Mateusz Boryń
mgr inż. Tomasz Kornuta
prof. nzw. dr hab. inż. Cezary Zieliński
Instytut Automatyki i Informatyki Stosowanej
Politechnika Warszawska

STRUKTURA RAMOWA DO IMPLEMENTACJI I TESTOWANIA SERWOMECHANIZMÓW WIZYJNYCH[1]

Artykuł prezentuje strukturę ramową, opartą na zrębach MRROC++ i FraDIA, ułatwiającą implementację oraz testowanie różnego rodzaju serwomechanizmów wizyjnych. Dla pełności, poza samą strukturą oraz ogólnym sposobem integracji zrębów, omówiono klasyfikację serwomechanizmów, implementację jednego z nich oraz przedstawiono wyniki eksperymentów.

A FRAMEWORK FOR IMPLEMENTATION AND TESTING OF DIVERSE VISUAL SERVO ALGORITHMS

The paper presents a framework facilitating the development of diverse visual servoing (VS) algorithms. The solution utilizes MRROC++ programming framework (for robot control) and FraDIA vision framework (for image processing, analysis and recognition). The article contains the visual servos classification, implementation of one type of VS and the results of experiments.

1. WSTĘP

Robot pozbawiony zmysłów może sprawnie i szybko wykonywać przydzielone mu prace w dobrze ustrukturyzowanym, deterministycznym środowisku (np. w fabryce). Jednak w przypadku zmieniającego się otoczenia, gdzie pozycja przedmiotów może być niedokładnie określona lub wręcz nieznana, niezbędne jest wyposażenie robota w zmysły (umożliwiające mu rozpoznawanie obiektów, określanie ich położenia itp.), a także odpowiednie zachowania, związane z tymi zmysłami (np. omijanie kolizji z wykrytym obiektem). Ponieważ zmysł wzroku jest najbardziej uniwersalnym (a przez to w rzeczywistości najbardziej skomplikowanym) ze znanych nam zmysłów, dlatego wykorzystanie kamer jako jego zamiennika leżało od zawsze w sferze zainteresowań robotyki [1].

Podstawowe zachowania robotyczne wykorzystują zmysł wzroku to realizacji serwomechanizmu wizyjnego [3], który przetwarza informację wizyjną do śledzenia obiektu zainteresowania. W systemie sterowania robota wykorzystującym serwowizję można wyróżnić dwa podstawowe bloki:

- podsystem wizyjny – blok analizy i rozpoznawania obrazów, którego celem jest agregacja tych danych do postaci użytecznej w sterowaniu,
- podsystem sterujący ruchem robota – który na podstawie danych wyekstrahowanych z obrazów generuje odpowiednie sygnały sterowania robotem.

[1] Praca finansowana przez grant badawczy Ministerstwa Nauki i Szkolnictwa Wyższego N514 1287 33.

W literaturze opisanych jest kilka programowych struktur ramowych ułatwiających tworzenie oraz badanie działania serwomechanizmów wizyjnych. Przykładem takiej struktury jest ViSP (ang. *Visual Servoing Platform*) [5] – modułowe środowisko programistyczne posiadające różne algorytmy wizyjne do rozpoznawania obiektów oraz sterowania wykorzystującego informację wizyjną. Innym przykładem jest Javiss [2], będący środowiskiem do symulacji działania serwomechanizmów wizyjnych. Pozwala na badania zachowania robota śledzącego określony przedmiot, np. poprzez zbieranie pomiarów pozycji końcówki roboczej.

Niniejszy artykuł przedstawia metodę integracji dwóch programowych struktur ramowych w sterownik robota wykorzystujący informację wizyjną. Pierwszą z nich jest FraDIA (ang. *Framework for Digital Image Analysis*) [4], wizyjna struktura ramowa, która umożliwia proste tworzenie różnorodnych zadań wizyjnych. Może ona również pracować jako podsystem wizyjny sterowników systemów wieloobrotowych opartych na drugiej strukturze ramowej – MRROC++ (ang. *Multi–Robot Research Oriented Controler*) [8,9].

W opracowanym rozwiązaniu wykorzystano paradygmat projektowania obiektowego do stworzenia lekkiej, łatwej do rozszerzania i modyfikacji struktury klas realizujących różnorodne serwomechanizmy wizyjne. W strukturze tej zaimplementowano i uruchomiono serwomechanizmy wizyjne różnego typu, natomiast w tym artykule skupiono się na omówieniu jednego z nich. Przeprowadzono również szereg eksperymentów dowodzących elastyczności stworzonej struktury, a ponadto pozwalających na zweryfikowanie poprawności działania danego serwomechanizmu.

2. SPOSOBY KLASYFIKACJI SERWOMECHANIZMÓW WIZYJNYCH

Serwomechanizm jest układem sterowniania o strukturze nadążnego układu regulacji położenia lub prędkości. W układzie tym wartość wzorcowa (zadana) porównywana jest z przetworzonym przez przetwornik bieżącym sygnałem wyjściowym i powstały w ten sposób uchyb podawany jest na blok sterowania. Zadaniem serwomechanizmów wizyjnych VS (ang. *Visual Servo*) [3,6] jest wykorzystanie informacji z kamer do sterowania robotem – kamery stanowią tu element pomiarowy systemu sterującego.
W pracy [7] zaproponowano podział serwomechanizmów według następujących kryteriów:

- sposobu sterowania robotem,
- faktu, czy końcówka robocza jest obserwowana,
- położenia kamery,
- metody wyznaczania uchybu.

