Robot jako system komputerowy

Materiał wykładowy
opracowany przez dr. hab. inż. Marka Gawrysiaka
profesora na Wydziale Mechanicznym Politechniki Białostockiej

Rozdział 1

Kiedy maszyna staje się robotem?

- Robot – pojęcie niejasne
- Historia urządzeń robotycznych
- Kiedy maszyna staje się robotem?
- Poziomy automatyzacji, rodzaje robotów
- Robot – automatycznie działająca maszyna wytwórcza
- Przykład zastosowania robota
- Wycieczka w historię, definicję i konfiguracje robota

Rozdział 2

Ręka robota – manipulator

- Robot hipotetyczny, jego funkcje i elementy
- Ręka robota i przedstawienie jej położenia
- Przekształcenia układów współrzędnych ręki
- Geometria i kinematyka ręki
- Napęd nadążny manipulatora
- Zbiór funkcji manipulatora

Rozdział 3

Zmysły robota – sensoryka

- Sensoryka dotykowa, siłowa i wzrokowa
- Zbiór funkcji sensorycznych
- Procesy równoległe i ich sterowanie
- Podstawowe rozkazy synchronizacji
- Program sterujący
- Aktywacja zestawu funkcji
- Sterowanie torem ruchu (trajektorią)

Rozdział 4

Programowanie robotów

- Typy danych
- Powiązanie układów współrzędnych
- Operatory działań
- Struktura programu sterującego
- Makrofunkcje
- Struktury składniowe
- Program montażu robotycznego

Podsumowanie

Białystok 2006

Rozdział 1
Kiedy maszyna staje się robotem?

Robot – pojęcie niejasne
Historia urządzeń robotycznych
Kiedy maszyna staje się robotem?
Poziomy automatyzacji, rodzaje robotów
Robot – automatycznie działająca maszyna wytwórcza
Przykład zastosowania robota
Wycieczka w historię, definicję i konfiguracje robota

Robot – pojęcie niejasne?

Słowo „robot" w językach słowiańskich oznacza pracę w ogóle lub jej odmiany. Twórcą tego słowa jest czeski pisarza K. ČAPEK (1890-1938). W latach dwudziestych naszego stulecia napisał on sztukę pod tytułem RUR (Rozumu Uniwersalne Roboty). W sztuce tej ČAPEK stworzył automatyczne urządzenia o ludzkim wyglądzie i wyposażone w „ludzkie" uczucia. W rzeczywistości, historycznie rzecz biorąc, koncepcja „robota" pojawiła się znacznie później niż odpowiednie urządzenia (systemy), które miały odpowiadać tej nazwie.

Nie istnieje do tej pory jasna i powszechnie zaakceptowana definicja robotów. Jeżeli zapytamy grupę osób co znaczy słowo „robot", to większość z nich prawdopodobnie odpowie, że jest to urządzenie automatyczne podobne do człowieka. Niektórzy opiszą urządzenie, które można bardziej dokładnie zdefiniować jako manipulator lub ręka automatyczna. W *Słowniku Języka Polskiego* można znaleźć następującą definicję robota: „maszyna, urządzenie techniczne imitujące działanie (czasem nawet wygląd) człowieka, odznaczające się określonym poziomem automatyzacji".

Inne definicje usiłują podkreślić, mniej lub bardziej, różne istotne (kluczowe) cechy robota: zastępowanie człowieka, autonomiczność (samodzielność) pracy, programowalność i „inteligencję" (robot inteligentny), możliwość przemieszczania się (mobilność), zastosowanie przemysłowe (robot przemysłowy) lub w usługach (robot usługowy). Z definicji tych wynika, że linia oddzielająca robot od zautomatyzowanych maszyn nie jest zawsze łatwa do określenia. Ogólnie biorąc, im maszyna jest bardziej skomplikowana i zindywidualizowana, tym bardziej prawdopodobne jest jej zaklasyfikowanie do grupy urządzeń robotycznych.

Historia urządzeń robotycznych

Historyczny rozwój urządzeń i systemów robotycznych wynika z powiązania dwóch różnych dróg twórczości (kreatywności): 1) wczesnej automatyzacji i zegarmistrzostwa, oraz 2) unowocześniania (innowacji) urządzeń w przemyśle maszynowym. Do mniej więcej połowy XVIII wieku utalentowani rzemieślnicy-artyści budowali nie tylko zegary wodne i sprężynowe, ale także modele zwierząt i ludzi (androidy). Lista tych automatycznie wprawianych w ruch zwierząt, ludzi czy ptaków jest ogromna. Cel tej twórczości technicznej był, jak byśmy go dziś nazwali, rozrywkowy, a nie produkcyjny. Rzemieślnicy, zaangażowani w „produkcję" tego rodzaju automatów, byli jednak ogromnymi innowatorami i nagromadzili potężne umiejętności praktyczne.

Można przyjąć, że pierwszym reprezentantem urządzenia robotycznego był zegar wodny (klepsydra wodna). Wynaleziony prawdopodobnie około 250 p.n.e. był on w stanie automatycznie powtarzać swój cykl pracy (odwracanie klepsydry). Odśrodkowy regulator szybkości, wynaleziony przez WATTa pod koniec XVIII wieku razem z systemem automatycznie sterowanych zaworów, uczynił silnik parowy pierwszym urządzeniem automatycznym zdolnym do utrzymywania prawie stałej szybkości obrotowej koła zamachowego, niezależnie od zmian obciążenia. Podobnie silnik spalinowy, wynaleziony w XIX wieku, służy jako przykład innego, automatycznie pracującego urządzenia, realizującego w sposób powtarzalny ssanie, sprężanie i zapłon mieszanki paliwowej. Rewolucja przemysłowa przyczyniła się do powstania wielu maszyn, operujących automatycznie; najpierw w przemyśle włókienniczym a później w budowie obrabiarek i innych operacjach przemysłowych. Najbardziej błyskotliwym wynalazkiem

tego rodzaju było krosno tkackie JACQUARDa. W celu szybkiej produkcji tkanin o różnych wzorach wykorzystywało ono system sterowania za pomocą taśmy dziurkowanej. Maszyna ta, wprowadzona do przemysłu już w roku 1801, opierała się na idei, która daje się stosować do prawie wszystkich definicji robota, to znaczy maszyna była programowalna i pomyślana do wykonywania różnych wzorów.

Pod koniec XVIII wieku zbudowano tokarkę ze śrubą pociągową. Główną cechą tej maszyny jest mechanizm śrubowy napędzający wózek (suport) z zamocowanym nożem skrawającym. Mechanizm ten zazębia się (za pomocą przekładni) z wrzecionem tokarki. Przez zmianę przełożenia przekładni można praktycznie otrzymać dowolny skok gwintu. Oznacza to, innymi słowy, zmienny program przez sterowanie śrubą pociągową. Można to traktować jako zwiastun (prekursor) technik śledzenia stosowanych szeroko w tokarkach i frezarkach. Późniejsze narzędzia są w pewnym zakresie systemami robotycznymi. Kolejne udoskonalenia tych obrabiarek prowadzą do stworzenia tokarek automatycznych o naturze czysto mechanicznej (automatów tokarskich), do masowej produkcji takich części, jak szpilki, śruby, nakrętki i podkładki. Maszyny te były, i ciągle są, programowalne mechanicznie. Wzorzec, produkowany w danej chwili, można po kilku godzinach wymienić na inny. Wiele takich maszyn wyprodukowano po raz pierwszy w latach między rokiem 1920 a 1930.

Po II wojnie światowej, wprowadzono do produkcji tokarki i frezarki sterowane numerycznie (NC, Numerically Controlled). Maszyny te były bardziej elastyczne od automatów tokarskich z punktu widzenia zmienialności programu. Na tym poziomie udoskonaleń pozycjonowanie narzędzia względem materiału obrabianego można zrealizować przez programowanie przemieszczeń od punkt do punktu. Gdy skomputeryzowane maszyny sterowane numerycznie (CNC, Computerized Numerically Controlled) zastapiły maszyny NC, programowanie stało się bardziej wymyślne (złożone) – tory i trajektorie narzędzi zaczęto obliczać przez komputer maszyny. Na tym poziomie udoskonaleń operator musiał zdefiniować zarówno rodzaj trajektorii (np. linię prostą lub łuk) i właściwe parametry trajektorii (np. współrzędne punktów łączących linię prostą lub początek układu współrzędnych i promień łuku, itd.). Inne ulepszenia polegają na przykład na: (1) równoległym, ciągłym pomiarze części obrabianych, w celu ustalenia momentu, w którym narzędzie wymaga naostrzenia, wymiany lub obrócenia; (2) obliczaniu optymalnych warunków pracy, takich jak prędkość skrawania, posuw i głębokość skrawania; oraz (3) wymienianiu narzędzi dostarczanych w kolejnych etapach (sekwencji) obrabiania (procesowania).

Opisaliśmy rozwój tokarki jako reprezentanta świata maszyn przemysłowych operujących automatycznie. Podobnie można przedstawić rozwój maszyn włókienniczych lub drukarskich. Skomputeryzowany system drukowania, dostępny dziś powszechnie, całkowicie zmienił obraz tradycyjnego drukarstwa mechanicznego.

Kiedy maszyna staje się robotem?

W definicjach robotów znajdujemy wiele powtarzających się określeń (słów kluczowych). Należą do nich: manipulator (mechanizm), programowalność, możliwość ruchu w kilku kierunkach, wyposażenie w narzędzie robocze (końcówkę roboczą, efektor końcowy), zdolność wykonywania pracy wykonywanej przez człowieka w procesie przemysłowym (przemieszczanie materiałów, narzędzi lub urządzeń specjalistycznych), wbudowany układ sterowania, wielofunkcyjność (zmiennie programowalne ruchy w celu wykonania różnych zadań), samodzielność (autonomiczność) itp. Z tego zestawu określeń można wyciągnąć wniosek, że np. pralka automatyczna czy obrabiarka automatyczna wykazują cechy urządzeń robotycznych. Czy jest więc coś złego w definiowaniu pralki, czy obrabiarki automatycznej jako robota?

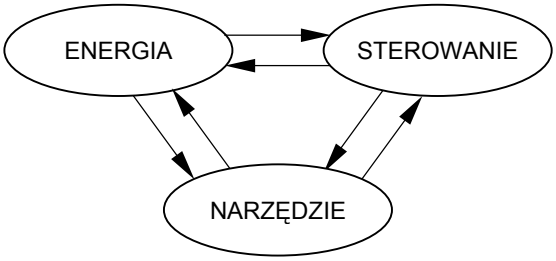
W rzeczywistości żyjemy w świecie produktów wytwarzanych na maszynach o przytoczonych wyżej charakterystycznych cechach robotów a różniących się jedynie poziomem złożoności. Produktami takimi na przykład, są: słoiki na dżemy czy produkty spożywcze; łożyska kulkowe i długopisy (pióra kulkowe); śruby, nakrętki, podkładki, gwoździe i nity; skarpety i buty; kostki (chipy) elektroniczne, oporniki, kondensatory i płytki drukowane; cukierki i lody. Listę tę można rozszerzyć, przez baterie i filmy fotograficzne, do wielu, wielu innych produktów. Produkty te łączy jedna wspólna cecha – są one całkowicie lub częściowo wytwarzane przez maszyny działające automatycznie.

Powstaje więc problem jak określić czy dana maszyna jest robotem, a jeżeli tak, to jaki jest to rodzaj robota. W tym celu musimy rozważyć pewne kryteria ogólne, bez których żaden układ (system) nie może istnieć. Aby rozważania były jasne, musimy sklasyfikować maszyny automatyczne według ich poziomu „intelektualnego" („umysłowego"). Taka klasyfikacja pomoże nam zrozumieć każdą koncepcję automatyzacji i umieścić ją we właściwym miejscu, w odniesieniu do innych koncepcji.

Poziomy automatyzacji, rodzaje robotów

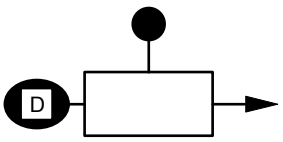
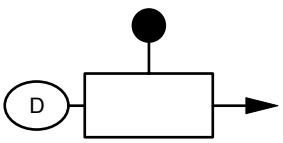
(Według: SANDLER B.Z.: *Robotics. Designing the mechanism for automated machinery*. Prentice-Hall Inc. 1991)

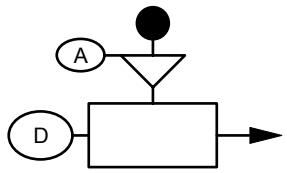
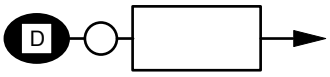
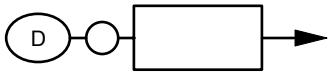
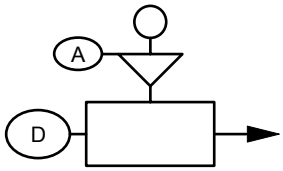
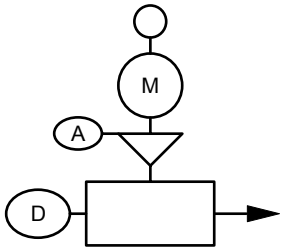
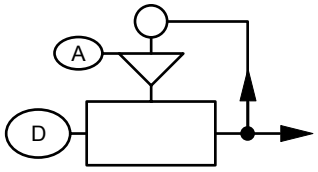
Każde narzędzie, używane przez człowieka, można opisać za pomocą schematu na rysunku 1. Źródło energii, jednostkę sterującą oraz samo narzędzie można wzajemnie powiązać w różny sposób. Te trzy składowe nie muszą być podobnej natury lub mieć podobny poziom złożoności. Za pomocą tego schematu (modelu) można przebadać każde urządzenie techniczne, określić czy należy ono do rodziny robotów, a jeżeli tak, to do jakiej gałęzi tej rodziny. Schemat ten może opisać każde narzędzie: młotek, szpadel, samolot, komputer, rakietę, pojazd księżycowy, czy maszynkę do golenia. Każdy z tych przykładów ma źródło energii, środki sterowania, oraz narzędzie do wykonywania wymaganych funkcji. Na tym etapie powinniśmy pamiętać, że nie istnieje żadna granica liczby elementów w żadnym systemie. Oznacza to, że system może składać się z wielu podobnych lub różnych źródeł energii, podobnych lub niepodobnych środków sterowania różnymi parametrami i, oczywiście, podobnych lub różnych narzędzi. Szczegóły tego rodzaju schematu określają czy dany system można zdefiniować jako robot czy nie. Popatrzmy teraz na tabelę 1. Pokazuje ona schematycznie różne możliwości, które tworzą 10 poziomów automatyzacji.

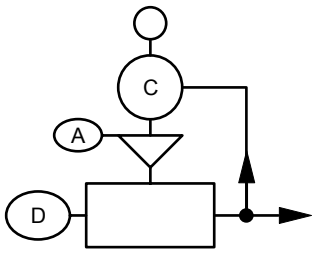
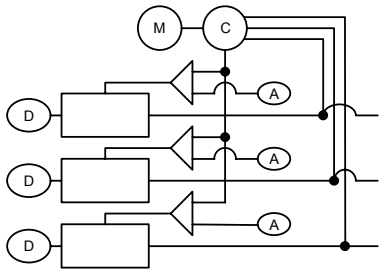


Rys. 1 Powiązanie narzędzia z energią i sterowaniem

Tab. 1 Poziomy automatyzacji

Legenda	***Uwaga:*** Zaczernione pola oznaczają działanie ręczne	
System; D Energia napędzająca; Wzmacniacz; Sterowanie; C Komputer; A Energia wzmacniająca; Narzędzie; Sprzężenie zwrotne; M Pamięć		
1		Źródłem energii jest osoba; środkami sterowania – jej ręce. Przykłady: młotek, łopata, szpadel, nóż, czy dłuto rzeźbiarza. Gdy osoba operuje młotkiem, tor młotka, moc i tempo uderzeń są sterowane przez nią samą, przez operatora. Sprzężeniem zwrotnym, czyli czujnikiem informującym operatora o rzeczywistym położeniu młotka, jest szybkość, a jego zgromadzona energia to energia mięśni ramienia, dłoni, pleców człowieka; udział w tym biorą także oczy. Oczywiście jest to słuszne również dla szpadla, dłuta i innych narzędzi ręcznych.
2		Źródłem energii jest silnik, ale środki sterowania są ciągle w ręku człowieka. Np. prosta tokarka, wiertarka, wiertarka dentystyczna, maszyna do szycia z silnikiem elektrycznym, golarka z napędem elektrycznym lub mechanicznym. Do pewnego zakresu ta grupa maszyn również zawiera maszyny napędzane siłami mięśni innej osoby (lub zwierzęcia), czy napędzane nogami tej samej osoby.

