.

Dwie trajektorie są opisane w postaci list wektorów sił uogólnionych $\boldsymbol{P}_{1..n}$ oraz $\boldsymbol{Q}_{1..n}$ gdzie n to ilość próbek. Dla każdej chwili czasowej i jest wyliczany błąd postaci:

$$\boldsymbol{E}_i = \boldsymbol{Q}_i^{-1} \boldsymbol{S} \boldsymbol{P}_i \tag{2.9}$$

Macierz $\boldsymbol{S}$ jest optymalnym w sensie metody najmniejszych kwadratów rzutowaniem wektora $\boldsymbol{Q}_i$ na wektor $\boldsymbol{P}_i$ znalezionym za pomocą metody Horna TODO: cyowania.

Błąd całkowity jest wyliczany jako błąd średniokwadratowy:

$$RMSE(\boldsymbol{E}_{1..n}) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} ||\boldsymbol{E}_i||^2} \tag{2.10}$$

.

2.4.2 Jakość kontaktu z otoczeniem

Największą zaletą sterowania impendancyjnego jest ugięcie ramienia robota w momencie kolizji ze środowiskiem. Ocena jakości tej cechy może być opisana jako odchylenie pozycji stawów $\boldsymbol{q}_i$ w stosunku do ustalonej pozycji sprzed kolizji $\boldsymbol{q}$ w chwili i.

Błąd całkowity jest wyliczany jako błąd średniokwadratowy:

$$RMSE(\boldsymbol{R}_{1..n}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} ||\boldsymbol{q_d} - \boldsymbol{q}_i||^2} \tag{2.11}$$

.

2.5 Teoria agenta upostaciowionego

Agentem możemy nazwać jednostkę która jest: zdolna do komunikowania się ze środowiskiem, zdolna do monitorowania swego otoczenia i podejmowania autonomicznych decyzji. Taka definicja gwarantuje spełnienie wielu cech których oczekujemy od nowoczesnych systemów robotycznych. Z definicji agenty są w stanie samodzielnie reagować na zmiany zachodzące w środowisku w sposób inteligentny. Rozszerzeniem teorii agentowej jest teoria agenta upostaciowionego. Agent upostaciowiony cechuje się tym czym zwykły agnet oraz posiada fizyczne ciało.

W teorii agenta upostaciowionego występują pojęcia rzeczywistych efektorów i receptorów. Służą one odpowiednio do oddziaływania na środowisko i do pozyskiwania wiedzy o tym środowisku. W praktyce oznacza to, że rzeczywistym efektorem może być silnik a rzeczywistym receptorem enkoder.

Agent a może się składać z trzech podsytemów:

- Podsystemu sterowania c który zajmuje się podejmowaniem decyzji na podstawie danych z innych podsystemów. Agent może mieć tylko jeden podsystem sterowania.
- Wirtualnego efektora e który pośredniczy w komunikacji pomiędzy podsystemem sterowania i rzeczywistym efektorem.
- Wirtualnego receptora r który pośredniczy w komunikacji pomiędzy podsystemem sterowania i rzeczywistym receptorem.

Podsystemy składają się z komponentów czyli algorytmów. Każdy z podystemów może mieć wiele komponentów i nie wszystkie muszą być uruchomione w konkretnej chwili. Agent posiada też zdefiniowaną maszynę stanów które mogą się zmieniać przy pomocy funkcji przejścia (predykatów). Stan maszyny stanów definiuje tak

zwane zachowanie agenta czyli sposób reakcji podsystemu sterowania i uruchomione komponenty. Dodatkowo agent ma możliwość wymiany danych innymi agentami oraz rzeczywistymi efektorami i receptorami poprzez bufory transmisyjne T.