Sposób sterowania robotem zależy od przestrzeni, w której obliczany jest uchyb:

- sterowanie poszczególnym i stawami, wykorzystujące uchyb w przestrzeni złączy (konfiguracyjnej) (ang. *joint error control* – JEC),
- sterowanie położeniem końcówki, gdzie uchyb określony zostaje w przestrzeni operacyjnej (ang. *position error control* – PEC).

W zależności od tego, czy fakt widoczności końcówki manipulatora w polu widzenia kamery wykorzystywany jest przez algorytmy rozpoznawania obrazu, serwomechanizmy wizyjne dzieli się na:

- nieobserwujące końcówki robota (ang. *endpoint open–loop* – EOL),
- obserwujące końcówkę robota (ang. *endpoint closed–loop* – ECL).

Najbardziej oczywistym kryterium podziału serwomechanizmów wizyjnych jest sposób umieszczenia kamery obserwującej środowisko pracy robota. Kamera może być nieruchoma

względem podstawy robota (ang. *stand–alone camera* – SAC, w pracy [6] typ ten zwany jest również ETH, od ang. *eye–to–hand*). Wtedy pozycja kamery względem końcówki roboczej zmienia się w czasie pracy manipulatora. Można również przymocować kamerę do końcówki roboczej manipulatora (ang. *eye–in–hand* – EIH).

Stosowane są dwa sposoby wyrażania uchybu dla serwomechanizmów wizyjnych: uchyb w przestrzeni zadaniowej (ang. *position based* – PB) oraz uchyb w przestrzeni cech obrazu (ang. *image based* – IB). W pierwszym przypadku (PB), najpierw w obrazie z kamery wykrywany jest konkretny przedmiot. Wynikiem tej operacji są współrzędne punktów obrazu, które można uznać za punkty charakterystyczne przedmiotu. Znany jest również model przedmiotu, tzn. położenie punktów charakterystycznych w przestrzeni trójwymiarowej. Na podstawie tych dwóch informacji można określić położenie oraz orientację przedmiotu względem kamery, a tym samym pozycję względem podstawy robota (pozycja kamery jest znana). Uchyb jest wynikiem porównania pozycji końcówki roboczej oraz pozycji przedmiotu względem podstawy robota. W drugim przypadku (IB) wybierane są cechy obrazu obiektu, które będą porównywane (np. położenie i wielkość przedmiotu w obrazie). Uchyb jest różnicą wektorów wartości cech zadanych i wartości cech aktualnych.