3		Źródłem energii jest silnik, środki sterowania pozostają ręczne, ale są sztucznie wzmacniane: Np. protezy sterowane „mięśniami" elektrycznymi czy układ wspomagania kierownicy w samochodzie podlegają temu przypadkowi w pewnym zakresie.
4		Źródłem energii jest osoba, ale funkcję sterowania pełni (szeregowo) system. Np. ręcznie napędzana maszynka do mięsa czy maszyna do pisania. Obracając korbą maszynki do mięsa operator dostarcza urządzeniu energii potrzebnej do transportowania mięsa do urządzenia tnącego (nożyk-sitko), posiekania mięsa i wyciśnięcia go przez otwory sitka. Prędkość dostarczania, czyli transportowania mięsa, jest koordynowana z tempem siekania przez skok ślimaka oraz wymiary i kształt maszynki. Podobnie, gdy naciskamy klawisz maszyny do pisania następuje sekwencja zdarzeń: taśma jest podnoszona, młoteczek z literą jest przyspieszany w kierunku papieru, a wózek, trzymający papier, przeskakuje o jeden krok. Ta sekwencja jest wbudowana w łańcuch kinematyczny urządzenia.
5		Źródłem energii jest silnik a sterowanie realizowane jest (szeregowo) przez kinematykę systemu. Np. automat tokarski, krosno automatyczne, maszyna do automatycznego naklejania etykiet na butelki, maszyny do napełniania i ważenia. Ta rodzina urządzeń należy do robotów typu „bang-bang". Systemy takie mogą być dość elastyczne. Na przykład automat tokarski można, przez zmianę wału krzywkowego, przekształcić z wytwarzania tulejek na podkładki. Na jednym i tym samym automacie można wytwarzaj najróżnorodniejsze części kołowosymetryczne. Tego typu maszyny produkują spinacze biurowe, agrafki, naboje, łożyska toczne, łańcuchy rolkowe i zębate.
6		Źródłem energii jest silnik a sterowanie odbywa się automatycznie, według sztywnego programu i jest wzmacniane. Np. system automatyczny sterowany przez sterowniki nadrzędne (master controllers), tj. przekaźniki elektryczne, pneumatyczne czy hydrauliczne. Systemy takie są elastyczne w ograniczonym zakresie.
7		Tak samo jak w 6, ale sterownik jest elastyczny lub programowalny. Np. automatyczne systemy śledzące (kopiujące). Przykładem może być obróbka łopatki turbiny. Kształt łopatki drewnianej (wzorcowej) jest kopiowany przez kopiał (lub czujnik), a przemieszczenia czujnika, gdy cały czas utrzymuje on łagodny styk z zarysem części drewnianej, są wzmacniane i przekształcane, przez sterowanie, w przemieszczenia głowicy frezarki. Innym przykładem jest programowalne krosno JACQUARDa czy obrabiarka sterowana numerycznie.
8		Tak samo jak w 4 i 7, ale z dodatkiem sprzężeń zwrotnych, tj. systemów wyjścia, blokowania, pomiaru i dostrajania. Przykładem może być frezarka z automatycznym dostrajaniem frezu, co wymaga ciągłego pomiaru wymiarów obrabianych i pomiaru przemieszczenia frezu. Dodatkowo frez musi być ostrzony, a grubość warstwy usuwanej z frezu może być brana w rachunek. Innym przykładem jest blokowanie krosna, gdy zerwie się nić osnowy lub wątku.

9		Tak samo jak w 8 z dodatkiem komputera i/albo pamięci. Np. maszyny automatyczne zdolne do obliczania warunków pracy, takich jak parametry skrawania, tory ruchu chwytaków czy noży. Do tej grupy maszyn należą więc te systemy, które są dają się uczyć (są nauczalne). Np. głowica malarska może być „przeprowadzona" ręcznie przez miejsca malowania; ruch ten zostanie następnie „zapamiętany" (czy nawet przeliczony i poprawiony) i po tym „nauczeniu się" malowanie będzie wykonywane całkowicie automatycznie, czasami szybciej niż podczas procesu nauczania.
10		Poziom ten różni się od 9 tym, że opiera się na komunikacji między maszynami i procesami, wykonującymi rozkazy sterownicze, aby sprowadzić cały system do harmonicznego działania. Przykładem może być linia automatyczna do produkcji tłoków silników spalinowych.

Robot – automatycznie działająca maszyna wytwórcza

Granice między przedstawionymi poziomami automatyzacji są bardzo nieostre. Weźmy na przykład maszynę, która jako całość należy do grupy 5. Dla wykonania jakiegoś zadania specjalistycznego można ją jednak wyposażyć w sprzężenie zwrotne sygnalizujące na przykład brak przedmiotów obrabianych i w układ powodujący po tym zatrzymanie maszyny, aby nie pracowała jałowo. Innym przykładem jest samochód. Jest on kierowany ręcznie, ale ma automatycznie działający silnik. Rozwiązanie sporu o definicję robota leży prawdopodobnie gdzieś między przypadkiem 5 i 7 powyższej klasyfikacji. Tak więc, zamiast mglistej koncepcji robota, bardziej użytecznym było by pojęcie automatycznie działających maszyn i systemów wytwórczych. To, co dostarcza działania w takich systemach, na wszystkich poziomach złożoności, może być natury czysto mechanicznej, elektromechanicznej, elektronicznej, pneumatycznej, hydraulicznej lub mieszanej.

Robocza (wykonawcza) część takiej maszyny jest zawsze mechaniczna. I nie zależy to od poziomu automatyzacji, to znaczy od tego czy mamy np. sterowany numerycznie czy skomputeryzowany elastyczny system wytwarzania. Innymi słowy, niezależnie od „inteligencji" sterowania, urządzenie wykonuje działanie mechaniczne. I tak rozwidlone haczyki maszyny dziewiarskiej wykonują specyficzny ruch mechaniczny w celu wytwarzania skarpetek; stoły krzyżowe realizują ruch mechaniczny zgodny z programem pozycjonowania podstawy układu elektronicznego w ten sposób, że elementy elektroniczne mogą być na tej podstawie montowane; frez frezarki porusza się wzdłuż zdefiniowanej trajektorii w celu wytworzenia części maszyny. Noże, chwytaki, palniki, przebijaki czy elektrody są narzędziami i cechą ich operowania jest ruch mechaniczny. Nawet jeżeli narzędziem jest promień świetlny, to jego źródło musi być przemieszczane względem części obrabianej.

W tym miejscu warto zastanowić się nad rolą mechaniki w urządzeniach automatycznych i robotycznych. Dziś automatyka i robotyka zwykle kojarzy się nie tyle ze zautomatyzowanymi mechanizmami, ile z elektroniką, techniki komputerową i programowaniem. Zapomina się przy tym często, że niezależnie od genialnej elektroniki, wymyślnych programów, elegancji języków komputerowych czy wygody wyświetlania na ekranie, wszystkie te elementy są ściśle powiązane i przeplecione mechaniką. To powiązanie dotyczy przynajmniej dwóch obszarów. Pierwszy polega na tym, że produkcja elektronicznych kostek, płytek i styków, tj. tzw. sprzętu (hardware'u), odbywa się za pomocą wysoko zautomatyzowanych środków mechanicznych (oczywiście w połączeniu z innymi technikami) z mechanicznych materiałów. Drugi obszar to problemy czysto mechaniczne, jakie pojawiają się w częściach i elementach tworzących komputer. Na przykład naprężenia cieplne, powodowane przez wydzielanie ciepła w elementach elektronicznych, powodują problemy czysto mechaniczne w konstrukcji obwodów; styki, łączące oddzielne bloki i płytki w jednostkę, podlegają zużyciu mechanicznemu i naciskom stykowym. Systemy przechowywania informacji, które mają często naturę czysto mechaniczną (napędy dyskietek, manipulatory wymieniające dyskietki) sprawiają problemy dynamiczne, kinematyczne i dokładnościowe. Innym przykładem są guziki (przyciski), w których występuje uderzanie styków, co z kolei prowadzi do pojawiania się fałszywych sygnałów, a więc do zmniejszenia jakości urządzenia. To krótkie i

dalekie od całkowitości wyliczenie problemów mechanicznych, jakie mogą się pojawić w „mózgach" robotów, ilustruje ważność mechanicznych aspektów automatyzacji i robotyzacji.

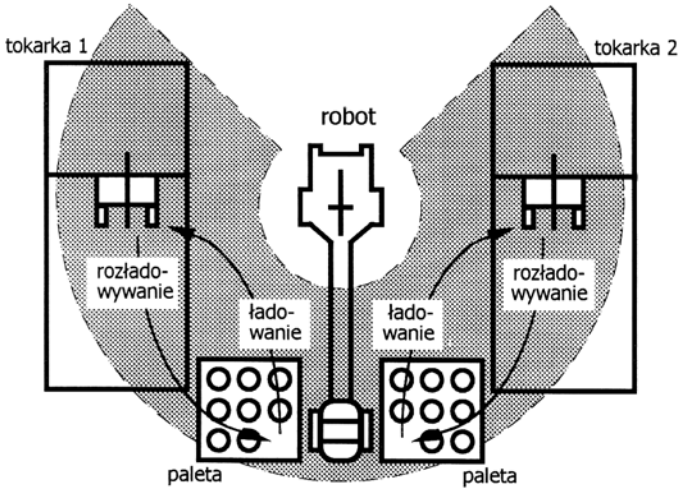
Konstruktor maszyn działających automatycznie będzie zawsze musiał rozwiązywać problemy mechaniczne dotyczące:

- rodzaju optymalnego rozwiązania koncepcyjnego w celu osiągnięcia konkretnego celu;
- rodzaj narzędzi lub organów, jakie mają być stworzone w celu operowania przedmiotem procesowanym (podlegającym manipulowaniu czy obróbce);
- środków ustalenia przemieszczeń mechanicznych, torów i ruchów narzędzi;
- sposobu dostarczenia wymaganej wielkości ruchu;
- środków zapewnienia wymaganej dokładności (jak nie przekroczyć dopuszczalnego odchylenia w ruchu narzędzi czy innych elementów).

Przykład zastosowania robota

Według: HOSHIZAKI J, BOPP E.: *Robot Application Design Manual*. Wiley 1990

Przykład na rysunku 2 ilustruje zdolności robota przemysłowego w prostym zastosowaniu „maszyna załadowana, maszyna niezaładowana". Przykład pokazuje jak robot czyta sygnały wejściowe i używa programu do decydowania gdzie poruszyć się i które wyjścia włączyć.



Rys. 2 Typowe zastosowanie robota

Robot podnosi z palety części (przedmioty procesowane) przeznaczone do obrabiania (procesowania) i ładuje je na tokarkę automatyczną. Kolejność tych działań, czyli sekwencja programu, wygląda następująco:

- Operator wybiera, jaki program robota należy uruchomić.
- Operator ładuje przedmioty (części) do palety.
- Operator naciska przycisk uruchamiający cykl pracy.
- Robot porusza się do pozycji „podnoszenie części". Czujniki (sensory) położenia ramion robota meldują, że robot znajduje się w odpowiedniej pozycji.
- Robot włącza elektryczny sygnał wyjściowy do zamknięcia chwytaka (efektora końcowego). Elektryczny sygnał wyjściowy idzie do zaworu pneumatycznego, który uruchamia cylinder pneumatyczny chwytaka w celu uchwycenia części.
- Robot, trzymając część w chwytaku, podnosi ją z palety.
- Robot upewnia się czy część jest w chwytaku – przez sprawdzenie czy przełącznik „część obecna" jest włączony. Jeżeli przełącznik obecności części jest wyłączony, to robot zatrzymuje ruch i sygnalizuje stan błędu. Przełącznik obecności części znajduje się wewnątrz chwytaka i włącza się wtedy, gdy część procesowana znajduje się w chwytaku.
- Robot sprawdza czy istnieje sygnał wejściowy „gotów do załadowania", pochodzący z obrabiarki. Jeżeli tak jest, to robot porusza się do obrabiarki i umieszcza część w uchwycie obrabiarki.

- Robot wysyła następnie elektryczny sygnał wyjściowy do obrabiarki, aby zamknęła uchwyt w celu zaciśnięcia części obrabianej.
- Obrabiarka, po zamknięciu uchwytu, przesyła robotowi sygnał wejściowy, że uchwyt jest zamknięty.
- Robot otwiera chwytak przez włączenie elektrycznego sygnału wyjściowego „chwytak otwarty". Sygnał wyjściowy włącza zawór pneumatyczny, który z kolei uruchamia cylinder pneumatyczny chwytaka w celu uwolnienia części.
- Robot porusza się od obrabiarki.
- Robot podaje sygnał wyjściowy do uruchomienia obrabiarki.
- Obrabiarka obrabia (procesuje) część (przedmiot procesowany).
- W celu wyjęcia przedmiotu z obrabiarki i umieszczenia go z powrotem na palecie sekwencja powtarzana jest w porządku odwrotnym.

Wycieczka w historię, definicję i konfiguracje robota

Według: STADLER W.: *Analytical Robotics and Mechatronics*, McGraw-Hill 1995

Wycieczka ta pokaże okres młodzieńczy robota i umieści go we szkielecie obecnej techniki.

W 1954 r. G. C. DEVOL opublikował patent na programowalną metodę przesyłania artykułów między różnymi częściami fabryki. W patencie napisał:

> Przedstawiany wynalazek umożliwia po raz pierwszy skonstruowanie maszyny mniej lub bardziej wieloczynnościowej (general purpose machine), która ma uniwersalne zastosowanie w ogromnej różnorodności zastosowań, gdy potrzebne jest sterowanie cykliczne.

W 1956 r. DEVOL spotkał J. F. ENGELBERGERa, młodego inżyniera w przemyśle lotniczym. Razem z innymi założyli oni pierwszą na świecie fabrykę robotów, Unimation Inc., i zbudowali swoją pierwszą maszynę w roku 1958. Ich inicjatywa znacznie wyprzedzała ten czas; Unimation, według ENGELBERGERa, nie widział zysku do 1975 r.

Pierwszy robot przemysłowy zaczął pracować w 1961 r. w fabryce samochodów prowadzonej przez General Motors w Trenton, New Jersey. Robot podnosił gorące części metalowe z formy odlewniczej i układał je w stos.

Japonia, dla porównania, zaimportowała swój pierwszy robot z AMF (American Machine and Foundry) w 1967, w którym to czasie Stany Zjednoczone były od dobrych 10 lat na czele w technice robotycznej. O ogromnym wysiłku, włożonym przez przemysł japoński w ten obszar, najlepiej świadczy fakt, że firma Unimation w końcu została zredukowana do przekazania swej pionierskiej techniki robotycznej do firmy Kawasaki Heavy Industries w postaci licencji w 1968 r. Jeden z byłych wyższych zarządzających firmą Unimation stwierdza:

> Wątpię czy opłaty należne licencjodawcom z tytułu sprzedaży licencji stanowiły więcej niż dziesiątą część kosztów, jakie firma Kawasaki musiałaby wydać na samodzielne opracowanie takiej samej techniki.

W 1990 r. istniało więcej niż 40 fabryk japońskich, włączając giganty jak Hitachi i Mitsubishi, które produkowały roboty na sprzedaż. Dla porównania istniało około tuzina firm w Stanach Zjednoczonych, z Cincinnati Milacron i Westinghouse's Unimation na czele. W 1979 amerykański lider, Unimation, był jedyną firmą na świecie, która aktywnie sprzedawała zaawansowany robot montażowy. W 1982 r. General Motors, największy pojedynczy użytkownik robotów na świecie, podpisał umowę z Fanuc Ltd, której celem było wspólne produkowanie i sprzedawanie robotów w Stanach Zjednoczonych. W pierwszych sześciu miesiącach działania więcej niż połowa ze 100 sprzedanych robotów powędrowała do General Motors zamykając inne firmy amerykańskie przed największym pojedynczym kupcem na rynku.

Pojawiły się inne wspólne działania: Bendix zobowiązał się do marketingu robotów Yaskawa, podczas gdy Yaskawa została już sprzedawana przez Hobart Industries i Nordson Corporation; IBM zobowiązał się z Seiki do marketingu robotów montażowych SCARA, które sprzedaje za połowę ceny jaką uzyskuje Unimation za zaawansowany model PUMA, ale ma za to 85% tego, co uzyskuje PUMA. W 1987 r. rynek Stanów Zjednoczonych miał wartość ponad 170 miliardów $.

