Układanie kostki Rubika jako zadanie testujące zdolności robota usługowego*

Cezary Zieliński[1], Wojciech Szynkiewicz[1], Tomasz Winiarski[1], Witold Czajewski[2], Maciej Staniak[1]

Streszczenie

Artykuł opisuje realizację zadania układania kostki Rubika, jako przykładu złożonej manipulacji wykonywanej przez dwuręcznego robota usługowego. Omówiono poszczególne fazy realizacji zadania, od etapu rozpoznania stanu i lokalizacji kostki z wykorzystaniem wizji poczynając, przez poszukiwania rozwiązania kostki z zastosowaniem algorytmów sztucznej inteligencji, na realizacji właściwego zadania manipulacji z wykorzystaniem serwomechanizmu wizyjnego i sterowania pozycyjno–siłowego kończąc. Zadanie manipulacji jest planowane w postaci ciągu prostych czynności (podstawowych umiejętności manipulacyjnych) bazujących na danych z różnych czujników. Przedstawiono także implementację układu sterowania z wykorzystaniem struktury programowej ramowej MRROC++.

1. WSTĘP

Roboty przemysłowe, które są stosowane w fabrykach, wykonują swe zadania polegając przede wszystkim na powtarzalności i precyzji ruchu, a więc wymagają bardzo uporządkowanego otoczenia. Na otoczenie robotów usługowych takiego wymagania nie można narzucić, ponieważ jest to środowisko dostosowane do człowieka. Dlatego roboty te będą musiały być wyposażone w niewspółmiernie bardziej złożone zmysły oraz będą musiały wykazywać o wiele większe zdolności do inteligentnych zachowań. Należy podkreślić, że stworzenie takich robotów będzie również miało zasadniczy wpływ na możliwości robotyzacji małych przedsiębiorstw produkcyjnych. Obecnie koszt urządzeń prepozycjonujących w gnieździe roboczym zazwyczaj przewyższa koszt samego robota. Przy robocie wyposażonym w odpowiednie czujniki urządzenia prepozycjonujące staną się zbędne. Stąd też zainteresowanie robotami usługowymi przejawiają również producenci robotów przemysłowych. Do tego należy dołożyć zainteresowanie przemysłu edukacyjnego oraz rozrywkowego tego typu urządzeniami. Biorąc pod uwagę pracę robotów w środowisku człowieka należy przyjąć, że urządzenia te będą musiały być dostosowane do otoczenia, które zostało ukształtowane na potrzeby ludzkie. Dlatego roboty tego typu muszą cechować się podobnymi umiejętnościami jak ludzie, czyli:

- zdolnością do manipulacji dwuręcznej różnorodnymi przedmiotami,

*Praca jest finansowana przez grant MNiI: 4T11A 003 25

[1]Instytut Automatyki i Informatyki Stosowanej, Wydział Elektroniki i Technik Informacyjnych, Politechnika Warszawska, ul. Nowowiejska 15/19, 00–665 Warszawa, C.Zielinski@ia.pw.edu.pl

[2]Wydział Elektryczny, Instytut Sterowania i Elektroniki Przemysłowej

- możliwością rozpoznawania, lokalizacji i chwytania obiektów na podstawie obrazu uzyskanego z kamery lub wielu kamer, zarówno ruchomych jak i nieruchomych,
- wykorzystaniem zmysłu dotyku (czucie sił) w manipulacji,
- zdolnością do komunikacji multimodalnej (porozumiewanie się z ludźmi za pomocą głosu i gestów),
- zdolnością do wnioskowania o środowisku i planowania swych działań, przy jednoczesnym zachowaniu możliwości do szybkiego reagowania na nagłe zmiany zachodzące w otoczeniu,
- umiejętnością bezkolizyjnego przemieszczania się w środowisku.