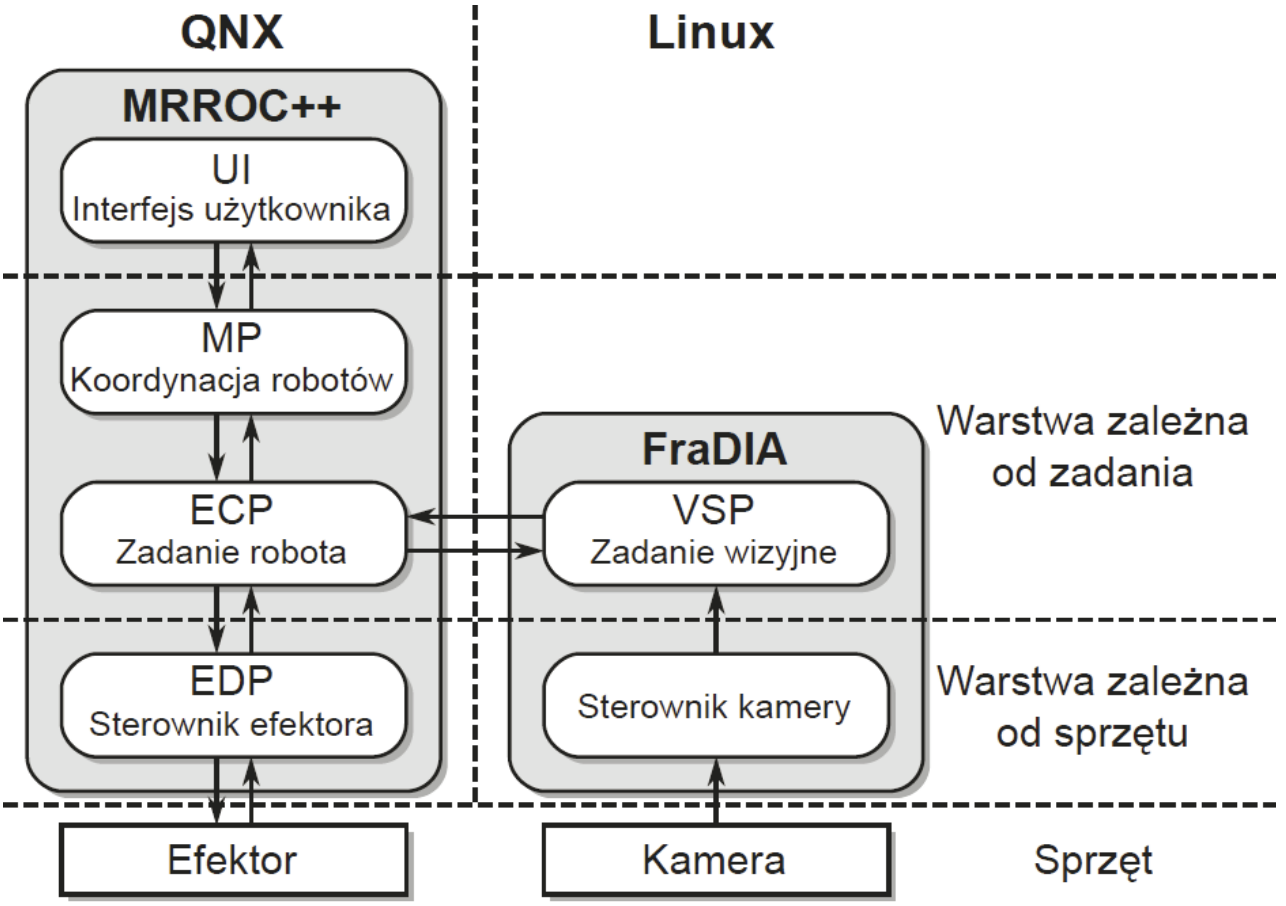
3. INTEGRACJA STRUKTUR RAMOWYCH MRROC++ I FRADIA

Programowa struktura ramowa MRROC++ jest narzędziem do tworzenia hierarchicznych, rozproszonych sterowników systemów wielorobotowych. Została ona z powodzeniem wykorzystana w różnorodnych aplikacjach robotycznych, m.in. do: powielania rysunków, śledzenia konturów obiektów oraz układania kostki Rubika [8].

Zapotrzebowanie na narzędzie ułatwiające oraz standaryzujące proces tworzenia aplikacji wizyjnych doprowadziło do powstania wizyjnej struktury ramowej FraDIA. Zaimplementowane w jej warstwie sprzętowej sterowniki źródeł obrazu pozwalają na skupienie uwagi programisty na właściwych algorytmach analizy oraz rozpoznawania obrazów. Ponieważ zrąb oferuje dodatkowo gotowe mechanizmy do komunikacji z procesami i systemu MRROC++, obie struktury mogą być użyte do konstrukcji złożonych sterowników robotycznych wykorzystujących wizję komputerową (rys. 1).

W skład sterownika systemu robotycznego wykorzystującego strukturę ramową MRROC++ wchodzą podzielone na warstwy procesy, pokazane na rys. 1:

- UI (ang. *User Interface*) – pojedynczy proces interfejsu użytkownika odpowiedzialny, za komunikację z operatorem,
- MP (ang. *Master Process*) – pojedynczy proces koordynujący pracę wszystkich efektorów systemu,
- ECP (ang. *Effector Control Process*) – proces odpowiedzialnych za realizację zadania zleconego pojedynczemu efektorowi,
- EDP (Effector Driver Process) – proces odpowiedzialny za bezpośrednie sterowanie pojedynczym efektorem,
- VSP (Virtual Sensor Process) – proces odpowiedzialny za agregację danych sensorycznych do postaci użytecznej w sterowaniu. W zależności od liczby czujników i rodzaju sprzężenia procesy VSP mogą być skojarzone z ECP lub MP.



Rys. 1. Integracja struktur ramowych MRROC++ i FraDIA w sterownik robota ze sprzężeniem wizyjnym

W przypadku systemu robotycznego wykorzystującego informację wizyjną FraDIA pełni rolę wyspecjalizowanego procesu VSP systemu MRROC++. FraDIA współpracuje nieinteraktywnie z procesami warstwy zadaniowej systemu MRROC++, co oznacza, iż cały czas odbiera oraz przetwarza obrazy z kamery, natomiast MRROC++ wysyła żądanie odesłania zagregowanego odczytu czujnika – a więc najnowszego wyniku przetwarzania i analizy – wtedy, gdy jest mu on potrzebny.

Przykładowe aplikacje robotyczne, w których wykorzystano zintegrowane struktury ramowe MRROC++ i FraDIA to chwytanie obiektów czy gra w warcaby prowadzona z robotycznym przeciwnikiem [4].