Rozdział 3

Środowisko badawcze

3.1 Budowa ogólna

Środowiskiem badawczym jest robot usługowy Velma. Został zaprojektowany i wykonany przez Zespół Rrogramowania Robotów i Systemów Rozpoznających. Do obrotowego korpusu przytwierdzono dwa ramiona robotyczne Kuka LWR-4+. Mają siedem stopni swobody i udzwig 7 kg. Na ich końcach znajdują się chwytaki Barretta oraz nadgarstkowe czujniki FTS. Głowa robota umieszczona jest na dedykowanej konstrukcji która za pomocą dwóch silników elektrycznych pozwala na zginianie i obrót głowy. Robot wyposażony jest w czujnik wizyjny Microsoft Kinect oraz dwie kamery połączone w stereoparę. Do głowy zamocowano mikrofon. Ramiona robota są połączone z komputerem sterującym przy pomocy magistral FRI a pozostałe przy pomocy magistrali EtherCAT.

System sterowania o twardych ograniczeniach czasowych pracuje z częstotliwością 500 Hz. Struktura oprogramowania została stworzona w oparciu o teorię agentową. Oprogramowanie robota pisane jest przy wykorzystaniu struktury ramowej FABRIC. Programy nadzoruje system Linux z nakładką Linux-RT. Dostępny jest symulator robota stworzony w przy wykorzystaniu Gazebo i silnika fizyki DART.

3.2 Pakiety oprogramowania

3.2.1 ROS

Struktura ramowa ROS (Robot Operating System) [1] zapewnia biblioteki usprawniające pisanie programów dla robotyki w C++ i Pythonie. Pozwala na komunikację pomiędzy programami na zasadzie tematów oraz na zasadzie usług. Posiada gotowe funkcje liczące kinematykę i wizualizujące pracę robota. Daje możliwość akwizycji danych. Podstawowymi narzędziami w pakiecie są:

- **Węzły** - Programy pisane w C++ lub Pythonie spełniające zdefiniowaną w sytemie funkcję.

- **Serwisy** - Usługi udostępniane przez węzły pozwalające na komunikację w architekturze zapytanie odpowiedź. Każdy program może udostępniać wiele serwisów. Służą do wywoływania zdalnych procedur.

- **Akcje** - Usługi podobne do serwisów lecz nieblokujące wykonywania węzła będącego serwerem.

- **Tematy** - Usługi udostępniane przez węzły pozwalające na asynchroniczną komunikację. Każdy program może udostępniać i pobierać wiele tematów. Służą do aktualizowania bierzącego stanu całego systemu.

- **Zarządca** - odpowiada za komunikację i nadzoruje pracę innych narzędzi pakietu.

3.2.2 Orocos

Orocos jest wolnym oprogramowaniem napisanym w C++ służącym do pisania aplikacji zgodnych z wymaganiami czasu rzeczywistego. Pozwala na tworzenie komponentów odwzorowujących modele stworzone przy pomocy teorii agentowej. Zestaw bibliotek składa się między innymi z:

- **RTL (ang. Real-Time Toolkit)** - Biblioteki służące do tworzenia komponentów w aplikacjach czasu rzeczywistego.

- **OCL (ang. Orocos Component Library)** - Biblioteki pomocne w trakcie uruchamiania komponentów.

- **OroGen** oraz **TypeGen** - Narzędzia do automatycznego generowania komponetów i typów danych.

- **Deployer** - Uruchamia komponenty zgodnie z opisem zawartym w pliku XML oraz pozwala na nadzór tych komponentów trakcie pracy.

3.2.3 FABRIC

Framework for Agent–Based Robot Control Systems - FABRIC [2] wykorzystuje Orocosa oraz strukturę ramową ROS zapewniając interfejs programistyczny pozwalający na tworzenie komponentów zgodnych z założeniami teorii agentowej.

Ma zaimplementowane algorytmy komunikacji pomiędzy poszczególnymi podsystemami poprzez zdefiniowane wiadomości. Posiada narzędzie wizualizujące stan predykatów, zachowań i podsystemów *rqt_ agent*. Pokazuje też przepływ danych między komponentami.