Każda z wymienionych cech stanowi obecnie pole intensywnych badań nad jej skuteczną realizacją. Niemniej jednak uzyskanie każdej z tych cech oddzielnie nie jest wystarczające. Robot usługowy musi dysponować wszystkimi tymi cechami jednocześnie, a więc zintegrowanie powyższych podsystemów w jeden sprawnie działający system jest dodatkowym trudnym wyzwaniem badawczym. Niniejsza praca koncentruje się na: manipulacji dwuręcznej, serwomechanizmach wizyjnych, sterowaniu siłowym, inteligencji maszynowej oraz integracji wszystkich powyższych podsystemów w spójną całość. Do tego celu została użyta programowa struktura ramowa MRROC++ [11–13]. W celu zademonstrowania współdziałania wszystkich podsystemów robota wybrano zadanie testowe polegające na ułożeniu kostki Rubika. Uchwycenie kostki wymaga jej lokalizacji, a następnie doprowadzenie chwytaka do miejsca chwytu – do tych czynności jest potrzebna jedna lub więcej kamer oraz serwomechanizm wizyjny. Do rozpoznania stanu początkowego kostki można użyć uproszczonych algorytmów rozpoznawania obrazów, gdyż uchwyconą kostkę można ustawić w polu widzenia kamery tak, aby widoczna była tylko jedna jej ścianka i aby wypełniała prawie cały obraz. Operację należy powtórzyć sześciokrotnie przechwytując kostkę naprzemiennie każdym z dwóch manipulatorów. Takie postępowanie minimalizuje szanse powstania błędu przy identyfikacji stanu początkowego, np. wskutek zmiennych warunków oświetleniowych. Jest to o tyle istotne, że taki błąd zniweczy poprawne ułożenie kostki. Znając stan początkowy, stosując algorytm IDA^* [8] poszukuje się rozwiązania, a więc sekwencji ruchu ściankami kostki. Następnie sekwencja ta jest przekładana na odpowiednie ruchy obu manipulatorów. Ich realizacja wymaga jednak reakcji na opory ruchu ścianek i ewentualne ich zacięcia. Do tego celu wykorzystywane jest sterowanie pozycyjno–siłowe. Jak widać zadanie to wymaga zarówno deliberacji (zaplanowania sekwencji ruchów) jak i szybkich reakcji na powstałe naprężenia. Niniejsza praca opisuje sposób wykonania powyższych podzadań oraz sposób integracji realizujących je modułów w spójną całość.

2. STRUKTURA SYSTEMU

Do manipulacji za pomocą dwóch rąk jest potrzebny robot dwuramienny, którego ramiona są zakończone chwytakami lub wielopalczastymi dłońmi. Stanowisko do badań nad manipulacją dwuręczną składa się z dwóch zmodyfikowanych manipulatorów IRp-6 (dodano na końcu łańcucha kinematycznego dodatkowy stopień swobody – tworząc nadgarstek o trzech stopniach swobody). Oba ramiona tak skonstruowanego

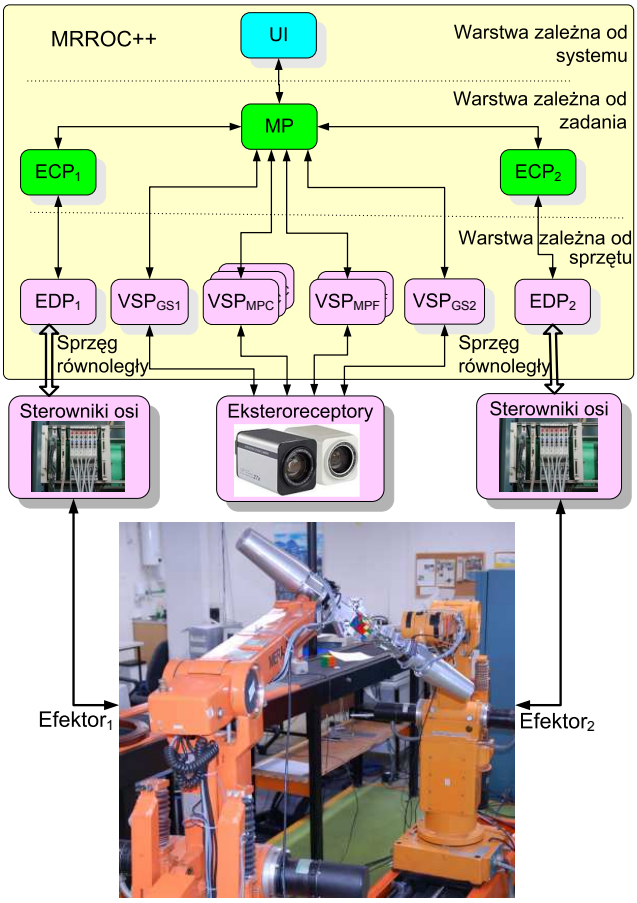
robota dwuramiennego mają po sześć stopni swobody. Dodatkowo jeden z robotów jest posadowiony na torze jezdnym, co umożliwia zmianę wzajemnego położenia obu ramion, a przez to dobór wspólnej przestrzeni roboczej obu ramion w zależności od wymogów zadania, np. rozmiarów obiektu, którym ramiona wspólnie manipulują. Każde z ramion jest zakończone chwytakiem, którego szczęki pozostają równoległe w całym zakresie otwierania/zamykania. W systemie eksperymentalnym zastosowano dwa typy czujników sił. Do pierwszego typu należą czujniki sił/momentów Gamma firmy ATI zbierające kompletną informację o siłach i momentach sił. Czujniki te są umieszczone w nadgarstkach manipulatorów pomiędzy chwytakiem a pozostałymi członami manipulatora. Czucie sił wykorzystano też bezpośrednio w samym chwytaku. W tym celu zastosowano czujniki nacisku. W każdej z dwóch szczęk chwytaka są umieszczone 4 rezystancyjne czujniki siły nacisku służące do wykrycia kontaktu z chwytanym obiektem. Ich rozmieszczenie dobrano tak, aby dwa czujniki (dla każdej szczęki) były aktywne w momencie, gdy chwytak uchwycił jedną warstwę kostki. Analogicznie, sygnalizacja kontaktu z czterech czujników informuje o uchwyceniu dwóch warstw kostki. Ponadto w chwytakach umieszczono miniaturowe kamery kolorowe.