4. TRZON STRUKTURY SERWOMECHANIZMÓW WIZYJNYCH

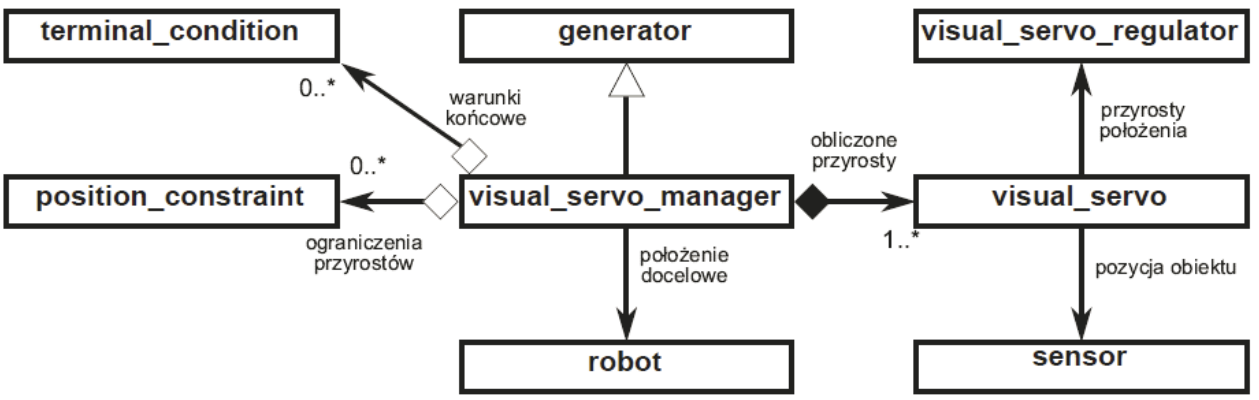
Na rys. 2 przedstawiono główne, bazowe klasy składające się na strukturę serwomechanizmów wizyjnych po stronie procesu ECP systemu MRROC++. Obiekty tych klas są elementami układu regulacji. Ich wykorzystanie zostanie przedstawione w dalszej części artykułu na przykładzie serwomechanizmu wizyjnego PEC-EOL-PB-EIH.

Abstrakcyjne klasy **robot** oraz **sensor** stanowią interfejsy do warstwy sprzętowej – a więc odpowiednio do procesów EDP stanowiących sterowniki manipulatorów oraz procesów VSP związanych z danym czujnikiem. W przypadku omawianej struktury serwomechanizmów procesem VSP jest FraDIA, a wszystkie serwomechanizmy wykorzystują klasę **fradia_sensor**, która jest wywiedziona z klasy **sensor**. Obiekty tej klasy służą do komunikacji pomiędzy strukturami ramowymi MRROC++ i FraDIA i są odpowiedzialne za dostarczanie zagregowanej informacji wizyjnej do sterownika robotycznego.

Klasa **visual_servo** jest klasą bazową dla poszczególnych typów serwomechanizmów. Obiekty klas pochodnych są odpowiedzialne za obliczenie uchybu w sposób specyficzny dla danego rodzaju serwomechanizmu wizyjnego. Informacja o położeniu przedmiotu względem kamery otrzymywana jest z interfejsu **sensor**.

Obiekty klasy abstrakcyjnej **visual_servo_regulator** realizują wybrany algorytm sterowania. Istnieje możliwość zaimplementowania wielu algorytmów regulacji, np. regulatorów P, PID, a także bardziej złożonych, np. algorytmów regulacji predykcyjnej. Dzięki oddzieleniu obliczania uchybu od algorytmu regulacji można łatwo porównać różne algorytmy regulacji działające dla jednego serwomechanizmu wizyjnego.

Klasa abstrakcyjna **generator** jest odpowiedzialna za przygotowanie rozkazu sterującego położeniem końcówki roboczej danego efektora. Z klasy **generator** dziedziczy abstrakcyjna klasa **visual_servo_manager**, która jest generatorem ruchu wchodzącym w skład serwomechanizmów wizyjnych. W tej klasie zaimplementowano ograniczenia prędkości i wartości przyspieszenia końcówki roboczej.



Rys. 2. Diagram zawierający najważniejsze abstrakcyjne klasy struktury serwomechanizmów wizyjnych po stronie systemu MRROC++

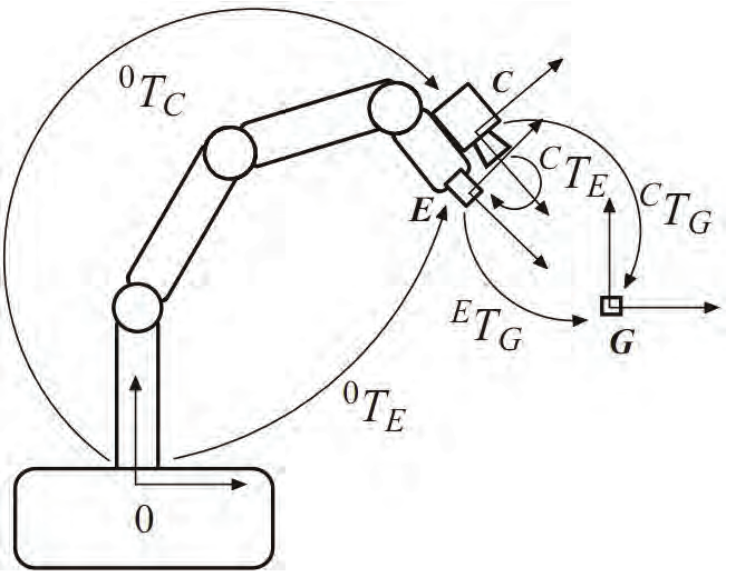
Generator ruchu jest też odpowiedzialny za ograniczenie pozycji końcówki roboczej w przestrzeni zadaniowej robota. W tym celu korzysta z obiektu klasy **position_constraint**, gdzie definiowane są ograniczenia pozycji. Obiekt klasy **visual_servo_manager** może mieć wiele obiektów reprezentujących różne ograniczenia pozycji. Istnieją dwa typy ograniczeń – pierwsze reprezentują obszar, w którym może znajdować się końcówka robocza, oraz orientację, jaką może przyjąć w tym obszarze. Drugi typ ograniczeń związany jest ze znajdującymi się na scenie obiektami, z którymi odgórnie zakłada się, iż robot nie może się zderzyć, przy czym złożony obszar można opisać wykorzystując kilka obiektów klasy **position_constraint**. Przykładowo, gdy zadaniem robota jest dosięgnięcie przedmiotu leżącego na stole, można ograniczyć przestrzeń zadaniową do prostopadłościanu znajdującego się tuż nad powierzchnią tego stołu. Można wtedy mieć pewność, że robot nie spróbuje sięgnąć po przedmiot leżący na podłodze obok stołu, powodując w rzeczywistości kolizję.