3.3 Symulator robota

Symulator robota Velma jest wytworzony w oprogramowaniu Gazebo które symuluje obiekty i zachowania między nimi zgodnie z prawami fizyki. Użytkownik ma możliwość zdefiniowania całego środowiska wraz z robotem. Posiada gotowe modele wielu receptorów i efektorów oraz daje możliwość tworzenia własnych. Daje możliwość emulowania sterowników sprzętu. Pozwala na wywarcie siły bądź momentu na dowolny przedmiot będący w symulacji. Emuluje czas jeśli symulacja nie jest przeprowadzana w czasie rzeczywistym.

Orginalny kod Gazebo został zmodyfikowany przez Zespół programowania Robotów i Systemów Rozpoznających aby można było korzystać z pakietu oprogramowania DART (ang. Dynamic Animation and Robotics Toolkit) symulującego fizykę zdefiniowanego świata. Pakiet DART został wybrany przez Zespół programowania Robotów i Systemów Rozpoznających ponieważ po jego zastosowaniu został wyeliminowany problem narastających oscylacji członów robota w trakcie symulacji. W przeciwieństwie do wielu symulatorów fizyki daje dostęp do wielu przydatnych informacji takich jak jakobian bądź macierz inercji przemiotów. Zastosowanow w nim algorytm LPC (ang. Linear complementarity problem) który sprowadza model fizyczny świata do zestawu równań kwadratowych. DART symuluje działanie wielu sił w tym Coriolisa i tarcia dynamicznego. Dzięki temu możliwe jest zaawansowane wykrywanie kolizji.

3.4 Agenty

System sterowania zbudowany jest z dwóch agentów. Agent *velma_ core* jest odpowiedzialny za kontrolę zadań związanych z manipulacją w przestrzeni operacyjnej i konfiguracyjnej robota. Drugi z agentów *velma_ task_ cs_ ros_ interface* jest interfejsem pomiędzy programami użytkownika pisanymi w ROS oraz agentem *velma_ core*. Zawiera tylko jeden podsystem o tej samej nazwie.

Podsystem *velma_ core_ cs* agenta *velma_ core* wylicza prawa sterowanie oraz zajmuje się interpolacją trajektorii. Podsytem *velma_ core_ ve_ body* kontroluje bazowe zachowania bezpieczeństwa. Są to ograniczenia prądowe, wykrywanie krańcowych położeń stawów oraz wykrywanie kolizji. Podsystem przekazuje też sterowanie do efektorów. Podsystemy w najniższej warstwie abstrakcji mogą być sto-

sowane wymiennie. Do pracy w rzeczywistym świecie uruchamiane są podsystemy *velma_core_re_lwr_r* i *velma_core_re_lwr_l* oraz podsystem *velma_ec_driver*. Służą odpowiednio do kontroli prawego i lewego ramienia LWR-4 oraz pozostałego sprzętu połączonego magistralą EtherCAT. Do pracy w trybie symulacji podsystemy w najniższej warstwie abstrakcji są wymieniane na podsystem *velma_sim_gazebo* w którym uruchamiany jest symulator świata Gazebo. Podsystem symuluje pracę wszystkich trzech podsystemów odpowiedzialnych za komunikację ze sterownikami sprzętu.

- **Interfejs akcji ROS** - Komponent *CartImpActionRight* znajduje się w podsytemie *velma_task_cs_ros_interface* i odbiera rozkazy wysłane przez użytkownika przez mechanizm akcji ROS oraz przesyła je do innych podsytemów. Zwraca też status wykonania operacji. Komponent służy do obsługi prawego ramienia i istnieje analogiczny komponent *CartImpActionLeft* służący do obsługi lewego ramienia.

- **Publikacja wektorów pozycji** - Komponent *TfPublisher* wysyła do użytkownika za pomocą ROSa położenie i obrót istotnych dla działania systemu miejsc robota takich jak chwytaki czy środek ciężkości korpusu.