Żadne wprowadzenie do robotyki nie byłoby kompletne bez sięgnięcia do pochodzenia słowa *robot*. Pierwszy raz słowa tego użył Karel ČAPEK w sztuce RUR (Rozumu Uniwersalne Roboty), napisanej w 1920 r. W czeskim słowo *robot* znaczy *robotnik*. W sztuce Rozum i jego syn odkrywają wzór chemicz-

ny na sztuczną protoplazmę, prawdziwą podstawę życia, i postanawiają zrobić robot. Po dwudziestu latach patrzą na swoją konstrukcję i młody Rozum postanawia usunąć organy, które uważa za zbędne u idealnego robotnika. Rozum mówi:

> Człowiek jest czymś, co czuje się szczęśliwym, gdy gra na fortepianie, lubi chodzić na spacer i, w samej rzeczy, chce czynić mnóstwo rzeczy, które są zbędne..., ale pracująca maszyna nie musi grać na fortepianie, nie musi czuć się szczęśliwa, nie musi czynić mnóstwa innych rzeczy. Wszystko, co nie przyczynia się bezpośrednio do postępu pracy powinno być wyeliminowane.

Opierając się na tej próbce dialogu, nie wydaje się, aby każdy niezwłocznie powinien ją obejrzeć. Sugeruje ona jednak wprowadzenie nowego słowa do języka angielskiego:

> **robot** [< Czech *robotnik*, serf, or *robota*, compulsory service]. 1. To perform boring or repetitive work; 2. To perform subhuman or dangerous work; 3. To perform work which is void of dignity or promise of improvement.

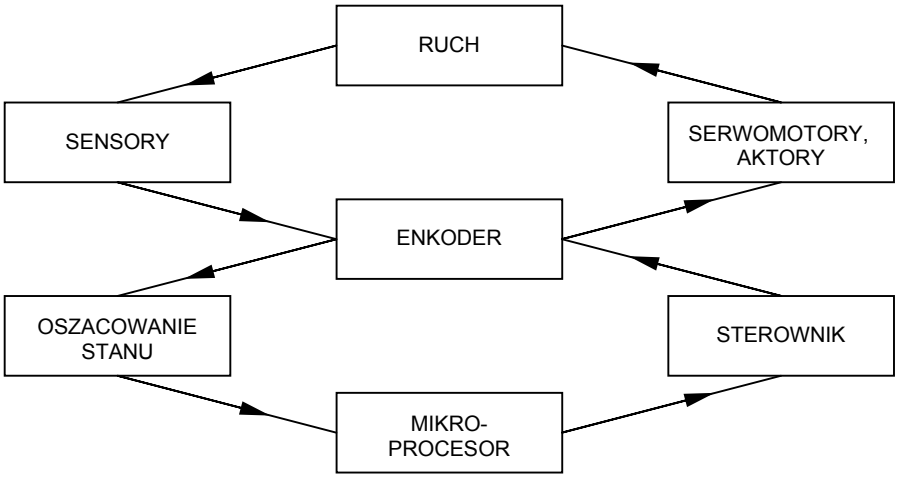
Czytelnik może dodać swoje własne definicje.

Liczbowe porównywania robotów zainstalowanych w różnych krajach mają niewielki sens dopóty, dopóki nie istnieje zgoda na to co tworzy (konstytuuje) robot. Taką zgodę osiągnięto niedawno, gdy Międzynarodowa Organizacja Standardów (ISO) ustaliła następującą definicję:

Definicja 1: Robot przemysłowy. Robot przemysłowy jest automatycznym, sterowanym ze wspomaganiem, dowolnie programowalnym, wielocelowym manipulatorem z kilkoma osiami dla operowania przedmiotami, narzędziami czy specjalnymi urządzeniami. Zmiennie programowane operacje czynią możliwym wykonywanie różnorodnych zadań.

Definicja ta była wcześniej definicją niemiecką. Jasno ona rozróżnia miedzy robotami typu „podnieś i umieść" (*pick-and-place robots*), które poruszają się z punktu do punktu z pełną szybkością i robotami, gdzie ma miejsce sterowanie ścieżką miedzy punktami.

W tym kontekście *dowolna programowalność* oznacza, że roboty są w stanie wykonywać więcej niż jedno zadanie; a *manipulacja* (i transport) wskazują, że robot nie jest jakimś narzędziem, lecz odwrotnie, używa narzędzi do wykonywania przepisanych zadań – generalnie zadań wytwarzania w środowisku produkcyjnym. Zgrubny diagram operacyjny dla robota pokazuje rysunek 3. Ponieważ ruch sterowany jest pierwotnym celem robota, jest to ruch, który jest wyczuwany (*sensed*), przetwarzany (*processed*) i sterowany (*controlled*).

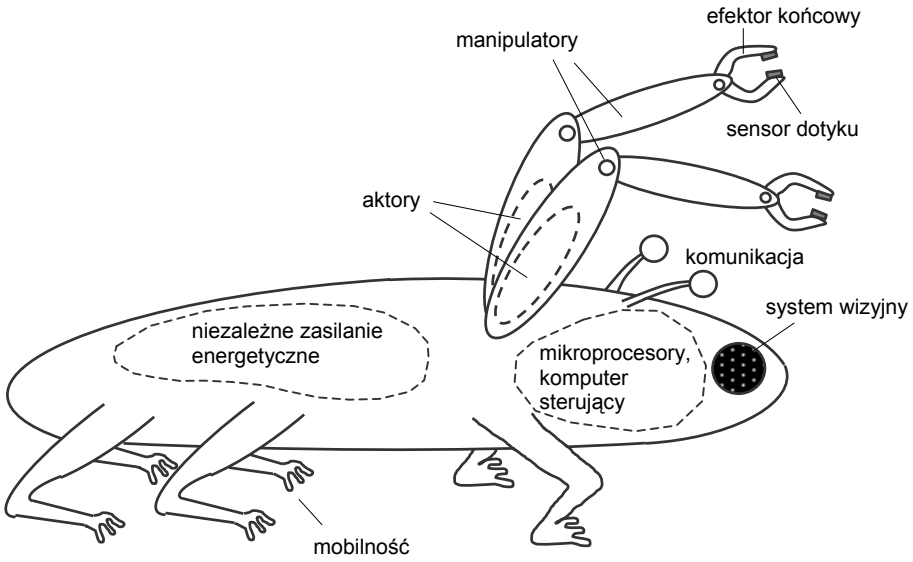


Rys. 3 Diagram pracy robota

Istotne komponenty robota ilustruje rysunek 4, pokazując to, co mogłoby być nazywane „roboach" (pol. „robot-karaluch"). Pokazane komponenty są obecne w każdym robocie, choć roboty niemobilne mogą nie mieć żadnego własnego zasilania energetycznego czy jednostek komunikacyjnych. Te komponenty można w skrócie opisać następująco:

- *Aktor* (actuator) – służy jako muskuł systemu, wytwarza ruch za pomocą energii elektrycznej, pneumatycznej lub hydraulicznej.

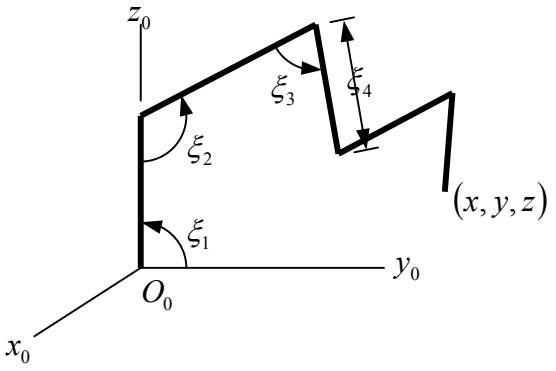
- *Komunikator* (communicator) – jednostka przesyłająca informację i otrzymująca instrukcje od oddalonego operatora.
- *Efektor końcowy* (end effector) – urządzenie chwytające na końcu ramienia manipulatora; używane do dokonywania zamierzonego styku z obiektem lub do wytwarzania finalnego efektu robota na jego otoczeniu.
- *Manipulator* (manipulator) – mechanizm zawierający kilka segmentów lub ramion.
- *Zasilacz energii* (power supply) – ogólnie urządzenia magazynujące energię, takie jak akumulatory dla jednostki mobilnej; także elektroniczne urządzenie zasilające.
- *Sensor* (sensor) – zwykle jakiegoś rodzaju przetwornik, którego wejściami są zjawiska fizykalne a wyjściami sygnały elektroniczne.



Rys. 4 Komponenty robota

Inna klasyfikacja robotów opiera się na ich konfiguracji fizycznej. Istnieją dwa istotne zbiory zmiennych dla robota: *zmienne zewnętrzne* czyli *współrzędne globalne*, które będą oznaczane po prostu przez (x, y, z), oraz korespondujące z nimi *zmienne wewnętrzne* czyli *współrzędne robota* $(\xi_1, \ldots, \xi_n)$, których wartości muszą być określane w taki sposób, że niektóre punkty struktury robota pokrywają się ze współrzędnymi globalnymi (x, y, z). Przypadek ogólny jest zobrazowany na rysunku 5.

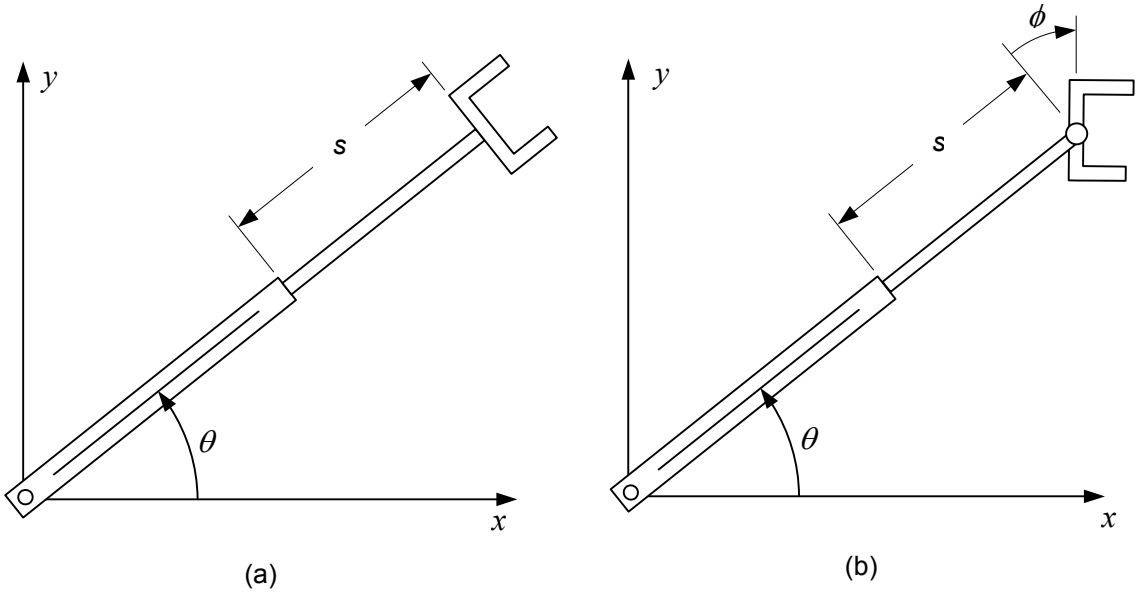
Załóżmy teraz, że tylko zmienna ξ_i może być aktywnie zmieniana za pomocą jakiegoś wewnętrznego aktora. Przykładowo na rysunku 5 nie może to być zmienna ξ_1, jeżeli jest ustalona jako 90°; wybranie ξ_4 wymaga możliwości zmieniania długości ξ_4. Tego rodzaju wymagania tworzą podstawę następującej definicji.



Rys. 5 Zmienne robota i zmienne otoczenia

Definicja 2: Liczba osi. Liczba osi robota jest równa liczbie zmiennych robota, które są aktywnie sterowane.

Używamy tu rozmyślnie słowa *osie*, ponieważ jest ono jednoznacznie identyfikowane z robotyką. Terminy *stopnie ruchliwości* i *stopnie swobody* są również używane w tym kontekście. Każde połączenie ruchowe (obrotowe lub posuwiste), sterowalne lub nie, dodaje stopień ruchliwości do robota. Termin *stopnie swobody* ma bardzo specyficzne znaczenie w mechanice. Ciało sztywne, poruszające się w przestrzeni ma sześć stopni swobody: trzy postępowe (translacyjne) i trzy obrotowe (rotacyjne). Załóżmy, że robot ma podnieść ciało sztywne; wtedy różne ruchy, jakie robot mógłby przenieść na ciało sztywne będą konstytuować (tworzyć) stopnie swobody robota. Jeżeli miałby on naśladować swobodny ruch ciała stałego, to potrzebowałby mieć sześć stopni swobody. Ciało sztywne w ruchu płaskim ma trzy stopnie swobody: dwa posuwiste i jeden obrotowy. Na przykład prosty manipulator biegunowy na rysunku 6a ma dwie aktywnie sterowane zmienne (robot dwuosiowy) i dwa stopnie swobody; przegub przy chwytaku zaopatruje go w trzeci stopień swobody (rysunek 6b). W tym miejscu staje się oczywiste, że w celu posiadania *n* stopni swobody robot musi mieć przynajmniej *n* osi. Gdy istnieje więcej osi niż stopni swobody, to mamy osie nadmiarowe (redundantne), które mogą służyć do poprawy charakterystyk ruchu. Te osie dodatkowe stwarzają jednak również problemy. Ma to miejsce wtedy, gdy szczegółowy ruch w przestrzeni ma być opisany w postaci zmiennych robota.



Rys. 6 Stopnie swobody prostego manipulatora biegunowego

Osie manipulatora są podstawą następujących pięciu głównych klasyfikacji robota z punktu widzenia ich konfiguracji fizycznych.

Definicja 3: Klasyfikacje robota.

Robot kartezjański. Robot porusza się wzdłuż trzech podstawowych osi posuwistych x_0, y_0 i z_0 (rysunek 7a).

Robot cylindryczny. Możliwości ruchowe manipulatora są określone przez współrzędne cylindryczne (r, θ, z) (rysunek 7b).

Robot sferyczny. Możliwości ruchowe manipulatora określają współrzędne sferyczne (r, θ, ϕ) (rysunek 8c).

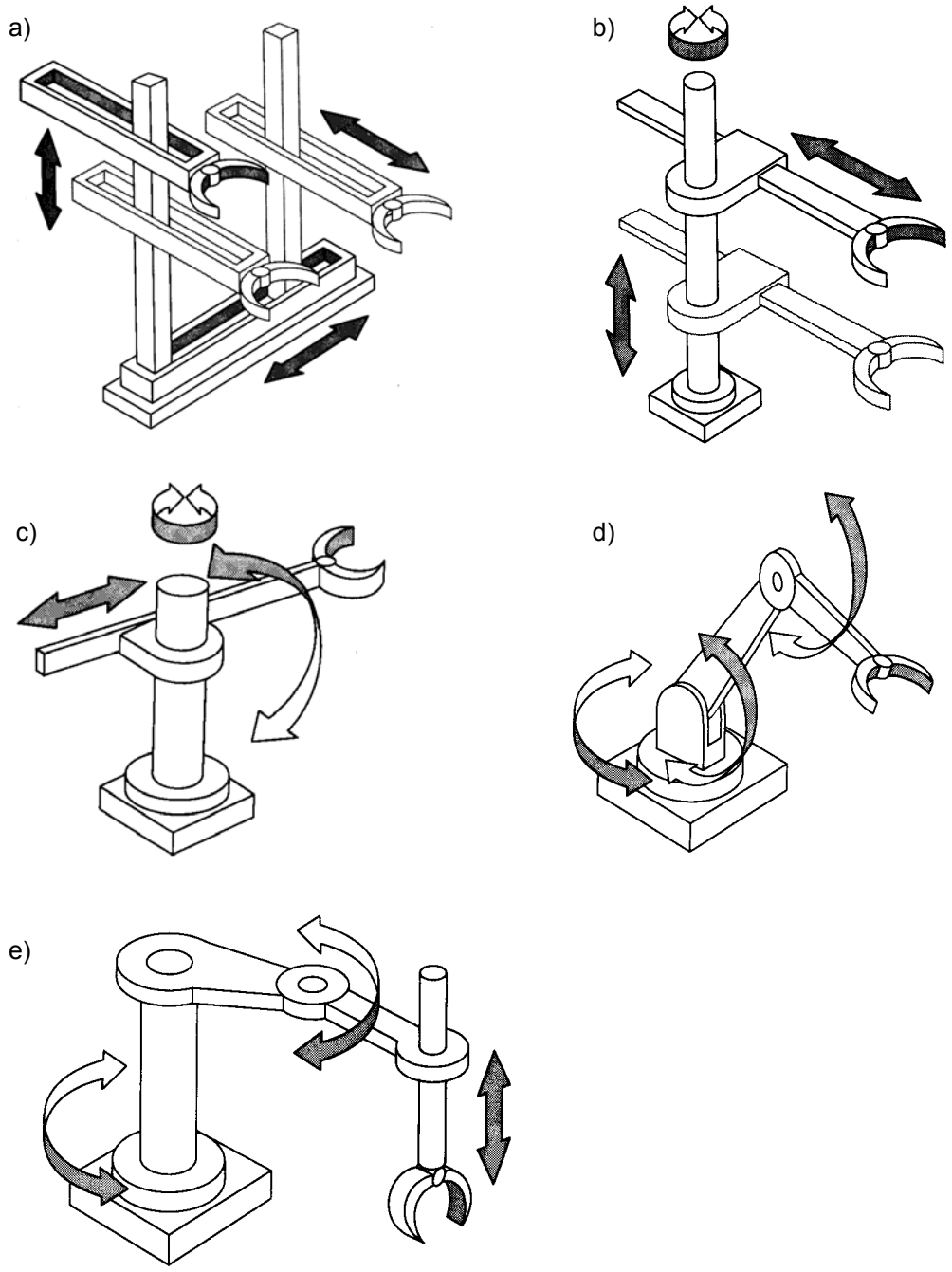
Robot obrotowy. Manipulator antropomorficzny (rysunek 7d).