Strukturę programową sterownika robota dwuręcznego stworzonego w środowisku MRROC++ przedstawiono na rys.1. Oprogramowanie składa się z wielu współbieżnie wykonujących się procesów (procesy są wielowątkowe), które działają pod kontrolą systemu operacyjnego czasu rzeczywistego QNX Neutrino. W rozważanym zadaniu, ze względu na duże wymagania obliczeniowe, poszczególne procesy sterownika wykonują się równolegle na kilku komputerach PC połączonych siecią Ethernet. Jest to warstwowa struktura hierarchiczna z podziałem funkcji pomiędzy poszczególne warstwy. Warstwa dolna realizuje obsługę sprzętu (efektorów i czujników) oraz wykonuje zlecenia warstw wyższych. Procesy Effector Driver Process (*EDP*) są właściwymi sterownikami robotów i realizują zlecenia przesyłane z procesów Effector Control Process (*ECP*). Procesy czujników wirtualnych Virtual Sensor Process (VSP) realizują obsługę czujników eksteroceptywnych. $VSP_{GSi}, i = 1,2$ obsługują czujniki nacisku zamontowane na szczękach chwytaków. Trzy procesy VSP_{MPC} obsługują kamery (dwie z nich są zamontowane w chwytakach, a jedna jest stacjonarna). Czujniki sił są obsługiwane w procesach VSP_{MPF}.

Warstwa środkowa wykonuje właściwe zadanie, czyli wszystkie algorytmy składające się na realizację zdania układania kostki. W implementacji sterownika dedykowanego realizacji zadania układania kostki Rubika procesy *ECP* są przezroczyste i jedynie przesyłają zlecenia z procesu koordynatora Master Process (MP). Warstwa górna jest interfejsem użytkownika.

3. WYKORZYSTANIE WZROKU

Serwomechanizmy wizyjne obliczające uchyb w przestrzeni zadaniowej (position based – PB) są wrażliwe na jakość kalibracji: parametrów kamery, układu kamera-robot oraz samego robota [5]. Część problemów z tym związanych usuwają serwomechanizmy obliczające uchyb w przestrzeni obrazu (image based – IB). Jednak trajektorie, które są generowane w przestrzeni obrazu mogą być niewykonywalne ze względ-



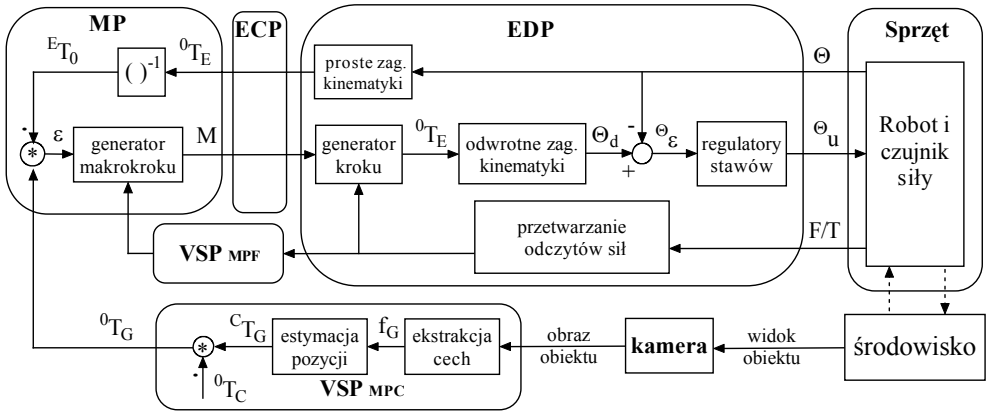
Rys. 1. Struktura sterownika robota dwuręcznego w środowisku MRROC++