Klasa **termination_condition** reprezentuje warunek zakończenia pracy generatora. Generator ruchu może mieć wiele warunków zakończenia. Generator zakończy działanie, gdy dowolny z nich zostanie spełniony. Warunek zakończenia pracy generatora ruchu jest konieczny w zadaniach, gdzie należy np. chwycić obiekt. Przykładowym warunkiem zakończenia może być warunek dosięgnięcia obiektu, który jest spełniony, gdy wartość uchybu serwomechanizmu wizyjnego, szybkość oraz wartość przyspieszenia końcówki roboczej będą bliskie zeru.

5. REALIZACJA SERWOMECHANIZMU WIZYJNEGO PEC-EOL-PB-EIH

Poniżej opisano realizację przykładowego serwomechanizmu wizyjnego, w którym robot jest sterowany uchybem zadaniowym, z kamerą nieobserwującą końcówki roboczej i umieszczoną na końcówce roboczej. Serwomechanizm oblicza uchyb w przestrzeni zadaniowej. Celem pracy prezentowanego serwomechanizmu jest śledzenie szachownicy.

Na rys. 3 przedstawiono manipulator, kamerę oraz śledzony przedmiot z zaznaczonymi pozycjami jednego obiektu względem drugiego.



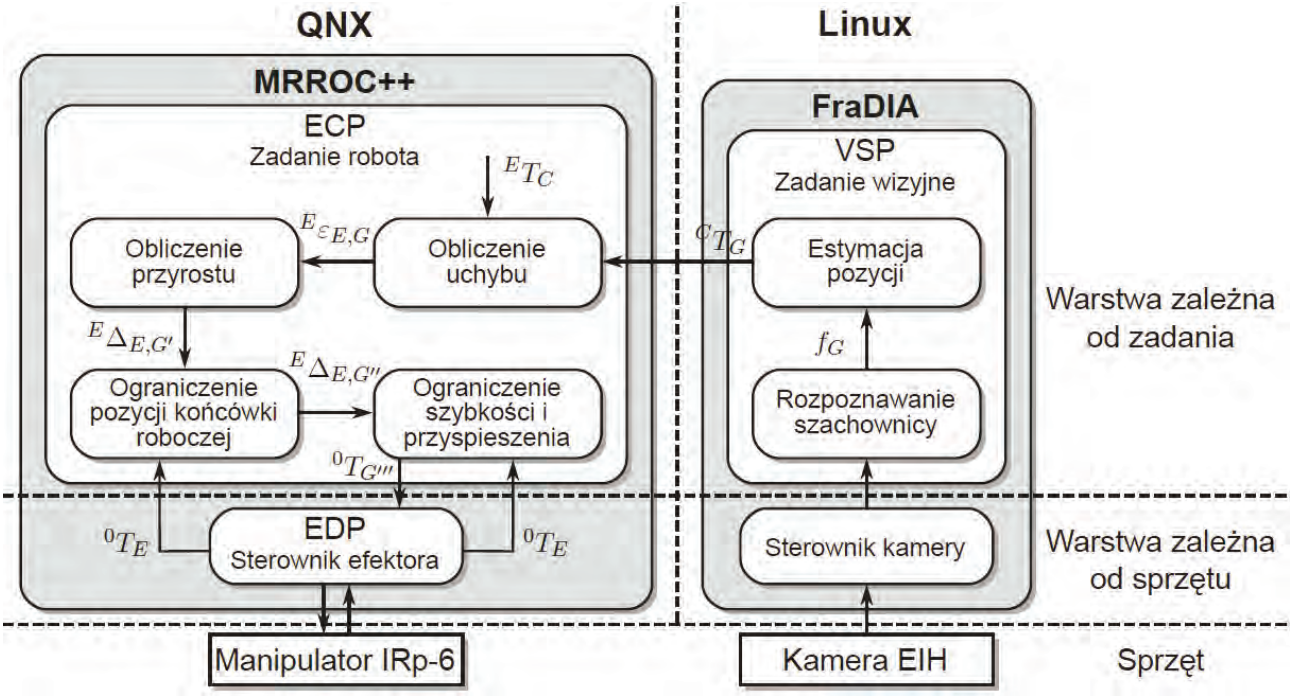
Rys. 3. Układy współrzędnych i transformacje pomiędzy robotem, kamerą oraz przedmiotem

Przyjęto następujące oznaczenia dla różnych układów oraz transformacji: *0* – układ współrzędnych związany z podstawą robota, *E* – układ współrzędnych związany z końcówką roboczą, *C* – układ współrzędnych związany z kamerą, *G* – układ współrzędnych związany z śledzonym przedmiotem, ${}^E T_G$ – pozycja przedmiotu względem końcówki roboczej, ${}^E T_C = const$ – pozycja kamery względem końcówki roboczej wyznaczona w procesie kalibracji oko–ręka, ${}^C T_G$ – pozycja przedmiotu względem kamery.