- **Pozycje stawów** - Odczyt pozycji stawów jest zależny od trybu pracy oprogramowania. Jeśli używamy trybu obsługi rzeczywistego sprzętu to za odczyt pozycji są odpowiedzialne podsystemy *velma_core_re_lwr_r* i *velma_core_re_lwr_l* oraz podsystem *velma_ec_driver* które posiadają pojedyncze komponenty służące do komunikacji z fizycznymi sterownikami magistral. W trybie symulatora emulacją tych trzech podsystemów zajmuje się podsytem *velma_sim_gazebo* który jako symulator nie jest podzielony na konkretne podsystemy.

- **Zadawanie momentów** - Zadawanie momentów obrotowych w stawach następuje w analogiczny do odczytywania pozycji sposób.

- **Kinematyka prosta** - Komponent *FK* pobiera dane o pozycjach stawów i wylicza pozycję chwytaka oraz innych części robota.

- **Interpolator trajektorii** - Komponent *INT_tool_r* z podsystemu *velma_core_cs* odpowiada za interpolacje trajektorii zadanej prawemu ramieniu. Interpolator służy temu by jedno polecenie przesunięcia ramienia zamienić na wiele mniejszych i rozłożonych w czasie. W konsekwencji algorytm sterowania nie jest narażony na duże uchyby a ruch ramienia jest wykonywany płynnie. Do działania komponent potrzebuje aktualnych i zadanych pozycji.

Komponent służy do obsługi prawego ramienia. Analogicznie do obsługi lewego ramienia służy komponent *INT_tool_l*.

- **Prawo sterowania impedancyjnego** - Komponent *cart_imp* z podsytemu *velma_core_cs* służy do wyliczania momentów zadawanych na stawy zgodnie z algorytmem prawa sterowania impedancyjnego w przestrzeni kartezjańskiej. Pobiera zadaną pozycję z interpolatora trajektorii.

Rozdział 4

Specyfikacja systemu

Sterowanie impedancyjne w przestrzeni operacyjnej zastosowane w robocie Velma kompensuje jedynie masę członów własnych. Gdy manipulator chwyci narzędzie powstaje znaczny uchyb, który nie jest kompensowany. W przypadku chwycenia przedmiotu o dużej masie ramię robota jest wręcz unieruchomione. Należy skonstruować i zaimplementować algorytm kompensacji siły grawitacji narzędzia o nieznanych parametrach.

4.1 Wymagania

Wymagania są narzucone przez system środowiska badawczego.

- Do działania systemu wymagane jest ramię robotyczne sterowane prawem sterowania impedancyjnego w przestrzeni kartezjańskiej
- System dostarcza gotowe komponenty potrzebne do pracy ramienia, w szczególności: generatora trajektorii, interpolatora oraz wyznaczania kinematyki prostej
- System pozyskuje dane o otoczeniu na podstawie odczytów z czujników położenia stawów.
- System samodzielnie kompensuje siłę grawitacji związaną z masą wszystkich członów robota
- Parametry chwytanego obiektu związane z masą i inercją nie są znane i mogą być zmienne w czasie

4.2 Założenia

Algorytm kompensacji grawitacji musi wyliczać dodatkowy moment w stawach robota. Do momentów obrotowych wyliczanych na postawie prawa sterowania impendancyjnego w przestrzeni operacyjnej robota należy dodać te związane z kompensacją masy narzędzia.