Robot SCARA. Możliwości ruchowe manipulatora sprowadzają się do ruchów obrotowych ograniczonych do płaszczyzny poziomej i do przesuwania tej płaszczyzny (rys. 7e).

Każda konfiguracja ma swe wady i zalety – szczególnie gdy chodzi o punkty przestrzeni, do których robot sięga.

Definicja 4: Przestrzeń robocza. Obwiednia robocza. Niech *A* będzie punktem na efektorze końcowym robota. Wtedy wszystkie punkty globalne (x, y, z), które *A może* zająć, tworzą *przestrzeń roboczą robota*; granica przestrzeni roboczej nazywana jest *obwiednią roboczą*.

Przeszkody zawierają punkty, których *A* nie może zająć. Tak więc podana definicja przestrzeni roboczej również wyklucza wszystkie wewnętrzne przeszkody, takie jak struktura (podstawa) podtrzymująca robot.



Rys. 7 Standardowe konfiguracje robotów: a) robot kartezjański, b) cylindryczny, c) sferyczny, d) antropomorficzny, e) SCARA

Rozdział 2

Ręka robota – manipulator

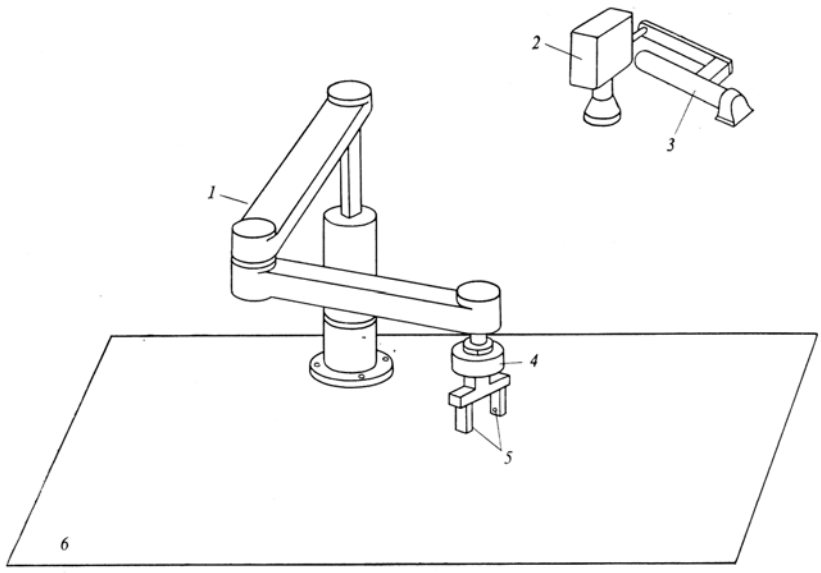
Według: ISII T., SIMOJAMA. I., INOUE H., HIROSE, M., NAKADZIMA N.: *Mechatronika* (w jęz. ros.). Mir 1988

Robot hipotetyczny, jego funkcje i elementy
Ręka robota i przedstawienie jej położenia
Przekształcenia układów współrzędnych ręki
Geometria i kinematyka ręki
Napęd nadążny manipulatora
Zbiór funkcji manipulatora

Robot hipotetyczny, jego funkcje i elementy

Jeżeli rozpatrujemy budowę robota jako systemu uniwersalnego, to należy zacząć od zbioru jego podstawowych funkcji. Celem wykładu jest określenie tego zbioru. Aby nasze rozważania były bardziej konkretne, przyjmiemy, że mamy do czynienia z pewnym robotem hipotetycznym. Będzie on przedmiotem naszych badań – obiektem na którym będziemy rozważać podstawowe zagadnienia robotyki. Jak powinien wyglądać taki hipotetyczny robot? Przede wszystkim powinien to być robot możliwie prosty, nieskomplikowany. Wprowadzenie nadmiernego uproszczenia może jednak doprowadzić do zwykłych rozważań abstrakcyjnych, które nie będą miały niewiele do czynienia z rzeczywistością. Dlatego robot hipotetyczny powinien być kompromisowym rozwiązaniem między abstrakcją a konkretną rzeczywistością. Rodzajem prac, jakie robot hipotetyczny wykonuje, powinny być prace montażowe, które najdłużej opierały się automatyzacji i stały się przez to z jednej strony atrybutem umiejętności człowieka z drugiej zaś atrybutem robotyzacji w odniesieniu do automatyzacji.

Robot hipotetyczny powinien być także kompromisem między robotem specjalistycznym a robotem uniwersalnym. Na podstawie wielostronnych badań robotów znaleziono taki kompromis w postaci hipotetycznego robota pokazanego na rysunku 8. Robot ten ma taką konstrukcję, że może służyć jako konkretny model podczas opracowywania realnego robota montażowego. Robot hipotetyczny jest prosty, dysponuje jednak przy tym dostateczną uniwersalnością.



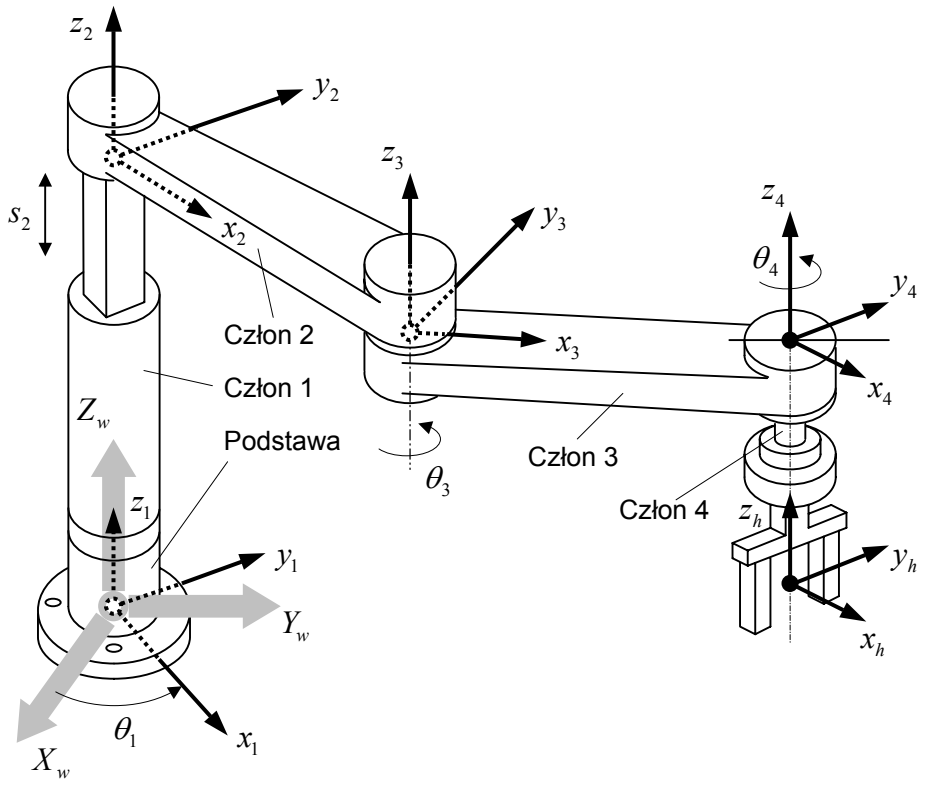
Rys. 8 Schemat robota hipotetycznego.
1 – ręka mechaniczna (manipulator) o czterech stopniach swobody, pozwalająca sterować pozycją dłoni w przestrzeni trójwymiarowej i orientować ją kątowo względem osi pionowej; dłoń ma mechanizm, który równoległego przemieszcza dwa palce. 2 – kamera telewizyjna; 3 – skaner (laserowe urządzenie przeszukujące); za pomocą dwóch maleńkich luster można dowolnie regulować kąt padania promienia; urządzenie to, w połączeniu z kamerą telewizyjną pozwala określać położenie plamki laserowej w przestrzeni trójwymiarowej; 4 – czujnik siły – urządzenie, mierzące wartości reakcji, działających na dłoń w sześciu osiach; 5 – czujnik taktylny (dotykowy); 6 – miejsce robocze (stół); w przypadku koniecznym można je wyposażyć w urządzenia wspomagające (zaciski, przenośnik itp.).

Ręka robota ma cztery stopnie swobody. Pozwala to jej zmieniać pozycję w przestrzeni trójwymiarowej i orientację dłoni (kiści) w płaszczyźnie poziomej – wskutek obrotu. W nadgarstek wbudowany jest wieloosiowy czujnik siły. Do chwytania przedmiotów robot ma dwa palce, które zaciskają się i rozchylają się równolegle. Palce wyposażone są w czujniki dotyku. Funkcję widzenia zapewnia kamera telewizyjnej i skaner laserowy. Skaner ten można wykorzystać do pomiaru położenia obiektów procesowanych w przestrzeni trójwymiarowej. Poniżej opiszemy taki robot hipotetyczny i przeanalizujemy funkcji jego podstawowych elementów. Zaczniemy od ręki robota, czyli manipulatora.

Ręka robota i przedstawienie jej położenia

Urządzenie, pozwalające manipulować przedmiotami, nosi nazwę *manipulatora*. Podobnie jak ręka człowieka składa się on z ramion i dłoni (kiści). Ręka jest wieloczłonowym mechanizmem o kilku stopniach swobody. Zakończona jest dłonią przeznaczoną do chwytania przedmiotów lub narzędzi, na przykład śrubokręta. W każdym przegubie ręki jest napędowy układ nadążny, który zapewnia odpowiednie sterowanie pozycją i orientacją dłoni lub narzędziem.

Do przedstawienia pozycji i orientacji ręki w przestrzeni trójwymiarowej służą prostokątne układy współrzędnych związane z członami ręki (rys. 9). Układy te nazywa się układami *lokalnymi*. W przestrzeni roboczej istnieje jeszcze jeden układ współrzędnych, nazywany *globalnym*, który przy obliczaniu pozycji i orientacji spełnia rolę podstawową. Pozycję ręki można przedstawić za pomocą wektorów położenia początków układów lokalnych w układzie globalnym. Orientację można zaś opisać przez obroty układów lokalnych względem układu globalnego. Bardzo często do przedstawienia tych obrotów wykorzystuje się kąty EULERa, przyjąwszy globalny układ współrzędnych jako bazowy.



Rys. 9 Stopnie swobody manipulatora hipotetycznego i układy współrzędnych, związane z członami.

W ogólnym przypadku, w celu osiągnięcia dowolnej pozycji i dowolnej orientacji dłoni w przestrzeni trójwymiarowej, ręka powinna dysponować sześcioma stopniami swobody. W robocie hipotetycznym postanowiliśmy ograniczyć się do ręki o czterech stopniach swobody (rys. 1). Podstawa (ostoja) ręki hipotetycznej znajduje się na stole roboczym. Ręka dysponuje trzema obrotowymi stopniami swobody ($\theta_1, \theta_2, \theta_3$) i jednym postępowym ($s_2$). Te cztery parametry można nazwać współrzędnymi przegubowymi. Każda z tych współrzędnych przegubowych jest sterowana napędem nadążnym, co pozwala pozycjonować i orientować rękę w przestrzeni w odpowiedni sposób. Dla ręki z rysunku 8 pozycję

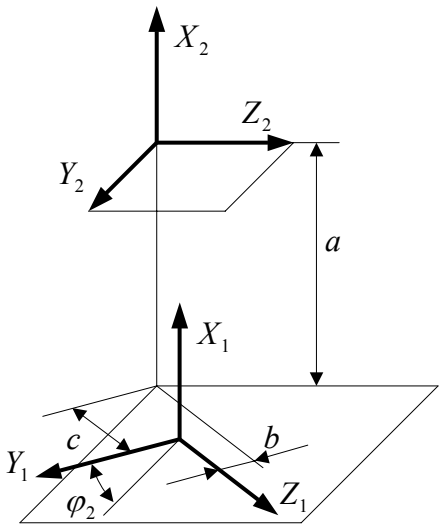
dłoni w układzie globalnym można przedstawić współrzędnymi X,Y,Z początku układu lokalnego dłoni. Podczas rozpatrywania orientacji w ograniczymy się tylko do kata obrotu α względem osi Z globalnego układu współrzędnych. W celu sformalizowania zależności istniejących między współrzędnymi przegubowymi ($\theta_1, s_2, \theta_3, \theta_4$) i współrzędnymi pozycji i orientacji dłoni (X,Y,Z,α) rozpatrzymy tylko przekształcenie lokalnych współrzędnych ręki. Zanim do tego przejdziemy zrobimy jednak najpierw wycieczkę do macierzowego aparatu matematycznego, który pomaga nam przechodzić z jednego układu współrzędnych do drugiego.

Wycieczka do macierzowego opisu przejścia z jednego układu współrzędnych do drugiego

Do opisu ruchu końcówki manipulatora w przestrzeni najbardziej przydatny jest aparat matematyczny oparty na wykorzystaniu macierzy 4×4, które przekształcają jednorodne współrzędne punktów przestrzeni trójwymiarowej. Jednorodnymi współrzędnymi punktu w przestrzeni trójwymiarowej nazywa się dowolne liczby $X_1', Y_1', Z_1', t,$ związane z jej współrzędnymi kartezjańskimi (X, Y, Z) równaniami

$$X = \frac{X_1'}{t}; \qquad X = \frac{Y_1'}{t}; \qquad X = \frac{Z_1'}{t}. \tag{1}$$

Na rysunku 10 mamy dwa układy współrzędnych: T_1 i T_2, przy czym układ T_1 różni się od układu T_2 obrotem wokół osi X_2 o kąt φ_2 i przesunięciem początku układu o odcinki a, b i c. Rozpatrzmy przejście od układu T_2 do układu T_1 za pomocą macierzy.



Rys. 10 Przejście z układu współrzędnych T_2 do układu T_1

We współrzędnych jednorodnych położenie punktu Q w odpowiednich układach zapisuje się w postaci

$$Q(X_1, Y_1, Z_1, t_1) \text{ oraz } Q(X_2, Y_2, Z_2, t_2) \tag{2}$$

Dla prostoty przejścia przyjmuje się $t_1 = t_2 = 1$. Wtedy wzory na przejście od układu T_2 do układu T_1 można zapisać następująco:

$$
\begin{aligned}
X_1 &= X_2 + t_1 a;\\
Y_1 &= Y_2 \cos\varphi_2 - Z_1 \sin\varphi_2 - t_1 b;\\
Z_1 &= Y_2 \sin\varphi_2 + Z_1 \cos\varphi_2 - t_1 c;\\
t_2 &= t_1 = 1.
\end{aligned}
\tag{3}
$$

Współczynniki, stojące przed X_2, Y_2, Z_2 i t_2, można przedstawić w postaci macierzy

$$
M_{12} = \begin{vmatrix}
1 & 0 & 0 & a\\
0 & \cos\varphi_2 & -\sin\varphi_2 & -b\\
0 & \sin\varphi_2 & \cos\varphi_2 & -c\\
0 & 0 & 0 & 1
\end{vmatrix}
\tag{4}
$$

Indeksy $_{12}$ w oznaczeniu macierzy przejścia M_{12} pokazują, że następuje przejście z układu M_2 do układu M_1.

Wzory (3) można przedstawić jako wynik mnożenia macierzy M_{12} przez macierz kolumnową r_2, przedstawiającą sobą współrzędne punktu Q w układzie M_2:

$$
\begin{vmatrix} X_1\\ Y_1\\ Z_1\\ t_1 \end{vmatrix} = \begin{vmatrix}
1 & 0 & 0 & a\\
0 & \cos\varphi_2 & -\sin\varphi_2 & -b\\
0 & \sin\varphi_2 & \cos\varphi_2 & -c\\
0 & 0 & 0 & 1
\end{vmatrix} \cdot \begin{vmatrix} X_2\\ Y_2\\ Z_2\\ t_2 \end{vmatrix}
\tag{5}
$$

W zapisie skróconym wyrażenie to wygląda następująco:

$$
r_1 = M_{12} r_2 .
\tag{6}
$$

Rozpatrzymy teraz zastosowanie macierzy o rozmiarze 4 × 4 do opisania kinematyki naszego manipulatora hipotetycznego (SCARA).