Na rys. 4 przedstawiono najważniejsze bloki układu sterowania serwomechanizmu wizyjnego PEC-EOL-PB-EIH wraz z danymi przepływającymi pomiędzy nimi.

Informacja o stanie środowiska jest dostarczana przez kamerę do zadania wykrywającego szachownicę. Wierzchołki pól szachownicy są punktami charakterystycznymi, na podstawie których estymowana jest pozycja szachownicy względem kamery. Do rozpoznawania szachownicy oraz estymacji pozycji zostały wykorzystane algorytmy z biblioteki OpenCV.

Wynikiem działania algorytmu estymacji pozycji szachownicy względem kamery jest macierz przekształcenia jednorodnego ${}^C T_G$, która jest następnie przekazywana ze struktury ramowej FraDIA do struktury ramowej MRROC++.



Rys. 4. Najważniejsze bloki serwomechanizmu PEC-EOL-PB-EIH

Uchyb zadaniowy wyrażony jest wzorem:

$$^{E}\varepsilon_{E,G} = {}^{E}T_{G} = {}^{E}T_{C}\,{}^{C}T_{G} \tag{1}$$

Na jego podstawie regulator proporcjonalny wylicza sterowanie, będące w istocie przyrostem położenia (ang. *Step Generator* – SG):

$$^{E}\Delta_{E,G'} = SG(^{E}\varepsilon_{E,G}) \tag{2}$$

Następnie obliczony przyrost podawany jest na wejście bloku, który sprawdza, czy po jego wykonaniu robot będzie nadal znajdował się w dozwolonym obszarze przestrzeni zadaniowej (ang. *Position Constraint* – PC):

$$^{E}\Delta_{E,G''} = PC(^{E}\Delta_{E,G'}, {}^{0}T_{E}) \tag{3}$$

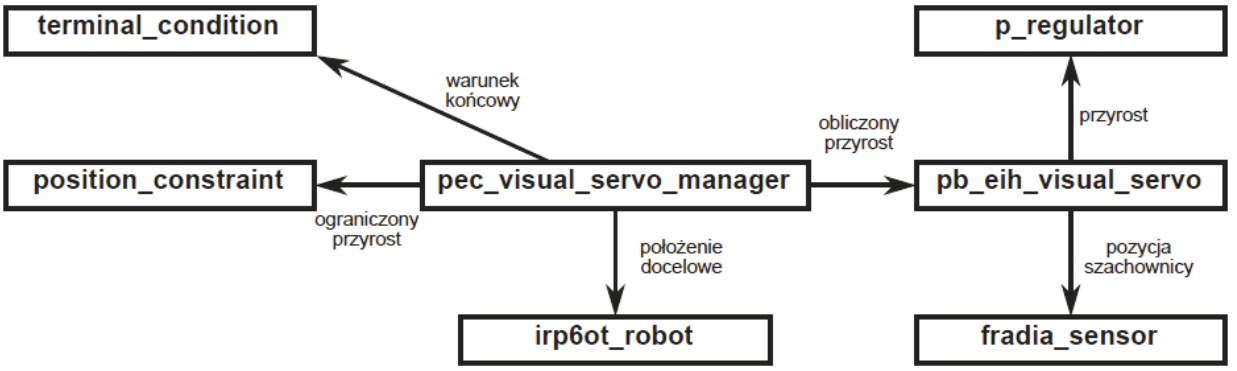
Otrzymany przyrost $^{E}\Delta_{E,G''}$ podawany jest następnie na blok, który ogranicza wartości prędkości i przyspieszenia (ang. *Velocity and Acceleration Constraint* – VAC):

$$^{E}\Delta_{E,G'''} = VAC(^{E}\Delta_{E,G''}) \tag{4}$$

Na podstawie tego przyrostu oraz aktualnego położenia obliczane jest zadane położenie końcówki roboczej, przekazane dalej do sterownika robota do realizacji:

$$^{0}T_{G'''} = {}^{0}T_{E}\,{}^{E}\Delta_{E,G'''} \tag{5}$$

Na rys. 5 przedstawiony jest diagram klas zaimplementowanego serwomechanizmu PEC-EOL-PB-EIH. W odróżnieniu od rys. 1 (na którym przedstawiono klasy bazowe całej struktury), na tym diagramie przedstawiono konkretne klasy wykorzystywane w zaimplementowanym serwomechanizmie.