du na nieliniową transformację pomiędzy przestrzenią obrazu a przestrzenią zadaniową [3]. Rozwiązaniem tego problemu może być użycie cech obrazu takich jak momenty [9], przejście do cylindrycznego układu współrzędnych [6] lub metoda polegająca na rozbudowaniu jakobianu obrazu. W tej pracy zastosowano rozwiązanie dwuetapowe. We wstępnej fazie chwytania obiektu, gdy jest on odległy od chwytaka, sterowanie odbywa się w przestrzeni zadaniowej i jest wykorzystywana kamera stacjonarna. W drugiej fazie, gdy chwytak znajduje się w pobliżu obiektu, wykorzystywana jest kamera wbudowana w chwytak, a sterowanie odbywa się w przestrzeni cech obrazu. Ponieważ trajektoria realizowana z wykorzystaniem algorytmu IB będzie krótka, nie wystąpią problemy ze zbieżnością algorytmu i utrzymywaniem obiektu w polu widzenia [4].

Dopóki końcówka nie wejdzie w kontakt ze środowiskiem, jej ruch sterowany jest tylko dzięki odczytom z kamer. Przetwarzaniem obrazu zajmuje się proces VSP_{MPC}, który najpierw oblicza cechy obrazu f_G, a następnie estymuje na ich podsta-

wie pozycję kostki w przestrzeni zadaniowej. Do celów lokalizacji kostki w obrazie założono, że jej ścianki składają się z kafelków o znanych *a'priori* kolorach. Piksele obrazu są klasyfikowane w zależności od przynależności do jednego z sześciu kolorów występujących na kostce. Klasyfikacja odbywa się w przestrzeni HSV. Następnie połączone piksele należące do tej samej klasy koloru są grupowane. Klasyfikacja oraz grupowanie są dokonywane algorytmem opisanym w [1]. Algorytm ten w oryginalnej postaci korzysta z przestrzeni kolorów YUV, w której wartości pikseli są bezpośrednio odczytywane z karty akwizycji obrazu. W naszym przypadku algorytm musiał ulec modyfikacji zarówno ze względu na to, że karta akwizycji obrazu dostarcza intensywności barw poszczególnych pikseli w przestrzeni RGB, jak i to, że klasyfikacja odbywa się w przestrzeni HSV. Przekształcenie RGB do HSV odbywa się za pomocą tablicy przekodowującej (*look-up table – LUT*).

Regiony wyodrębnione w obrazie są klasyfikowane ze względu na ich wielkość, krągłość, liczbę krawędzi i liczbę sąsiadów. Czworokąty o dużej krągłości (często przypominające bardziej romby niż równoległoboki) o minimum trzech sąsiadach uznawane są za kafelki. Regiony te są następnie grupowane algorytmem będącym połączeniem algorytmu K-mean oraz heurystycznej analizy położenia kafelków, by znaleźć przynależność kafelków do poszczególnych ścianek. Na „największej" ściance są wyodrębniane cechy obrazu f_G chwytanego obiektu G, które są położeniem środków czterech narożnych kafelków tej ścianki w przestrzeni obrazu. Ponieważ te cztery punkty leżą na jednej płaszczyźnie (na ściance kostki), na tej podstawie można obliczyć jej pozycję względem układu kamery C. Dzięki znajomości pozycji kamery względem globalnego układu odniesienia 0 (^0T_C) można wyznaczyć pozycję kostki względem globalnego układu odniesienia – ^0T_G. W ten sposób uzyskuje się wstępną lokalizację kostki. W przypadku, gdy wyodrębnienie czterech narożnych kafelków „największej" ścianki jest niemożliwe, estymacja położenia kostki jest dokonywana na podstawie największego możliwego do wykrycia kwadratu składającego się z kafelków. Struktura serwomechanizmu wizyjnego jest przedstawiona na rys. 2. W fazie