Przekształcenia układów współrzędnych ręki

Aby uzyskać przekształcenia, które powiążą układ współrzędnych dłoni (x_h, y_h, z_h) z globalnym układem współrzędnych (X_w, Y_w, Z_w), każdemu *i*-temu członowi ręki przypisuje się lokalny układ współrzędnych (x_i, y_i, z_i) sztywno powiązany z tym członem. Globalny układ współrzędnych X_w, Y_w, Z_w jest związany z przestrzenią roboczą i wykorzystywany jako podstawa podczas wyprowadzania współrzędnych bezwzględnych. Układ x_1, y_1, z_1 jest układem współrzędnych związanym sztywno z członem 1. Układ ten może się obracać o kąt θ_1 względem osi Z_w globalnego układu współrzędnych. Obrót ten wynika ze stopnia swobody, jaki ma przegub 1. Układ współrzędnych x_2, y_2, z_2 jest związany sztywno z członem 2. Ponieważ połączenie jest przesuwne, to rozpatrywany układ współrzędnych może przemieszczać się w kierunku osi z układu x_1, y_1, z_1. Przy tym zachodzi równoległe przeniesienie na wielkość s_2. Układy x_3, y_3, z_3 i x_4, y_4, z_4 są układami współrzędnych związanymi sztywno, odpowiednio, z członami 3 i 4. Dzięki przegubom 3 i 4 układy te mogą obracać się względem układów poprzednich o kąty θ_3 i θ_4. I wreszcie ostatni lokalny układ współrzędnych – układ x_h, y_h, z_h, związany sztywno z dłonią. Można uważać, że układ ten jest przeniesiony równolegle z początku układu x_4, y_4, z_4 w ujemnym kierunku osi z_4 o wielkość *d*, która jest stała. Zależy ona od wymiarów dłoni i umieszczonego na niej czujnika siły. Rozpatrzmy teraz punkt, który w lokalnym układzie współrzędnych dłoni ma współrzędne x_h, y_h, z_h. W globalnym układzie współrzędnych punkt ten będzie mieć

współrzędne x_w, y_w, z_w. Aby rozpatrywać to w jednorodnym układzie współrzędnych, trzeba te dwa układy wzajemnie powiązać. Można do tego wykorzystać następującą zależność zawierającą macierze przekształceń współrzędnych:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & l_1 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & l_2 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_h \\ y_h \\ z_h \\ 1 \end{bmatrix}. \quad (7)$$

W macierzy tej s_2 jest odległością między początkami układów współrzędnych x_1, y_1, z_1 i x_2, y_2, z_2; l_1 jest odległością między początkami układów współrzędnych x_2, y_2, z_2 i x_3, y_3, z_3; l_2 jest odległością między początkami układów współrzędnych x_3, y_3, z_3 i x_4, y_4, z_4; θ_1, s_2, θ_3 i θ_4 są przemieszczeniami w czterech połączeniach członów (przegubach) ręki dopuszczających, stosownie do stopni swobody, obrót i przesuw. Przemieszczenia te można rozpatrywać jako regulowane wielkości nadążnych układów napędowych. Jeżeli przeprowadzi się odpowiednie przekształcenia, to otrzymuje się, że

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_1+\theta_3+\theta_4) & -\sin(\theta_1+\theta_3+\theta_4) & 0 & l_2\cos(\theta_1+\theta_3)+l_1\cos\theta_1 \\ \sin(\theta_1+\theta_3+\theta_4) & \cos(\theta_1+\theta_3+\theta_4) & 0 & l_2\sin(\theta_1+\theta_3)+l_1\sin\theta_1 \\ 0 & 0 & 1 & s_2-d \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_h \\ y_h \\ z_h \\ 1 \end{bmatrix}. \quad (8)$$

Zależność ta przedstawia przekształcenie układu współrzędnych dłoni do globalnego układu współrzędnych. Ponieważ pozycję dłoni można opisać współrzędnymi (gdy początek układu x_h, y_h, z_h rozpatruje się w układzie globalnym), to przez podstawienie $x_h = y_h = z_h = 0$ w wyrażeniu (8) można określić współrzędne ręki X, Y, Z w układzie globalnym. W pokazanej zależności główna macierz wyraża obrót między układem lokalnym dłoni i układem globalnym. Zgodnie z tą macierzą można uważać, że układ lokalny dłoni otrzymuje się przez obrót układu globalnego względem osi z o kąt $\theta_1+\theta_3+\theta_4$ będący katem orientacji α. Dlatego można zapisać

$$\begin{aligned} X &= l_2\cos(\theta_1+\theta_3)+l_1\cos\theta_1, \\ Y &= l_2\sin(\theta_1+\theta_3)+l_1\sin\theta_1, \\ Z &= s_2-d, \\ \alpha &= \theta_1+\theta_3+\theta_4. \end{aligned} \quad (9)$$

Znając wartości współrzędnych członów ręki hipotetycznej można więc wyliczyć pozycję i orientację dłoni w globalnym układzie współrzędnych.

Geometria i kinematyka ręki

Jeżeli określi się przemieszczenia czterech połączeń ręki hipotetycznej, to za pomocą pokazanych wyżej zależności można dość prosto obliczyć pozycję i orientację dłoni. Dla zadanych współrzędnych położenia i orientacji (X, Y, Z, α), należy określić współrzędne przegubowe członów ($\theta_1, s_2, \theta_3, \theta_4$) i za ich pomocą obliczyć następnie pozycję i orientację dłoni. Z teoretycznego punktu widzenia jako wielkości niewiadome należy wtedy przyjąć wartości współrzędnych połączeń i rozwiązać wprowadzone wyżej równanie przekształcenia współrzędnych. Takie rozwiązanie można nazwać *rozwiązaniem kinematyki ręki*. Należy mieć przy tym jednak na uwadze, że ze wzrostem liczby stopni swobody ręki rozwiązanie analityczne staje się coraz trudniejsze i mniej dokładne. Przez to nie ma szerszego zastosowania w praktyce. Dlatego często podczas znajdowania rozwiązania dla ręki wykorzystuje się formalizację opartą na geometrii (rysunek 11).

Z twierdzenia kosinusów dla trójkąta *OE'W'* otrzymuje się

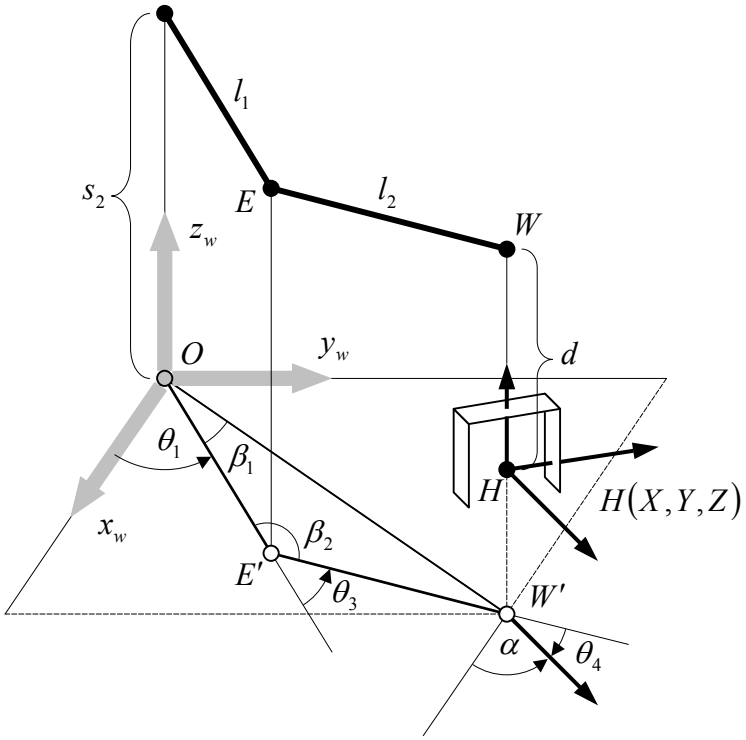
$$\cos\beta_2 = \frac{l_1^2 + l_2^2 - \left(X^2 + Y^2\right)}{2l_1 l_2}. \tag{10}$$

Jeżeli trzeci przegub nie ma ograniczeń kata obrotu θ_3, to dla kąta β_2 istnieją dwa rozwiązania. Jeżeli $0 \le \theta_3 \le \pi$, to dla kąta β_2 istnieje tylko jedno rozwiązanie. Otrzymuje się wtedy

$$\theta_3 = \pi - \beta_2. \tag{11}$$

Podobnie dla trójkąta *OE'W'* z twierdzenia kosinusów otrzymuje się

$$\cos\beta_1 = \frac{\left(X^2 + Y^2\right) + l_1^2 - l_2^2}{2l_1\sqrt{X^2 + Y^2}}. \tag{12}$$



Rys. 11 Geometryczna ilustracja znajdowania rozwiązania dla ręki.

Dla przyjętego wcześniej ograniczenia z zależności tej można otrzymać również tylko jedna wartość β_1.

Ponieważ $\beta_1 + \theta_1 = \operatorname{arc}\operatorname{ctg}\frac{Y}{X}$, to

$$\theta_1 = -\beta_1 + \operatorname{arc}\operatorname{ctg}\frac{Y}{X}. \tag{13}$$

Przemieszczenie (przesunięcie) w drugim przegubie i kąt obrotu w czwartym przegubie (uwzględniając rysunek 4) można przedstawić następująco:

$$\begin{aligned} s_2 &= Z + d, \\ \theta_4 &= \alpha - \left(\theta_1 + \theta_3\right). \end{aligned} \tag{14}$$

Ponieważ ręka hipotetyczna ma prostą konstrukcję, rozwiązania otrzymane dla ręki tez są stosunkowo proste. Dla ręki o sześciu stopniach swobody zadanie okazuje się nieco złożone. W zależności od założonych zakresów kątów przemieszczeń przegubów może powstać taka sytuacja, kiedy te same

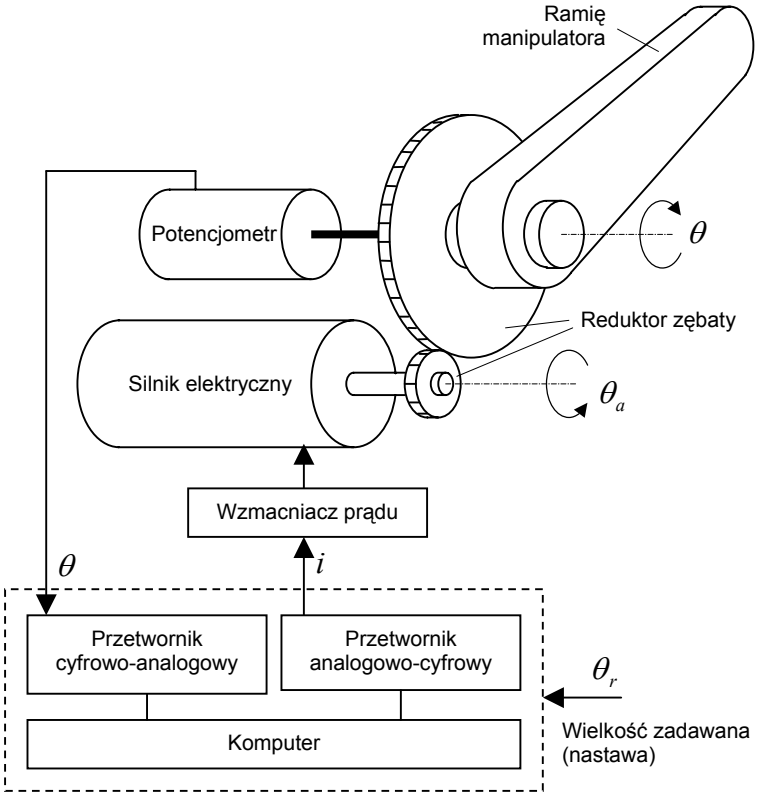
wartości pozycji i orientacji otrzymuje się dla więcej niż jednej kombinacji współrzędnych przegubów. Dlatego należną uwagę należy przywiązać do wyboru i sprawdzenia rozwiązania.

Napęd nadążny manipulatora

Manipulator jest wieloczłonowym mechanizmem, którego przeguby ustalane są w określonej pozycji za pomocą napędu nadążnego. Napęd ten jest umieszczony z reguły w każdym przegubie manipulatora i steruje współrzędną przegubową. W ten sposób odbywa się sterowanie pozycją i orientacją dłoni lub narzędzia umieszczonego na końcu manipulatora. Do sterowania połączeniami hipotetycznej ręki zastosujemy napęd nadążny wyposażony w silnik prądu stałego (rysunek 12). Wartość prądu i, płynącego z przetwornika cyfrowo-analogowego, jest wzmacniana i przykładana do silnika elektrycznego, na którego wyjściu powstaje moment M prawie proporcjonalny do wartości prądu. W ten sposób, zakładając, że zasilanie silnika następuje od pewnego idealnego wzmacniacza prądu, moment obrotowy na wyjściu silnika można określić następująco:

$$M = k_t i_a = k_t A i \,, \tag{15}$$

gdzie: k_t – współczynnik momentu obrotowego silnika; *A* – współczynnik wzmocnienia prądu we wzmacniaczu.



Rys. 12 Schemat nadążnego napędu ramienia manipulatora

Zbudujmy teraz równania ruchu napędu. Posłużymy się przy tym następującymi oznaczeniami: θ_a – kąt obrotu wału silnika; θ – kąt obrotu ramienia manipulatora; r – promień koła zębatego osadzonego na wale silnika; N – przełożenie reduktora; P – siła styczna w zazębieniu reduktora; T_1 – moment obciążenia zewnętrznego przyłożonego do ramienia manipulatora; J_m – moment bezwładności wirnika silnika; J_T – moment bezwładności obciążenia (ramienia manipulatora). Równania ruchu można przedstawić w postaci

$$
\begin{aligned}
&J_m \cdot \ddot{\theta}_a = M - P \cdot r, \\
&J_1 \cdot \ddot{\theta} = P \cdot N \cdot r - T_1, \\
&\theta_a = N \cdot \theta.
\end{aligned}
\tag{16}
$$

Po przekształceniu otrzymuje się następującą zależność między prądem silnika i a katem obrotu θ ramienia manipulatora:

$$
\left(J_m + \frac{J_1}{N^2} \right) \ddot{\theta} = \frac{k_t \cdot A}{N} i - \frac{T_1}{N^2}. \tag{17}
$$

Jeżeli chcemy otrzymać nadążny (śledzący) napęd pozycjonowania, zapewniający regulację proporcjonalną i różniczkową, to trzeba wprowadzić sprzężenie zwrotne z sygnałem wejściowym

$$
i = k_e \left(\theta_r - \theta \right) + k_v \left(\ddot{\theta}_r - \ddot{\theta} \right), \tag{18}
$$

gdzie: θ_r – zadawana wartość (nastawa) układu nadążnego; k_e – położeniowy współczynnik wzmocnienia proporcjonalnego sprzężenia zwrotnego; k_v – prędkościowy współczynnik wzmocnienia proporcjonalnego sprzężenia zwrotnego. Dla pozycjonującego napędu nadążnego o takim sprzężeniu zwrotnym równanie ruchu można zapisać w postaci

$$
\left(J_m + \frac{J_1}{N^2} \right) \ddot{\theta} + \frac{k_t \cdot A \cdot k_v}{N} \dot{\theta} + \frac{k_t \cdot A \cdot k_e}{N} \theta = \frac{k_t \cdot A \cdot k_v}{N} \dot{\theta} + \frac{k_t \cdot A \cdot k_e}{N} \theta_r - \frac{T_1}{N^2}. \tag{19}
$$

Jeżeli wprowadzimy nowe oznaczenia

$$
J = J_m + \frac{J_1}{N^2}, \qquad K_1 = k_t A \frac{k_e}{N}, \qquad K_2 = k_t A \frac{k_v}{N},
$$

oraz błąd statystyczny układu nadążnego $e = \theta_r - \theta$, to otrzymamy równanie

$$
J\ddot{e} + K_2 \dot{e} + K_1 e = J\ddot{\theta}_r + \frac{T_1}{N^2}. \tag{20}
$$

Przy braku oddziaływania zewnętrznego ($T_1 = 0$) w stanie ustalonym błąd układu nadążnego staje się równy zeru. Zaś w obecności oddziaływania zewnętrznego w stanie ustalonym powstaje odchylenie $T_1 / N^2 K_1$ proporcjonalne do tego oddziaływania. Charakterystyki rozpatrywanego napędu nadążnego można regulować doborem parametrów K_1 i K_2. Pierwszy odpowiada współczynnikowi oddziaływania po odchyleniu, drugi – pochodnej tego współczynnika. Za pomocą parametru K_1 można zmieniać sztywność i częstość skrętnych drgań własnych. Wybór małej wartości K_1 jest równoznaczny ze zmniejszeniem sztywności i z możliwością zginania w przegubie pod wpływem małej siły zewnętrznej. Inaczej mówiąc, skrętne drgania własne będą mieć małą częstość $\omega_n = \sqrt{K_1 / J}$ i szybkość reakcji układu nadążnego pogorszy się. Po określeniu częstości ω_n, wykorzystując współczynnik K_2, można „odregulować" charakterystykę tłumienia układu. Kryterium tłumienia układu jest współczynnik tłumienia $\zeta = \left(K_0 + K_2 \right) / 2\sqrt{JK_1}$. Zwykle wybiera się taką wielkość K_2, aby współczynnik tłumienia ζ wynosił około 0,8.