- Wyliczone przez prawo sterowania momenty zadawane w stawach zostaną zmodyfikowane przez algorytm kompensacji grawitacji
- Algorytm kompensacji ma minimalizować uchyb statyczny wynikający z chwycenia przedmiotu o nieznanej masie i inercji.
- Trajektoria końcówki ramienia powinna być zbliżona do trajektorii analogicznego ruchu bez uchwyconego przedmiotu.
- Algorytm kompensacji nie może zaburzać cech sterowania imedancyjnego pozwalających na uginanie się robota w momencie kolizji

4.3 Przypadki użycia

Algorytm kompensujący ma działać przez cały czas pracy robota gdy pracuje z narzędziem. Przypadki użycia algorytmu (rys.) są zbieżne z przypadkami użycia robota kiedy pracuje w trybie sterowania impedancyjnego.

Aktorami korzystającymi z systemu są:

- **Robot** - robot wykonujący zadanie chwycenia obiektu
- **Użytkownik** - system zewnętrzny wydający polecenia

Można wyszczególnić następujące przypadki użycia:

- **PU1 Manipulacja bez narzędzia** - Konfiguracja ramienia robota może się zmieniać lecz robot nie trzyma żadnego narzędzia. Algorytm kompensacji grawitacji nie powinien zmieniać prawa sterowania.
- **PU2 Chwytanie narzędzia** - Robot zaciska chwytak na narzędziu o nieznanych parametrach a następnie je podnosi.
- **PU3 Manipulacja z narzędziem** - Konfiguracja ramienia może się zmieniać a algorytm kompensacji grawitacji powinien przeciwdziałać sile grawitacji chwyconego narzędzia.
- **PU4 Odkładanie narzędzia** - Robot odkłada narzędzie a następnie rozwiera chwytak.

- **PU5 Kompensacja grawitacji** - Robot kompensuje siłę grawitacji chwyconego narzędzia

Rozdział 5

Implementacja systemu

W ramach pracy zmodyfikowano istniejący system robotyczny opisany w rozdziale 3. Dzięki temu możliwe było spełnienie założeń i wymagań z rozdziału 4 Zmodyfikowano komponenty odpowiedzialne za przekazywanie

5.1 Modyfikacja kodu

5.2 Kompensowanie wpływu grawitacji narzędzia

Kiedy chwytak robota chwyci narzędzie zmienia się łańcuch kinematyczny. W pracy zajmujemy się przypadkiem w którym po uchwyceniu narzędzie zostaje nieruchome w stosunku do chwytaka. Można przyjąć, że zmieniają się wtedy parametry ostatniego członu związane z masą i inercją.

Głównym problemem w trakcie manipulacji z nieskompensowaną grawitacją jest znaczny uchyb statyczny. Aby wyeliminować ten efekt można do istniejącego algorytmu sterowania impedancyjnego dodać algorytm PID.

Należy mieć na uwadze, że niektóre człony są takie same w zaprezentowanych prawach sterowania. W impedancyjnym prawie sterwowania nie ma członu całkującego. W rezultacie nowe prawo sterowania jest postaci:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{K_x e_x} + \boldsymbol{D_x \dot{e}_x} + \int_0^t \boldsymbol{I e_x} dt \tag{5.1}$$

gdzie:

- $\boldsymbol{\mathcal{F}}$ to wektor sił wynikowych
- $\boldsymbol{K_x}$ to diagonalna macierz sprężystości
- $\boldsymbol{D_x}$ to diagonalna macierz sztywności
- $\boldsymbol{I}$ to diagonalna macierz członu całkującego

- $\boldsymbol{e_x}$ to wektor uchybu

Rozdział 6

Działanie systemu

Odnosząc się do wymagań z podrozdziału

Rozdział 7

Podsumowanie

7.1 Wnioski

7.2 Perspektywy rozwoju

Bibliografia

[1] Oficjalna strona systemu ROS. http://www.ros.org/.

[2] Dawid Seredyński, Tomasz Winiarski, Cezary Zieliński. FABRIC: Framework for Agent-Based Robot Control Systems. K. Kozłowski, redaktor, *12th International Workshop on Robot Motion and Control (RoMoCo)*, strony 215–222. IEEE, 2019.

Spis rysunków

Spis tablic