Rys. 2. Struktura układu sterowania: serwomechanizm wizyjny i sterowanie pozycyjno-siłowe

chwytania odległych obiektów proces *MP* odbiera pozycję ^0T_G od procesu VSP_{MPC} oraz pozycję ^0T_E (pozycję końcówki względem globalnego układu odniesienia 0) od

procesu *EDP* i na tej podstawie oblicza uchyb ε ($^{E}T_{G}$). Uchyb ten jest wykorzystywany do generacji trajektorii ruchu *M* prowadzącej do obiektu. Trajektoria *M* składa się ciągu makrokroków, czyli sekwencji pozycji $^{0}T_{E'}$, jakie końcówka manipulatora ma osiągnąć. Pozycje $^{0}T_{E'}$ są przekazywane przez proces *ECP* kolejno do procesu *EDP* do realizacji (*ECP* jest przezroczyste w tym zadaniu, gdyż cała trajektoria jest generowana w *MP* z powodu ścisłej współpracy efektorów). W *EDP* makrokrok rozbijany jest na kroki – każdy realizowany w ciągu 2ms. Co krok jest rozwiązywane odwrotne zagadnienie kinematyki, w wyniku czego jest otrzymywane położenie zadane w przestrzeni konfiguracyjnej Θ_d (pożądane położenia kątowe wałów silników), a następnie jest wyznaczane sterowanie $^{\Theta}u$ (wypełnienie PWM). Przetwarzanie obrazu (dzięki któremu zostanie obliczona pozycja zadana dla następnego makroku) odbywa się równolegle z ruchem końcówki, co zapewnia płynność ruchu.

Wstępna lokalizacja kostki obserwowanej przez nieruchomą, zewnętrzną kamerę pozwala na wykonanie ruchu chwytaka w pobliże kostki z precyzją od kilku do kilkunastu milimetrów (w zależności m.in. od jakości obrazu oraz odległości i kąta obserwacji kostki). Późniejsze, dokładne ustawienie chwytaka względem kostki następuje z pomocą kamery umieszczonej w samym chwytaku. W związku z tym, na tym etapie ruchu manipulatora odległość kamera–kostka jest relatywnie niewielka, a obraz obserwowanej kostki jest na tyle duży i wyraźny, że jest możliwa dokładna ocena wzajemnego położenia kostki i chwytaka. Położenie i orientacja chwytaka są na bieżąco aktualizowane i w każdej iteracji chwytak zbliża się do kostki zachowując odpowiednią orientację (prostopadle do ścianki i tak, aby „palce" chwytaka znalazły się wzdłuż krawędzi kostki). Błędy estymacji położenia i orientacji kostki nie przekraczają pojedynczych milimetrów i stopni, co jest w zupełności wystarczające do naprowadzenia chwytaka w taki sposób, by mógł chwycić kostkę. W ostatniej fazie ruchu, przed samym chwyceniem kostki jej obraz jest tak duży, że lokalizacja następuje tylko i wyłącznie na podstawie kształtu środkowego kafelka obserwowanej ścianki – ten pojedynczy kafelek wypełnia niemal cały obraz z kamery. Należy zwrócić uwagę, iż w tej fazie rozpoznawania obrazów zniekształcenia wprowadzane przez optykę minikamery o krótkiej ogniskowej są bardzo duże (silna „beczkowatość" obrazu), jednak są one na bieżąco kompensowane na podstawie wcześniej sporządzonego modelu.

Po uchwyceniu kostki następuje etap jej identyfikacji. Poszczególne ścianki kostki są prezentowane z odpowiedniej odległości kamerze stacjonarnej. Roboty przechwytują kostkę prezentując kolejne ścianki kamerze zgodnie z wcześniej ustalonym algorytmem. Podczas przechwytywania kostki nadal jest wykorzystywana analiza obrazów. Pozwala ona na kompensację błędów kalibracji dwóch manipulatorów i dokładne poprowadzenie chwytaka tak, aby precyzja chwytu była możliwie jak największa. Ścianka kostki ustawiana jest prostopadle do kamery i wypełnia niemal cały obraz. Możliwe i konieczne jest także takie ułożenie kostki, by wyeliminować ewentualne refleksy utrudniające jej identyfikację oraz zapewnić mniej więcej stałe i równomierne oświetlenie. W związku z tym, że lokalizacja kostki względem kamery jest teraz znana, identyfikacja poszczególnych kafelków kostki i rozpoznanie jej stanu początkowego jest znacznie prostsze, niż gdyby kostka mogła przyjmować dowolne orientacje. Kolory poszczególnych kafelków są identyfikowane w przestrzeni HSV, która pozwala skutecznie rozróżnić wszystkie kolory kostki poza czerwonym i poma-

rańczowym. Te dwa ostatnie różnią się w przestrzeni HSV jedynie jasnością, co implikuje zastosowanie niezmiennego oświetlenia podczas całej procedury identyfikacji stanu kostki (kafelki o niższej jasności uznawane są za czerwone, natomiast kafelki o wyższej jasności za pomarańczowe).