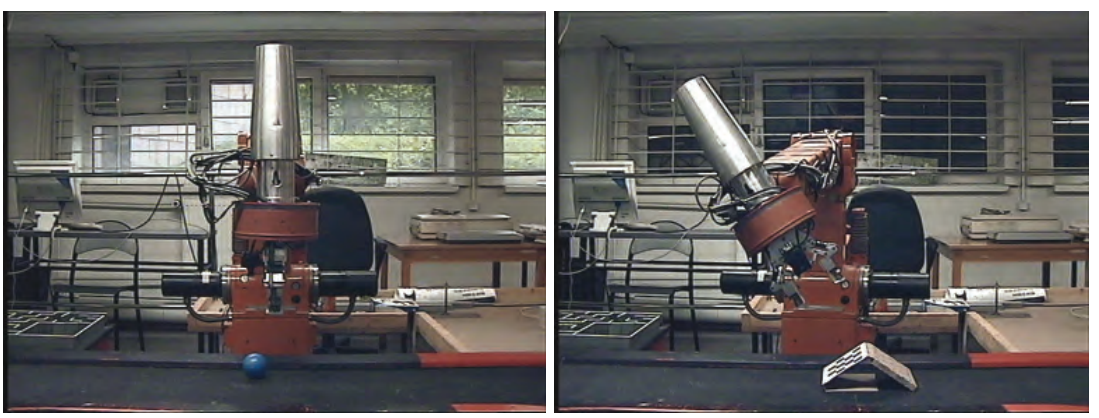


Rys. 5. Diagram klas użytych w implementacji serwomechanizmu PEC-EOL-PB-EIH

Klasa **fradia_sensor** jest implementacją interfejsu do wirtualnego sensora (dziedziczy z klasy **sensor**). Jest odpowiedzialna za odbieranie wielkości ${}^{C}T_{G}$ od wirtualnego sensora zaimplementowanego w strukturze ramowej FraDIA. ${}^{C}T_{G}$ jest wykorzystywane w obiekcie klasy **pb_eih_visual_servo** do obliczenia uchybu zadaniowego ${}^{E}\varepsilon_{E,G}$ (wzór 1 oraz blok „Obliczenie uchybu" na rys. 4). Klasa **pb_eih_visual_servo** pochodzi od klasy **visual_servo**.

Uchyb ${}^{E}\varepsilon_{E,G}$ jest przekazywany do generatora kroku – tutaj jest to regulator proporcjonalny. Obliczenie przyrostu (wzór 2) jest wykonywane w obiekcie klasy **p_reglator**, która pochodzi z klasy **visual_servo_regulator**. Wynikiem tej operacji jest przyrost pozycji ${}^{E}\Delta_{E,G'}$.

Ograniczenie pozycji końcówki roboczej jest realizowane w obiekcie klasy **position_constraint**, gdzie została zaimplementowana funkcja *PC* użyta we wzorze 3. Po ograniczeniu pozycji otrzymujemy wielkość ${}^{E}\Delta_{E,G''}$.



Rys. 6. Manipulator obserwujący dwa różne obiekty testowe

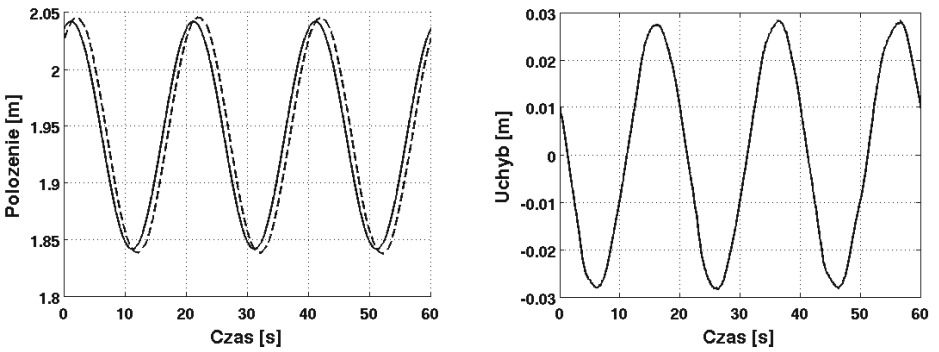
Ograniczenie szybkości i wartości przyspieszenia końcówki roboczej (wzór 4) jest zaimplementowane w generatorze ruchu serwomechanizmu wizyjnego: **pec_visual_servo_manager**. Po zastosowaniu ograniczenia na szybkość końcówki roboczej otrzymujemy ${}^{E}\Delta_{E,G'''}$. Uwzględniając aktualne położenie końcówki roboczej ${}^{0}T_{E}$, uzyskujemy zgodnie ze wzorem 5 nową pozycję końcówki roboczej ${}^{0}T_{G'''}$. Pozycja ta ma być osiągnięta w kolejnym kroku sterowania, więc jest wysyłana do sterownika manipulatora reprezentowanego przez obiekt klasy **irp6ot_robot**. Klasa ta odpowiada za komunikację z procesem EDP, który bezpośrednio steruje robotem.

Klasa **pec_visual_servo_manager** sprawdza również warunek zakończenia pracy generatora ruchu (**termination_condition**). Warunek jest spełniony, gdy uchyb będzie bliski zeru, szybkość szachownicy względem podstawy robota będzie bliska zeru oraz wartość przyspieszenia będzie bliska zeru.

6. EKSPERYMENTY

W celu weryfikacji działania struktury stworzono zadanie dwurobotowe (rys. 6). Pierwszy robot (taśmociąg) poruszał w zadany sposób przedmiotem (np. ruchem sinusoidalnym lub jednostajnie przyspieszonym). Drugi robot (manipulator) miał za zadanie śledzić przedmiot leżący na taśmociągu. Podczas pracy zbierane były pomiary pozycji końcówki roboczej manipulatora oraz pozycji taśmociągu. Na podstawie tych pomiarów obliczano rzeczywisty uchyb serwomechanizmu wizyjnego.