Rozpatrzone powyżej równanie ze sprzężeniem zwrotnym realizuje się za pomocą programu komputerowego. Kąt obrotu ramienia manipulatora wyjściowego mierzy się potencjometrem. Zmierzona wartość wchodzi do przetwornika analogowo-cyfrowego i następnie do na wejście komputera. Wyliczane jest oddziaływanie sterujące sprzężeń zwrotnych – położeniowego i prędkościowego. Otrzymany wynik przetwarzany jest z postaci cyfrowej w analogową. W ten sposób regulowane jest natężenie prądu wprowadzającego w ruch silnik elektryczny.

Zbiór funkcji manipulatora

Powyżej rozpatrzyliśmy podstawowe elementy do sterowania manipulatorem (przekształcenie współrzędnych dla ręki, układ nadążny itp.). Spróbujemy teraz powiązać wzajemnie te elementy i sformułujemy podsystem sterujący manipulatorem.

Powiedzieliśmy, że położenie dłoni w układzie globalnym można przedstawić współrzędnymi X,Y,Z charakteryzującymi początek układu współrzędnych dłoni. Orientację dłoni można opisać kątem α, o jaki układ ten jest obrócony wokół osi Z układu globalnego. Zadawane wielkości pozycji dłoni i jej orientacji można (przypisując im indeks r) zapisać w postaci X_r, Y_r, Z_r, α_r. Rozwiązanie dla ręki już sformalizowaliśmy. Współrzędne przegubów, odpowiadające współrzędnym X_r, Y_r, Z_r, α_r, można zapisać w postaci $\theta_1, s_2, \theta_3, \theta_4$. Są to zadawane wielkości pozycjonujących napędów nadążnych, wprawiających w ruch połączenia ręki. W rozpatrzonych napędach nadążnych działania sprzężenia zwrotnego zwykle powtarzają się w krótkim okresie, około 2 ms. Celem regulacji jest pokrywanie się otrzymanych wartości kątów między członami z zadawanymi wartościami tych kątów.

Rozwiązanie kinematyki ręki o sześciu stopniach swobody wymaga dość długich obliczeń. Utrudnia to ich wprowadzenie w cykl pracy napędu nadążnego. Dlatego z całego cyklu wydziela się obliczenia dotyczące znajdowania rozwiązania kinematyki ręki, tzn. rozdziela się obliczenia wielkości zadanych i obliczania dotyczące pracy napędów nadążnych. Podczas projektowania układów sterowania zakłada się, że przy obliczeniach dotyczących pracy układów śledzących częstotliwość powtórzeń wynosi około 500 Hz, a przy obliczeniach wielkości zadanych jest dość mała. Przyjmiemy jednak, że budujemy taki układ sterowania, w którym obliczenia dotyczące zarówno znajdowania rozwiązania dla ręki, jak i pracy napędów nadążnych, wykonywane są z jedną częstotliwością, charakterystyczną dla napędów nadążnych, wynoszącą około 500 Hz.

Przedstawimy teraz w ogólnych zarysach program, w którym podczas jednego cyklu pracy napędu nadążnego znajduje się rozwiązanie dla ręki i wykonuje niezbędne obliczenia dla napędu nadążnego.

Algorytm rozwiązania kinematyki ręki i regulacji napędu nadążnego

1. Etapy 2 i 3 wykonywane są wielokrotnie podczas przerywania przez przekaźnik czasowy (timer) z częstotliwością około 500 Hz.
2. Określane jest rozwiązanie dla ręki, to znaczy na podstawie wartości współrzędnych X_r, Y_r, Z_r, α_r wyliczane są wartości $\theta_{1r}, s_{2r}, \theta_{3r}, \theta_{4r}$.
3. Dla czterech członów ręki wykonywane są kroki 3.1-3.4.
   3.1. Wartości θ kątów obrotu członów przekształcane są, za pomocą przetwornika analogowo-cyfrowego (ADC), i następnie zliczane.
   3.2. Obliczane jest odchylenie (rozregulowanie, uchyb) $e = \theta_r - \theta$. Na podstawie otrzymanych wcześnie wartości odchylenia (rozregulowania) określana jest wartość $\dot{e}$.
   3.3. Obliczane jest działanie regulujące $i = k_e \cdot e + k_v \cdot \dot{e}$.
   3.4. Wartość i przekształcana jest, za pomocą przetwornika cyfrowo-analogowego (DAC), i w postaci sygnału sterującego podawana do wzmacniacza prądu.

Zatrzymamy się teraz na sterowaniu pracą dłoni. Składa się ono ze sterowania szerokością otwarcia dwóch palców i sterowania siłą zacisku. W pierwszym przypadku można postąpić tak samo, jak w przypadku rozpatrzonych pozycjonujących napędów nadążnych ręki. Podczas sterowania siłą zacisku dobrze jest stosować proporcjonalną zależność między natężeniem prądu wprawiającego w ruch silnik elektryczny a powstającym przy tym momentem obrotowym. Prąd i, odpowiadający zadanej sile zacisku, przetwarzany jest za pomocą przetwornika cyfrowo-analogowego i podawany do wzmacniacza prądu. Mechanizm palców rozwija przy tym prądzie moment Nk_tAi, co wystarcza do utrzymania przedmiotu. Do sterowania pracą dłoni niezbędne są dwa rodzaje sterowania – sterowanie pozycją i sterowanie siłą. Na początku cyklu pracy napędu nadążnego badana jest wartość „chorągiewki" wskazującej na typ równania i, zgodnie z otrzymaną wartością, następuje przełączenie na sterowanie pozycyjne lub sterowanie siłowe.

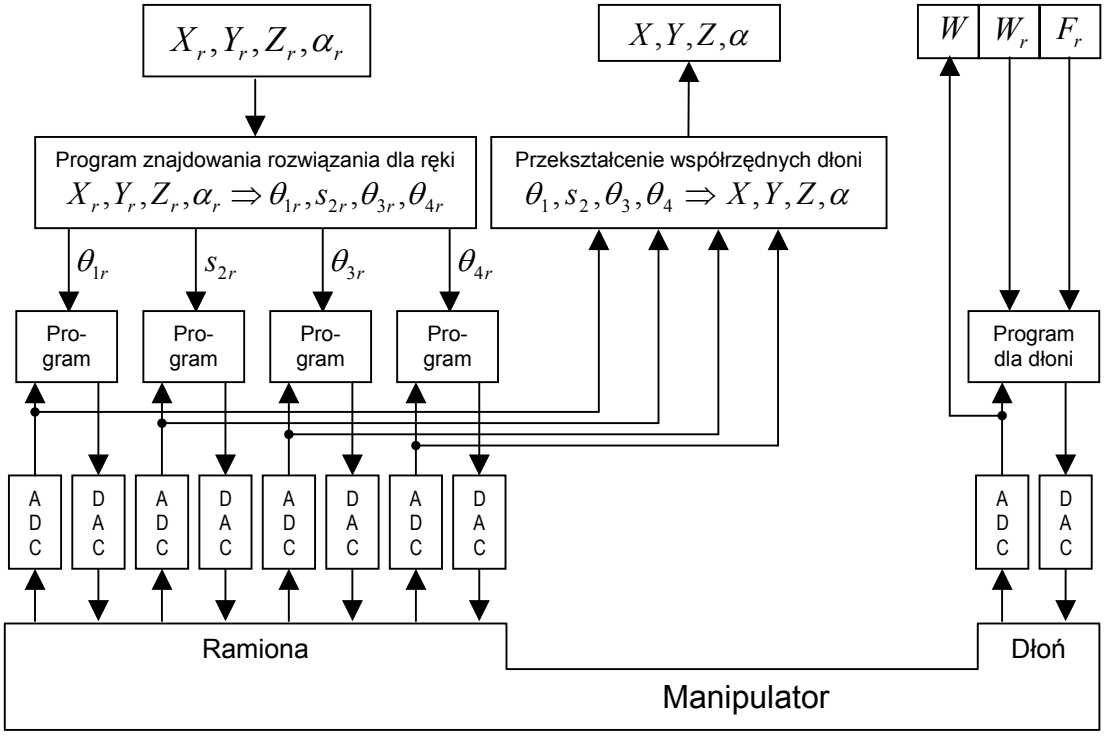
Jeżeli poszukiwanie rozwiązania dla ręki i obliczania w napędach nadążnych wykonuje się jednocześnie, to można uważać, że pozycja i orientacja dłoni obrabiane są dla wartości zadanych. Oznacza to, że z zadanych dla ręki wyżej ukazanych wartości można określić wielkości, które charakteryzują rze-

czywistą pozycję i orientację dłoni. Taką pozycję i orientację warto znać na przykład wtedy, gdy ręka zaczyna pracować, gdy wykonuje się specjalne sterowanie, przy powstaniu błędu, a także w innych przypadkach. Niezbędny jest więc taki podsystem manipulatora, który pozwoliłby określać (mierzyć, zliczać) wartości współrzędnych przegubów i wyliczać, dla rozpatrywanej chwili, pozycję i orientację dłoni.

Algorytm wyliczenia pozycji i orientacji dłoni

1. Współrzędne $\theta_1, s_2, \theta_3, \theta_4$ członów ręki przetwarzane są za pomocą przetwornika analogowo-cyfrowego i są zliczane.
2. Za pomocą zależności (9) wyliczane są wartości X, Y, Z, α .

Teraz można zbudować oprogramowanie dla układu sterującego manipulatora. Schemat tego oprogramowania pokazuje rysunek 13. Układ sterowania wyposażony jest we własny zegar napędowy (timer, który zadaje cykl pracy napędów nadążnych. Zadawane co 2 ms wartości X_r, Y_r, Z_r, α_r – odpowiadające globalnemu układowi współrzędnych i znajdujące się w rejestrze wartości zadawanych – przekształcane są w zadawane wartości członów. Zaczynają pracować programy napędów nadążnych ramion i ręka wprawiana jest w ruch. Wyliczenia powtarzane są co 2 ms. Realizowane jest sterowanie, które dąży do pokrycia się wartości w rejestrze wielkości zadawanych z parametrami charakteryzującymi stan ręki. Przy wykorzystaniu takiego układu sterowania i ruchu ręki można więc w rejestrze zadawanych wielkości ustalić tylko zadawane wartości pozycji i orientacji w globalnym układzie współrzędnych. Analogicznie w napędzie nadążnym dłoni, przez działanie zegara, automatycznie wykonywane jest wielokrotne sterowanie o stałym cyklu. W napędzie nadążnym dłoni niezbędne jest przełączenie rodzajów sterowania. Mamy tu bowiem zarówno sterowanie pozycją (szerokością rozwarcia palców) jak i sterowanie siłą (zaciskiem).



Rys. 13 Układ sterowania manipulatorem.

Pięć podstawowych funkcji manipulatora

Rozpatrzymy teraz zestaw podstawowych funkcji, które zapewniają manipulatorowi zamierzone działanie. W naszym przypadku robota hipotetycznego wystarczy rozważenie następujących pięciu funkcji:

1. Funkcji **put_arm_reference** (X_r, Y_r, Z_r, α_r). Funkcja ta która zapewnia taki ruch ręki, że dłoń przyjmuje pozycję i orientację odpowiadającą w globalnym układzie współrzędnych współrzędnym

X_r, Y_r, Z_r, α_r. Podczas spełniania tej funkcji wartości argumentów X_r, Y_r, Z_r, α_r ustala się w rejestrze wielkości zadawanych (rysunek 6). Wielkości te przekształcane są, według programu uruchamianego zegarem sterowania, w zadawane wielkości połączeń. W wyniku wielokrotnej pracy napędów nadążnych ręka przemieszcza się we wskazaną pozycję i przyjmuje niezbędną orientację.

2. Funkcji **get_arm_position** (X, Y, Z, α). Dla każdego członu ręki zliczane są odpowiednie wartości i wykonywane obliczenie pozycji i orientacji ręki na podstawie współrzędnych członów. Wyniki przekształcane są we współrzędne X, Y, Z, α .

3. Funkcja **put_hand_opening** (W_r). Szerokość rozwarcia palców W_r umieszczana jest w rejestrze wielkości zadawanych a układ nadążny dłoni przełącza się na sterowanie pozycyjne.

4. Funkcja **put_grasp_force** (F_r). Wartość siły zacisku F_r umieszczana jest w rejestrze a układ nadążny dłoni przełącza się na sterowanie siłowe.

5. Funkcja **get_hand_opening** (W). Zliczana jest wartość szerokości rozwarcia palców i przekazywana do argumentu W .

Polecenia sterowania manipulatorem robota hipotetycznego można sprowadzić do przytoczonych powyżej pięciu funkcji. Wykorzystując różne kombinacje tych funkcji można programować różnorodne działania manipulatora. Pokażemy to w następnych rozdziałach wykładach.

Rozdział 3
Zmysły robota – sensoryka

Według: ISII T., SIMOJAMA. I., INOUE H., HIROSE, M., NAKADZIMA N.: *Mechatronika* (w jęz. ros.). Mir 1988

Sensoryka dotykowa
Sensoryka siłowa
Sensoryka wzrokowa
Procesy równoległe i ich sterowanie
Podstawowe rozkazy synchronizacji
Program sterujący
Aktywacja zestawu funkcji
Sterowanie torem ruchu (trajektorią)

Sensoryka dotykowa

Nasz robot hipotetyczny ma dłoń z dwoma palcami, które podczas rozsuwania i zsuwania się pozostają wzajemnie równoległe. Na wewnętrznych powierzchniach palców znajdują się sensory, reagujące na dotyk. Funkcja sensoryki dotykowej ma postać: **get_touch_sensor** (L, R). Za pomocą tej funkcji zlicza się „zmysły" taktylne (dotykowe) lewego i prawego palca i wprowadza do argumentów L, R .

Sensoryka siłowa

W nadgarstek robota hipotetycznego wbudowany jest sześcioskładowy (sześcioosiowy) sensor siły. Dla „czucia siłowego" można posługiwać się funkcją **get_wrist_force** ($f_x, f_y, f_z, m_x, m_y, m_z$). Za jej pomocą można(rysunek 14):

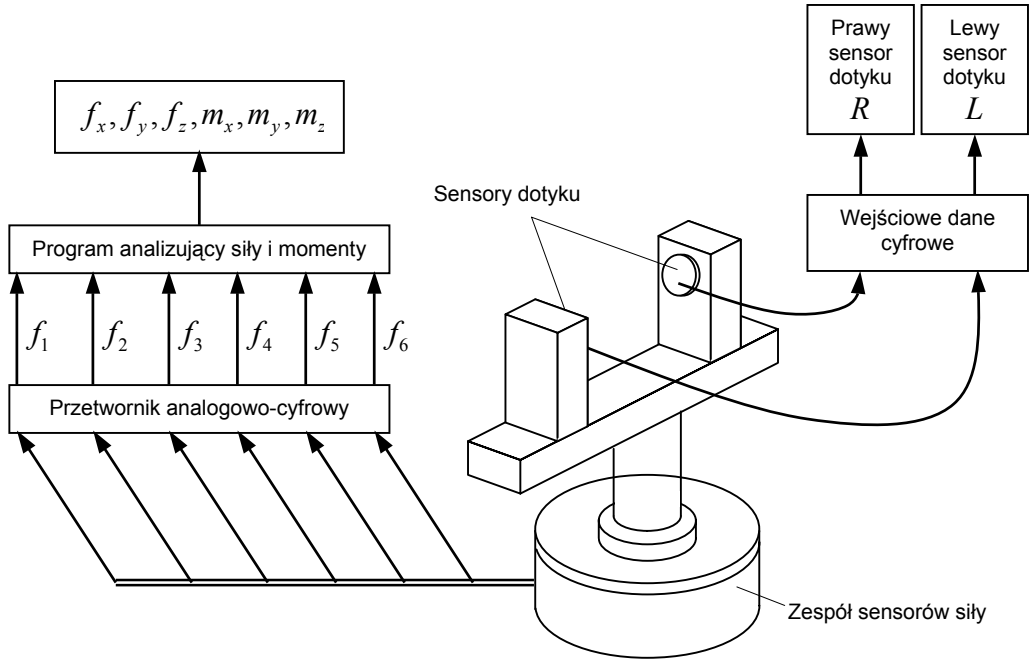
- odczytywać sygnały z sensora siły,
- obliczać siły f_x, f_y, f_z , działające w osiach x, y, z i momenty m_x, m_y, m_z względem tych osi w układzie współrzędnych narzędzia, oraz
- wprowadzać otrzymane wyniki do argumentów.