4. GENERACJA RUCHÓW ŚCIANEK KOSTKI

Po rozpoznaniu stanu ścian kostki jest wyznaczany ciąg obrotów warstw prowadzący do jej ułożenia. Standardowa kostka Rubika $3 \times 3 \times 3$ jest zbudowana z 27 małych kostek (sześcianów), zaś jej przestrzeń stanu składa się z około 4.3×10^{19} stanów (możliwych ułożeń kostki). Ta liczba stanów może być osiągnięta z każdego dowolnego stanu. Człowiek, zaczynając od losowo wybranego stanu kostki, potrafi ją ułożyć wykonując zazwyczaj od 40 do 100 ruchów.[†]

Jeśli przyjmie się obrót dowolnej warstwy kostki o kąt φ równy 90°, 180° lub -90° jako ruch elementarny, to stosując jeden z tych obrotów do dowolnej z sześciu ścianek można uzyskać jeden z 18 stanów. Wykonując kolejne ruchy otrzymuje się drzewo stanów, w którym korzeniem jest początkowy stan zadany, a współczynnik rozgałęzienia jest równy 18. Jednak, wykorzystując pewne własności, można wyeliminować części sekwencji ruchów prowadzących do zdublowania stanów w drzewie, np. przez usunięcie dwóch kolejnych obrotów tej samej warstwy o 90°, gdyż ten sam efekt otrzyma się wykonując pojedynczy obrót o 180°. W rezultacie, dzięki wykorzystaniu tych własności, średnia wartość współczynnika rozgałęzienia drzewa stanów wynosi $13,34847$ [7]. Niestety nie da się uniknąć powtórzeń stanów i dlatego liczba stanów w drzewie dla poziomu 18 wynosi $2,46 \times 10^{20}$, czyli jest prawie 6-krotnie większa od liczby wszystkich możliwych kombinacji kostki. Z powyższych rozważań wynika iż, dla znalezienia optymalnego (lub nawet suboptymalnego) rozwiązania tego zadania jest konieczne zastosowanie efektywnego algorytmu przeszukiwania drzewa stanów. W tym celu wykorzystano algorytm Iterative Deepening A* (IDA*) z funkcją heurystyczną wykorzystującą obliczoną wcześniej bazę wzorców [7]. Algorytm IDA* jest modyfikacją algorytmu A* i bazuje na ciągu iteracji wykorzystujących przeszukiwanie wgłąb [8]. W każdej iteracji przeszukiwana jest przestrzeń stanów tylko do wybranego poziomu. Z każdą kolejną iteracją zwiększa się numer poziomu. Zastosowanie funkcji heurystycznej $h(n)$ (będącej optymistycznym oszacowaniem odległości węzła n od węzła docelowego) polega na obcinaniu rozgałęzień pochodzących od węzła, dla którego szacowana długość ścieżki do rozwiązania przekracza odległość tego węzła do poziomu, na którym kończy się przeszukiwanie w danej iteracji. Innymi słowy, są rozwijane tylko te węzły, dla których wartość funkcji heurystycznej $h(n)$ spełnia nierówność $h(n) \leqslant (i - p)$, gdzie i jest numerem iteracji, a p poziomem, na którym znajduje się dany węzeł.

Aktualnie zaimplementowana wersja algorytmu pozwala na znalezienie rozwiązania optymalnego w czasie od kilku do kilkunastu sekund (dla obliczeń wykonywanych na komputerze typu PC z procesorem 2 GHz i pojemnością pamięci operacyjnej 512 MB) dla przypadków wymagających co najwyżej 15 ruchów. Jeśli rozwią-

[†]Szacuje się, aczkolwiek nie podano jeszcze formalnego dowodu, iż każdą kostkę można rozwiązać w co najwyżej 18 ruchów [7].

zanie wymaga większej liczby ruchów niż 15, wówczas poszukiwanie rozwiązania optymalnego jest zbyt czasochłonne i jest wyznaczane rozwiązanie suboptymalne, które zazwyczaj nie przekracza 20–21 ruchów.