Na rys. 7 przedstawiono pomiary zebrane przy weryfikacji działania serwomechanizmu śledzącego szachownicę poruszającą się ruchem sinusoidalnym.

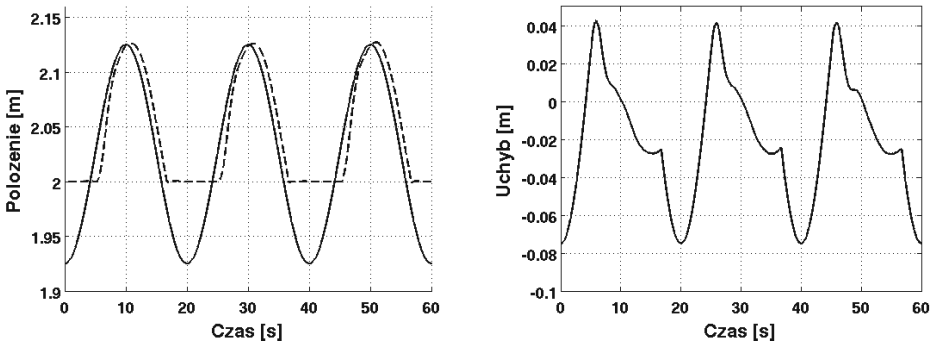


Rys. 7. Położenie przedmiotu (linia ciągła), położenie końcówki roboczej (linia przerywana) oraz uchyb dla serwomechanizmu PB–EIH śledzącego szachownicę

Na rys. 8 przedstawiono eksperyment, w którym szachownica poruszała się ruchem sinusoidalnym, lecz ograniczenia położenia końcówki roboczej nie pozwalały dokładnie naśladować ruch przedmiotu – współrzędna y położenia końcówki roboczej nie mogła być mniejsza niż 2 m.

7. PODSUMOWANIE

W artykule omówiono metodę integracji programowych struktur ramowych MRROC++ oraz FraDIA w sterownik robota mogący wykorzystać informację wizyjną. Na jego bazie zaimplementowano strukturę umożliwiającą tworzenie różnorodnych serwomechanizmów wizyjnych. W szczegółach omówiono implementację serwomechanizmu PEC–EOL–PB–SAC oraz przedstawiono eksperymenty przeprowadzone na rzeczywistym sprzęcie. Warte podkreślenia jest, iż w rzeczywistości serwomechanizm przetestowano dla różnych obiektów, a samą strukturę wykorzystano również do implementacji serwomechanizmów typu PEC–EOL–PB–EIH oraz PEC–EOL–IB–EIH.



Rys. 8. Położenie przedmiotu (linia ciągła), położenie końcówki roboczej (linia przerywana) oraz uchyb dla serwomechanizmu PB–EIH śledzącego szachownicę, z działającymi ograniczeniami położenia

Głównym celem dalszych prac jest rozbudowa istniejącej struktury o możliwość jednoczesnego korzystania z kilku kamer (np. jednej EIH oraz jednej SAC) oraz zbadanie możliwości przełączania między nimi. Przykładem zastosowania tego rozwiązania jest analiza sceny za pomocą nieruchomej kamery zawieszonej nad sufitem, a w przypadku wykrycia obiektu zainteresowania stopniowe zwiększanie roli kamery zintegrowanej z chwytakiem w miarę zbliżania się do tego obiektu. Dalsze prace obejmują również implementację różnego rodzaju regulatorów (PID, PI^2D).

BIBLIOGRAFIA

1. G.J. Agin. Real Time Control of a Robot with a Mobile Camera. Raport Techniczny 179, SRI International, 1979.
2. E. Cervera. A Cross-Platform Network-Ready Visual Servo Simulator. *Proceeding of the International Conference on Intelligent Robots and Systems*, s. 2314–2319, 2006.
3. S. A. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. IEEE Robotics and Automation Magazine, wol. 12(5), s. 651–670, 1996.
4. T. Kornuta. Application of the FraDIA vision framework for robotic purposes. Computer Vision and Graphics, Lecture Notes in Computer Science, wol. 6375, s. 65–72. Springer, 2010.
5. E. Marchand, F. Spindler, and F. Chaumette. ViSP for visual servoing: a generic software platform with a wide class of robot control skills. IEEE Robotics and Automation Magazine, wol. 12, s. 40–52, 2005.
6. F. Chaumette, S. A. Hutchinson. Visual Servoing and Visual Tracking. Rozdział w The Handbook of Robotics. B. Siciliano, O.Khatib (edytorzy), s. 563–610. Springer, 2008.
7. M. Staniak and C. Zielinski. Structures of visual servos. *Robotics and Autonomous Systems*, 58(8): 940–954, 2010.
8. C. Zielinski, W. Szynkiewicz, T. Winiarski, M. Staniak, W. Czajewski, T. Kornuta. Rubic's cube as a benchmark validating MRROC++ as an implementation tool for service robot control systems. *Industrial Robot: An International Journal*, wol. 34, nr 5, s. 368–375, 2007.
9. C. Zieliński, T. Winiarski. Motion Generation in the MRROC++ Robot Programming Framework. *The International Journal of Robotics Research*, wol. 29, nr 4, s. 386–413, 2010.