Sensoryka wzrokowa

Urządzeniem pozwalającym robotowi „widzieć" jest zwykle kamera telewizyjna. Składa się ona się z obiektywu i urządzenia tworzącego sygnał wizerunku na ciele stałym. Przez kwantowanie (zastąpienie opisu ciągłego nieciągłym) sygnału wizerunku, otrzymywanego podczas skanowania jego powierzchni, powstają sygnały cyfrowe, które można obrabiać na komputerze. Obróbka taka pozwala rozpoznawać widziane obrazy. Istnieją liczne sposoby obróbki wizerunków i rozpoznawania obiektów. Opublikowano bardzo dużo wyników badań poświęconych widzeniu technicznemu. Przeprowadzono dużo badań dotyczących sposobów obróbki zorientowanych na problem, z uwzględnieniem ograniczeń nakładanych na obiekty rozpoznawane. Termin „sensoryka wzrokowa" jest bardzo szeroki. Dlatego określenie zwartego zestawu funkcji, który obejmowałby wszystkie zadania sensoryki wzrokowej, jest dość złożone. W naszym wykładzie, podczas rozpatrywania sensoryki wzrokowej robota hipotetycznego, ograniczymy się do identyfikacji pozycji i konfiguracji ciał stosunkowo prostych. Określony w ten sposób funkcje można traktować jako jądro obróbki informacji graficznej przedstawionej na płaszczyźnie w postaci obrazu bitowego.

Jeżeli robot ma manipulować obiektem za pomocą „wzroku technicznego", to powinien rozpoznawać nie tylko konfigurację, lecz również pozycję i orientację tego obiektu. Ręka robota hipotetycznego ma ograniczenie ze względu na stopnie swobody (tylko cztery). Ręka ta jest przeznaczona do manipulowania przedmiotami znajdującymi na płaszczyźnie. Można przyjąć, że wymaganie rozpoznawania konfiguracji, pozycji i orientacji obiektu spełnia system tak zwanej 2,5-wymiarowej sensoryki wzrokowej z organ widzenia na górze. Niezbędne do tego jest określenie 2,5-wymiarowej funkcji pomiaru

odległości. Można do tego wykorzystać urządzenie przeczesujące przestrzeń promieniem laserowym (skaner) przy zastosowaniu zasad triangulacji.



Rys. 14 Sensoryka dotykowa (taktylna) i sensoryka siłowa.

Odwzorowanie bitowe i kodowanie według długości serii

W systemie obróbki obrazów prostokątne pole widzenia rozbija się na komórki, a obraz przedstawia się w postaci bloku (macierzy) dwuwymiarowego. Elementy tego bloku elementy różnią się jasnością punktów siatki. Rozmiar bloku wynosi przykładowo 256×256 lub 512×512. Każdy punkt siatki nazywany jest *elementem obrazu*. Jeżeli wartość każdego elementu przedstawimy informacją jednobitową (element jasny/ciemny), to otrzymamy obraz bitowy (dwójkowy), to znaczy $B = \{b_{ij}\}$, gdzie $b_{ij} = 1$ albo 0. Często obróbce podlega samo odwzorowanie bitowe. Aby zagęścić dane i zwiększyć szybkość obróbki, stosuje się kodowanie długości serii (jednakowo jasnych elementów obrazu). Gdy wzdłuż wierszy pojawiają się w szeregu dwa lub więcej elementów obrazu mających wartość 1, bierze się punkt początkowy i długość szeregu o wartości 1. Na przykład fragmenty ciał, których obrazy pokazuje rysunek 15, podczas skanowania wzdłuż wierszy można określić w następujący sposób:

(numer skanowania i), (punkt początkowy 1) (długość l_1):

(numer skanowania j), (punkt początkowy 2) (długość l_2), (punkt początkowy 3) (długość l_3).

Przy długości serii 1 można uważać, że ma się do czynienia z szumem (zakłóceniem) i nie kodować.

Analiza powiązania

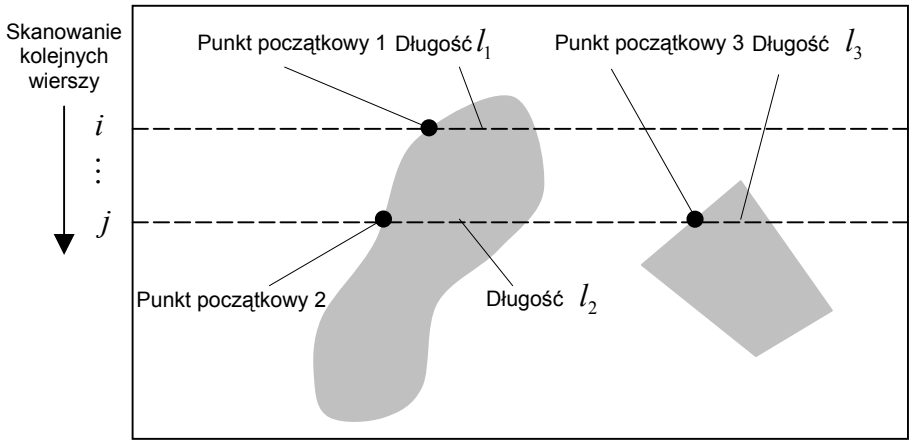
Dane, przetworzone według długości serii, można pogrupować według figur. Jeżeli w dwóch sąsiednich wierszach dane długości serii maja część wspólną w postaci dwóch i więcej elementów obrazu, to można przyjąć, że dane te są *powiązane*. Załóżmy, że $b_{i,n}$ jest punktem początkowym a $l_{i,n}$ długością *n*-tej serii, otrzymanej w *i*-tym wierszu. W takim przypadku warunek powiązania będzie mieć postać

$$s_{i,n} + 2 \le s_{i+1,m} + l_{i+1,m}$$

i

$$s_{i+1,m} + 2 \le s_{i,n} + l_{i,n} .$$

Badając powiązanie danych długości serii można pogrupować dane odniesione do długości serii według figur, które na obrazie istnieją niezależnie od siebie.



Rys. 15 Kodowanie według długości serii.

Podstawowe deskryptory cech figur

Cechy figur płaskich można wyrażać drogą kojarzenia różnych parametrów. W prostych systemach widzenia technicznego, produkowanych przemysłowo, w celu rozpoznania obiektu oblicza się wartości tak zwanych deskryptorów cech figur. Wykorzystuje się przy tym parametry przedstawiające skojarzenia tych wartości. Na przykład w systemie widzenia technicznego SRI istnieją następujące typowe deskryptory:

N – liczba otworów w figurze;

S – powierzchnia figury;

L – obwód figury;

X_c, Y_c – współrzędne środka figury;

$R_{\max}$ – promień maksymalny (maksymalna odległość od środka figury do punktu na obwodzie figury);

$R_{\min}$ – promień minimalny (minimalna odległość od środka figury do punktu na obwodzie figury);

$R_{śr}$ – promień średni (średnia odległość od środka figury do punktu na obwodzie figury);

$\alpha_{\max}$ – kąt między promieniem maksymalnym i osią X;

$\alpha_{\min}$ – kąt między promieniem minimalnym i osią X;

Major – podwojona duża oś elipsy bezwładności;

Minor – podwojona mała elipsy bezwładności;

β – kąt utworzony przez oś X i dużą średnicę elipsy bezwładności;

Length – długość większego boku prostokąta opisanego, którego boki są równoległe do głównych osi bezwładności;

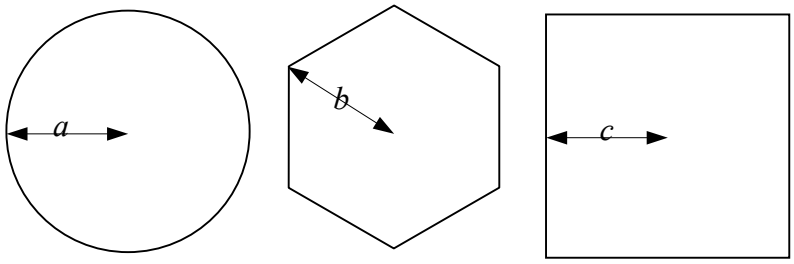
Width – długość mniejszego boku prostokąta opisanego, którego boki są równoległe do głównych osi bezwładności;

$X_{\max}, X_{\min}, Y_{\max}, Y_{\min}$ – maksymalne i minimalne wartości rzutów figury na kierunki osi X i Y, odpowiednio.

Rozpoznawanie figur na podstawie kształtu

Jeżeli rozpoznawane figury znane są wcześniej, to kojarząc podstawowe deskryptory cech można określić parametry klasyfikacji rozpatrywanych obiektów. Takie parametry nazywa się *czynnikami kształtu*. Jako przykład rozpatrzymy klasyfikację okręgu, sześciokąta foremnego i kwadratu. Czynniki kształtu mogą tu być: stosunek powierzchni figury do powierzchni prostokąta opisanego na tej figurze i

stosunek szerokości opisanego prostokąta do jego długości. Kierując się wartościami tych czynników kształtu można łatwo rozpoznawać pokazane trzy figury. Rysunek 16 pokazuje wartości deskryptorów podstawowych cech i czynników kształtu. Wykorzystanie czynników kształtu i ich wartości wymaga pomysłowości w budowaniu procedury rozwiązania. Do szeroko stosowanych czynników kształtu można zaliczyć: $4\pi S/L^2$, $Major/Minor$, $R_{\min}/R_{\max}$, $\alpha_{\min}/\alpha_{\max}$, stosunek pola figury do pola otworu itd.



Parametry		Okrąg	Sześciokąt	Kwadrat
Cechy podstawowe	Obwód L	$2\pi a$	$6b$	$8c$
	Pole S	πa^2	$\left(3\sqrt{3}/2\right)b^2$	$4c^2$
	Szerokość	$2a$	$\sqrt{3}\,b$	$2c$
	Długość	$2a$	$2b$	$2c$
Czynniki kształtu	*S*/szerokość×długość	$\pi/4 = 0{,}785$	3/4 = 0,75	1
	Szerokość/długość	1	$\sqrt{3}/2 = 0865$	1

Rys. 16 Rozpoznawanie figur na podstawie kształtu

Operacje logiczne

Operacjami logicznymi wykonywanymi na figurach są przede wszystkim sumy logiczne, mnożenie, negowanie i wykluczanie.

BOR (B1, B2) — z dwóch obrazów bitowych, B1 i B2, tworzony jest nowy obraz z obliczaniem sumy logicznej poszczególnych elementów;

BAND (B1, B2) — iloczyn logiczny dwóch obrazów bitowych B1, B2;

BNOT B1 — negacja obrazu bitowego B1;

BXOR (B1, B2) — wykluczanie jednego z obrazów bitowych, B1 albo B2 – wykluczające ALBO.

Oprócz tego dla obrazów z kodowaniem linii serii stosuje się również operacje logiczne ROR, RAND, RNOT, RXOR.

Przemieszczanie figur

W celu przemieszczenia figury na rysunku stosuje się trzy następujące funkcje:

BTRANS (B, x_0, y_0) — obraz bitowy B przenoszony jest równolegle do (poziomego) kierunku X o wielkość x_0 i równolegle do (pionowego) kierunku Y o wielkość y_0;

BROTATE (B, α_0) — obraz bitowy B jest obracany, względem środka, o kąt α_0;

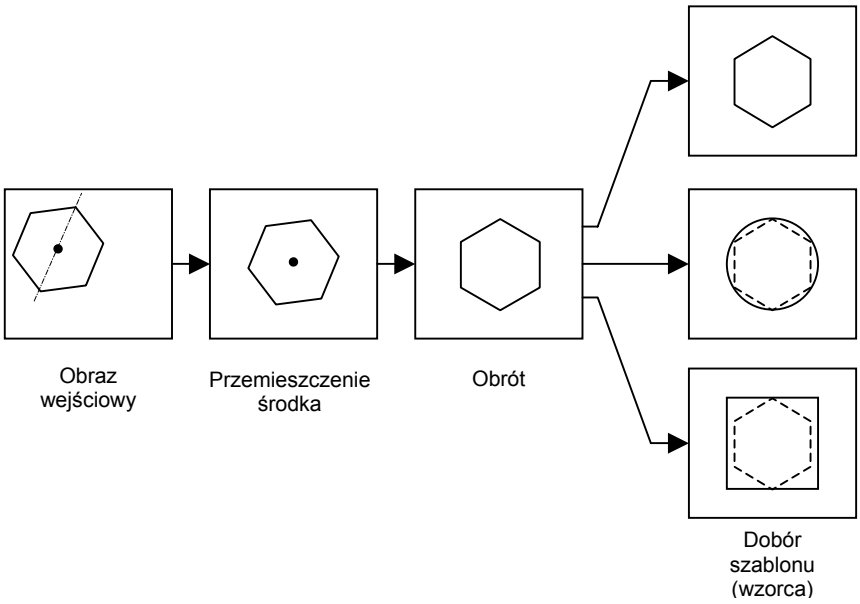
BSCALE (B, s) — obraz bitowy B jest powiększany (zmniejszany) s razy.

Dobór wzorców

Obrazy figur, które koniecznie należy rozpoznać, można stworzyć wcześniej. Takie obrazy nazywa się *wzorcami*. Aby wśród napływających obrazów stwierdzić występowanie lub brak obrazów pokrywających się z wzorcami, można bezpośrednio porównywać obrazy wejściowe ze wzorcami. Sposób ten nazywa się *doborem wzorców*. Normalizuje się wtedy obrazy według wielkości oraz wykonuje się konieczne operacje zapewniające pokrywanie się środków ciężkości i kierunków osi symetrii. Wzorzec lub obraz wejściowy przemieszcza się równolegle, obraca się, zmienia jego skalę itd. Jest to sposób dość prymitywny i nie zawsze skuteczny.

Deskryptory cech podstawowych zawierają w sobie położenie środka ciężkości figury, kierunku głównych osi bezwładności, kwadrat opisany itd. Skuteczność porównywania można istotnie ponieść wtedy, gdy podczas równoległych przemieszczeń i obrotów obrazu posłużymy się tymi deskryptorami. W przypadku pokazanym na rysunku 17 obraz wejściowy najpierw przenosi się równoległe tak, aby jego środek (ciężkości) pokrył się ze środkiem płaszczyzny rysunku. Następnie obraca się obraz tak, aby główna oś bezwładności była prostopadła, i porównuje się obraz wejściowy z trzema wzorcami. Podczas najprostszego porównywania wykorzystuje się operację wykluczania (wykluczające ALBO), która, na podstawie otrzymanej liczby niepokrywających się elementów obrazu, pozwala wnioskować o tym, jaki wzorzec nadaje się.

Aby robot mógł „obcować” z obiektem po jego rozpoznaniu za pomocą widzenia technicznego, należy określić pozycję i orientację obiektu. Podczas określania pozycji obiektu na obrazie można posługiwać się deskryptorem pozycji środka ciężkości figury, a przy określaniu orientacji – kierunkiem głównej osi bezwładności.