5. GENERACJA RUCHÓW DWURĘCZNYCH

Planowanie i realizacja zadania układania kostki polega na wyznaczeniu trajektorii ruchu i chwytów dla obu chwytaków w celu wykonania wyznaczonej sekwencji obrotów ścian kostki. Po uzyskaniu z modułu przetwarzania obrazu zgrubnych danych o lokalizacji kostki w globalnym układzie współrzędnych, jest obliczana bezpieczna (bezkolizyjna) ścieżka dojścia do obiektu, przy założeniu znajomości przeszkód. Ścieżka jest opisywana za pomocą funkcji sklejanych (krzywe B-sklejane trzeciego lub krzywe NURBS) w przestrzeni zadaniowej, z takim usytuowaniem punktów kontrolnych, aby zapewnić bezpieczne dojście do pozycji umożliwiającej uchwycenie kostki. Następnie jest planowany nominalny chwyt. W rozważanym przypadku, ze względu na konstrukcję chwytaków (chwytaki dwupalczaste o odpowiednio ukształtowanych szczękach równoległych) oraz znajomość modelu obiektu (zakłada się, że kształt i rozmiary kostki są znane), problem planowania chwytu sprowadza się do wyboru chwytni – miejsca uchwycenia kostki. W układzie współrzędnych związanym z kostką pozycje chwytni są znane i stałe (są to narożniki kostki). Zatem planowanie chwytu nominalnego polega na podaniu pozycji układu współrzędnych związanego z chwytakiem względem układu związanego z kostką. Dla każdej z 6 ścian jest potencjalnie możliwych 8 konfiguracji chwytów (określonych przez pozycję układu chwytaka w układzie kostki).

W aktualnej wersji algorytmu zakłada się, że tylko jeden z chwytaków obraca ściankę, drugi wykonuje ruchy kompensacyjne zapobiegające zakleszczeniom kostki. Zadana trajektoria ruchu chwytaka jest opisana w układzie związanych z kostką jako obrót wokół jednej z osi tego układu o zadany kąt φ. Naturalną, w takim przypadku, parametryzacją orientacji jest reprezentacja oś-kąt. Istotnym problemem jest wybór takiej konfiguracji obu ramion, dla której tworzą one wraz z uchwyconą kostką zamknięty łańcuch kinematyczny, aby możliwe było wykonanie całego obrotu o zadany kąt bez konieczności zmiany uchwytu (przechwytywania). Przejście do wykonania obrotu kolejną ścianką wymaga zmiany chwytu. To z kolei wymaga zaplanowania bezkolizyjnej trajektorii przechwytywania. Przy jej planowaniu wykorzystuje się znajomość modelu geometrycznego chwytaka.

Ze względu na nieuniknione błędy wynikające m.in. z niedokładnej kalibracji, zgrubnej znajomości pozycji kostki, itp., przy realizacji kolejnych faz manipulacji niezbędna jest modyfikacja nominalnych trajektorii ruchu i chwytów na podstawie bieżących odczytów z czujników. W trakcie wykonania trajektorii dojścia realizuje to serwomechanizm wizyjny. Podczas chwytania i w trakcie obracania ścianki jest stosowane sterowanie pozycyjno-siłowe, w którym są wykorzystywane dane zarówno z czujników sił w nadgarstkach, jak i czujników nacisku w szczękach chwytaków.

6. WYKORZYSTANIE CZUCIA SIŁY

Czujnik sił i momentów umieszczony w nadgarstku manipulatora wykonuje pomiary i przetwarza informację o siłach, zgodnie z algorytmem omówionym w [10], poniżej opisano w jakich podzadaniach i do czego ta informacja jest używana. W pierwszej fazie zadania serwomechanizm wizyjny w wyniku analizy obrazu pochodzącego z kamer generuje trajektorię ruchu manipulatora mającą prowadzić do przejęcia kostki od operatora. Niestety z różnych przyczyn trajektoria ta odbiega od faktycznej, koniecznej do uchwycenia kostki. I tutaj czujnik sił, za pośrednictwem procesu VSP_{MPF} (rys. 2), pozwala na wykrycie momentu kontaktu z kostką i w szczególności zatrzymanie manipulatora w odpowiednim miejscu.

Uchwycenie kostki przez dwa manipulatory pozwala na rozpoczęcie właściwej manipulacji. Także i w tym podzadaniu ograniczona dokładność modeli kinematycznych manipulatorów i niepełna informacja o środowisku wymusza zastosowanie pozycyjno-siłowych algorytmów sterowania. Jako sposób definiowania zadania sterowania pozycyjno-siłowego wybrano formalizm TFF (*Task Frame Formalizm*) [2]. Zadanie to jest definiowane w procesie MP jako makrokrok do realizacji w procesie EDP. Dlatego informacja z czujnika siły jest dostarczana w pierwszej kolejności do procesu EDP. Przykładowo, podczas wzajemnego obrotu ścian kostki jeden z manipulatorów ma zadaną podatność we wszystkich kierunkach liniowych, a także dwóch spośród trzech kierunków obrotowych (działania momentów sił), z wyjątkiem osi wykonywanego obrotu. Jego zadaniem jest przytrzymanie kostki i eliminowanie nadmiernych, niepożądanych naprężeń, tym samym pełni rolę urządzenia podrzędnego. Mając takie wsparcie, drugi manipulator może bezpiecznie wykonać właściwy obrót w trybie pozycyjnym.

7. PODSUMOWANIE

W artykule omówiono realizację zadania układania kostki Rubika jako przykładu złożonej manipulacji wykonywanej przez dwuręcznego robota usługowego. Jako fizyczny model takiego robota wykorzystano dwa współpracujące manipulatory IRp-6 z dodanymi 6-tymi stopniami swobody i chwytakami oraz wieloma czujnikami. Przedstawiono wieloprocesową strukturę układu sterowania skonstruowanego na bazie programowej struktury ramowej MRROC++. Opisano algorytmy rozpoznawania obrazów, generacji ruchów, serwomechanizmów wizyjnych oraz pozycyjno-siłowych niezbędnych do realizacji zadania układania kostki Rubika. Eksperymenty wykazały, że wstępna wersja układu sterującego, wykorzystująca stacjonarną kamerę oraz czujniki sił, jest w stanie efektywnie pobrać z ręki człowieka kostkę Rubika, zidentyfikować jej stan początkowy oraz sprawnie obracać jej ściany bez zakleszczania.

LITERATURA

[1] J. Bruce, T. Balch, M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In: IEEE/RSJ Int. Conference on Intelligent Robots and Systems. *Proceedings*, 2000. Vol. 3, s. 2061–2066.

[2] H. Bruyninckx, J. De Schutter. Specification of force-controlled actions in the "task frame formalism": A synthesis. IEEE Trans. on Robotics and Automation, Aug, 1996, Vol. 12, No. 4, s. 581–589.
[3] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In: Workshop on Vision and Control. *Proceedings*, Block Island, USA, June, 1997.
[4] G. Chesi et al. A tutorial on visual servo control. IEEE Trans. on Robotics, Oct, 2004, Vol. 20, No. 5, s. 908–914.
[5] S. A. Hutchinson, G. D. Hager, P. I. Corke. A tutorial on visual servo control. IEEE Trans. on Robotics and Automation, Oct, 1996, Vol. 12, No. 5, s. 651–670.
[6] M. Iwatsuki, N. Okijama. A new formulation of visual servoing based on cylindrical coordinate system. IEEE Trans. on Robotics, Apr, 2005, Vol. 21, No. 2, s. 266–273.
[7] R. Korf. Finding optimal solutions to Rubik's cube using pattern databases. In: National Conf. on Artificial Intelligence (AAAI-97). *Proceedings*, July, 1997, s. 700–705.
[8] S. Russell, P. Norvig. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, N.J., Prentice Hall 1995.
[9] O. Tahri, F. Chaumette. Point-based and region-based image moment for visual servoing of planar objects. IEEE Trans. on Robotics, Dec, 2005, Vol. 21, No. 6, s. 1116–1127.
[10] T. Winiarski, C. Zieliński. Implementation of position–force control in MRROC++. In: 5th Int. Workshop on Robot Motion and Control. *Proceedings*, Dymaczewo, Poland, June, 23–25, 2005, s. 259–264.
[11] C. Zieliński. The MRROC++ system. In: 1st Workshop on Robot Motion and Control. *Proceedings*, Kiekrz, Poland, June 28–29, 1999, s. 147–152.
[12] C. Zieliński, A. Rydzewski, W. Szynkiewicz. Multi-robot system controllers. In: 5th Int. Symposium on Methods and Models in Automation and Robotics. *Proceedings*, Międzyzdroje, Poland, Aug 25–29, 1998. Vol. 3, s. 795–800.
[13] C. Zieliński et al. Mechatronic design of open-structure multi-robot controllers. Mechatronics, Nov, 2001, Vol. 11, No. 8, s. 987–1000.

RUBIK'S CUBE PUZZLE AS A BENCHMARK TASK FOR SERVICE ROBOTS

The paper presents Rubik's cube manipulation as an example of highly demanding benchmark task for two-handed service robot. Vision is used both for acquiring the cube (servoing) and identifying its initial configuration. During two-handed manipulation force/torque measurements are used to prevent jamming of the cube. Rubik's cube solver, nominal trajectory generation, visual servoing, and position–force control have been successfully integrated by the MRROC++ robot programming framework.