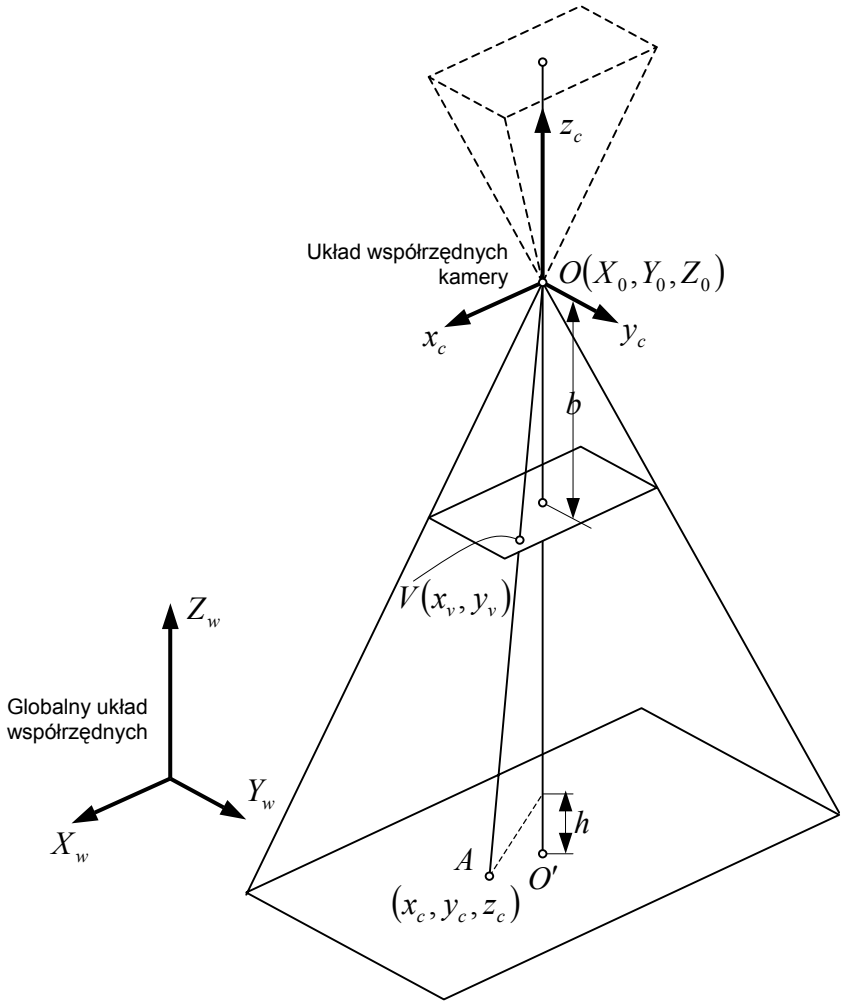
Rys. 17 Zgodność ze wzorcem

Przejście z układu współrzędnych kamery do globalnego układu współrzędnych

W systemie widzenia technicznego wyniki rozpoznania można przedstawić współrzędnymi układu prostokątnego, umieszczonego na postrzeganej płaszczyźnie. Aby robot mógł manipulować przedmiotami, potrzebne jest przejście z układu współrzędnych kamery do globalnego układu współrzędnych. Rozpatrzmy przypadek, , kiedy kamera patrzy w dół na przestrzeń roboczą (rysunek 18). Początek układu współrzędnych kamery (x_c, y_c, z_c) znajduje się w środku obiektywu a postrzegana płaszczyzna – za obiektywem. Dla ułatwienia konstrukcji geometrycznych przemieścimy symetrycznie postrzeganą płaszczyznę i umieścimy ją przed obiektywem. Ciało (punkt A), z którego formowany jest obraz na postrzeganej płaszczyźnie w punkcie $V(x_v, y_v)$, znajduje się na przedłużeniu wektora OV. Do określenia pozycji w przestrzeni trójwymiarowej nie wystarcza jedno ograniczenie. Można tu założyć, że punkt A znajduje się w płaszczyźnie umieszczonej nad powierzchnią stołu roboczego na wysokości h. W układzie współrzędnych kamery ciało ma współrzędne x_c, y_c, z_c. Załóżymy, że płaszczyzna

XY globalnego układu współrzędnych pokrywa się z powierzchnią stołu roboczego. W układzie tym początek współrzędnych kamery można przedstawić wielkościami X_0, Y_0, Z_0 a obrót względem osi Z – kątem α. Ponieważ wykorzystaliśmy założenie, że punkt A znajduje się nad stołem roboczym na wysokości h, to $z_c = -Z_0 + h$. Można więc zapisać, że

$$\begin{aligned} x_c &= x_v \frac{Z_0 - h}{b}, \\ y_c &= y_v \frac{Z_0 - h}{b}, \\ Z_c &= -Z_0 + h. \end{aligned} \tag{1}$$



Rys. 18 Przejście z układu współrzędnych kamery do globalnego układu współrzędnych

Oba układy współrzędnych można powiązać następująco:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & X_0 \\ \sin\alpha & \cos\alpha & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}. \tag{2}$$

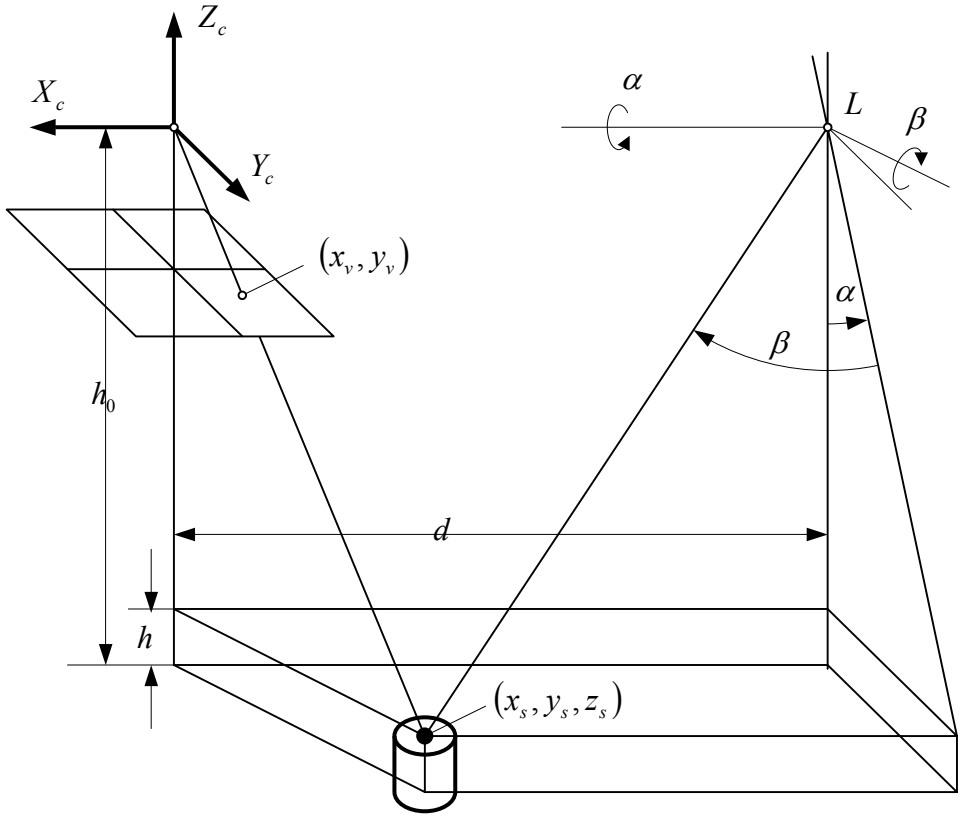
Jeżeli współrzędne x_c, y_c, z_c podstawi się we wzory (1), to obraz (x_v, y_v) na równoważnej powierzchni postrzegania w globalnym układzie współrzędnych można określić położywszy

$$x_w = x_v \frac{z_c - h}{b} + X_c,$$
$$y_w = y_v \frac{z_c - h}{b} + Y_c, \tag{3}$$
$$z_w = h.$$

Pomiar odległości

Za pomocą tylko jednej kamery telewizyjnej nie można zmierzyć pozycji przedmiotu w przestrzeni trójwymiarowej. Można to jednak zrobić za pomocą jednoczesnego wykorzystania kamery telewizyjnej i skanera laserowego. W skanerze znajdują się dwa maleńkie lustra. Zmieniając pochylenie tych luster można posłać promień laserowy w dowolnym kierunku. Za pomocą kamery telewizyjnej można mierzyć współrzędne centralne niedużej plamki, tworzonej podczas padania promienia laserowego na ciało. Jeżeli znany jest kąt padania promienia i współrzędne obrazu plamki na powierzchni postrzegania kamery, to, kierując się zasadami triangulacji, można obliczyć pozycję plamki w przestrzeni trójwymiarowej na powierzchni ciała, na które pada promień laserowy.

Aby nie komplikować obliczeń, dobrze jest umieścić skanera laserowy i kamerę telewizyjną jak na rysunku 19. Promień lasera wychodzi z punktu L i kieruje się pod kątami α i β. W układzie współrzędnych kamery punkt L znajduje się w płaszczyźnie XZ. Początek układu współrzędnych kamery i punkt L znajdują się na takiej samej wysokości h_0 od stołu roboczego. Pozycję plamki promienia laserowego w przestrzeni trójwymiarowej opisuje się współrzędnymi x_s, y_s, z_s, a na płaszczyźnie postrzegania kamery współrzędnymi x_v, y_v. Jeżeli przyjąć, że plamka znajduje się nad stołem na wysokości h, to można zapisać następujące wyrażenia:



Rys. 19 Pomiar odległości z wykorzystaniem kamery telewizyjnej i skanera laserowego

$$x_s = x_v \frac{h_0 - h}{b},$$
$$y_s = y_v \frac{h_0 - h}{b}, \tag{4-6}$$
$$Z_c = -h_0 + h.$$

Kąty pochylenia promienia laserowego α i β można określić z rysunku 6. Wtedy współrzędną X plamki można przedstawić wzorem

$$x_s = -d + \frac{h_0 - h}{\cos\alpha} \operatorname{tg}\beta . \tag{7}$$

Wykorzystując wzory (4) i (7) można ustalić, że

$$h_0 - h = \frac{db\cos\alpha}{b\operatorname{tg}\alpha - x_0\cos\alpha}. \tag{8}$$

Jeżeli wyrażenie to wstawimy do wzorów (4-6), to otrzymamy następujące współrzędne pozycji plamki w przestrzeni trójwymiarowej:

$$x_s = \frac{d\cos\alpha}{b\operatorname{tg}\alpha - x_v\cos\alpha} x_v,$$
$$y_s = \frac{d\cos\alpha}{b\operatorname{tg}\alpha - x_v\cos\alpha} y_v, \tag{9}$$
$$z_s = \frac{d\cos\alpha}{b\operatorname{tg}\alpha - x_v\cos\alpha} b.$$

Cztery funkcje i struktura systemu sensoryki wzrokowej

System sensoryki wzrokowej rozpatrywanego robota hipotetycznego wykonuje następujące zadania:

- *wylicza deskryptory* podstawowych cech za pomocą danych otrzymanych z kodowania długości serii;
- *rozpoznaje obiekty* za pomocą czynników kształtu;
- *obrabia obrazy* (mapy) bitowe, *dobiera wzorce i rozpoznaje kierunek i pozycję* ciała (obiektu) w globalnym układzie współrzędnych;
- *mierzy odległość* za pomocą telekamery i skanera laserowego.

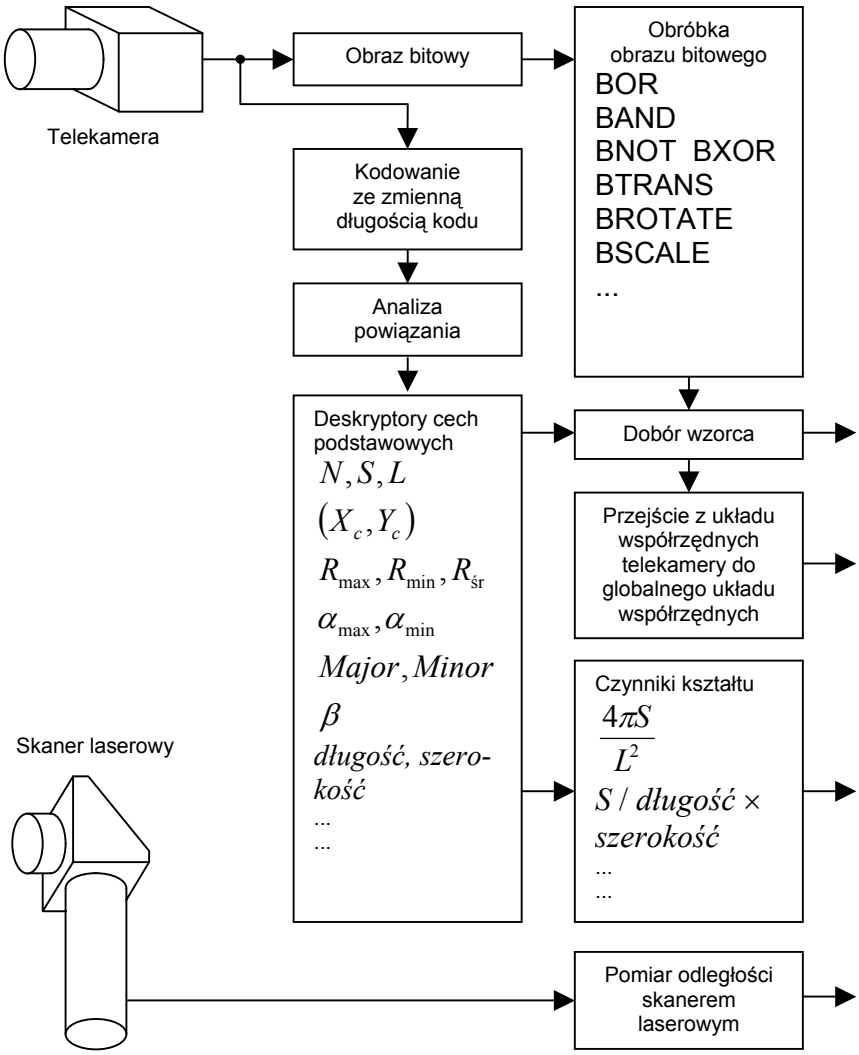
Uproszczony schemat układu widzenia technicznego pokazuje rysunek 20.

Podstawowe funkcje oprogramowania sensoryki wzrokowej można przedstawić następująco:

1. Funkcja **get_descriptors** (descriptorlist). Sygnałem wejściowym jest obraz z telekamery; na tej podstawie oblicza się wartości 14 deskryptorów cech graficznych, przytoczonych wyżej.
2. Funkcja **template_matching** (templatelist, result, x, y, φ). Wykonywane jest porównanie z szablonami (wzorcami) zamieszczonymi w funkcji templatelist. Nazwa wzorca, z którym zapewnione jest najlepsze pokrywanie się, umieszczana jest w result. W x, y, φ umieszczane są, odpowiednio, współrzędne środka i wartość kąta.
3. Funkcja **calculate_world_coord** ($x, y, \varphi, h, x_w, y_w, \alpha_w$). Współrzędne środka i kąt obiektu x, y, φ, otrzymane podczas porównywania ze wzorcem, przekształcane są w we współrzędne x_w, y_w, α_w globalnego układu współrzędnych. Przy tym zakłada się, że obiekt ma wysokość h. Wśród argumentów tej funkcji x, y, φ są wielkościami wejściowymi, h – parametrem, a x_w, y_w, α_w – wielkościami wyjściowymi.

4. Funkcja **get_spot_position** (α, β, x, y, z). Przy skierowaniu promienia laserowego pod kątami α i β obliczana jest pozycja plamki promienia w przestrzeni trójwymiarowej i znajdowane są odpowiadające współrzędne x, y, z .

Oprócz tych funkcji istnieją operacje logiczne na figurach, przenoszenie figur itd. Powodem ograniczyć się tylko do tych czterech funkcji jest stworzenie odpowiedniej równowagi podsystemu sensoryki wzrokowej w stosunku do innych podsystemów.



Rys. 20 Schemat sensoryki wzrokowej

Procesy równoległe i ich sterowanie

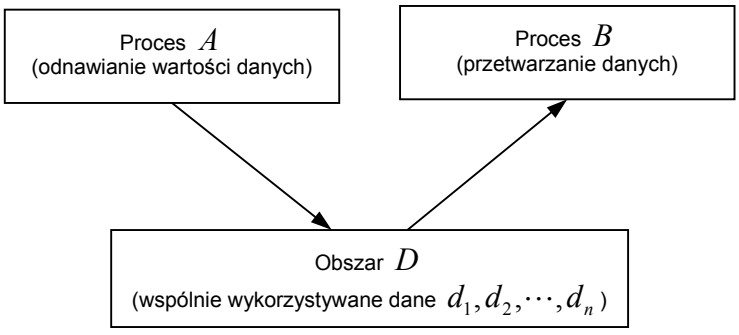
Ustaliliśmy już zbiór podstawowych zamierzonych działań, czyli funkcji robota. Podczas projektowania działań robota zbiór ten można traktować jako zbiór przyszłych programów komputerowych. Ze zbioru tych działań wybiera się proste i podstawowe elementy funkcjonalne, odpowiadające wejściowym i wyjściowym rozkazom funkcji robota. Różne funkcje użyteczne, związane ze „zmysłami" i działaniami robota, można opracowywać jako programy komputerowe, w których zestaw podstawowych zamierzonych działań wykorzystywany jest w postaci rozkazów podstawowych. Rozkazy te można włączyć, jako elementy, do zbioru funkcji robota i w ten sposób wzbogacić ten zbiór.

Zbiór elementów funkcjonalnych nie określa jeszcze systemu robotycznego. Aby zbudować system robotyczny, należy te elementy w odpowiedni sposób połączyć, uwzględniając cel ogólny. Wychodząc ze zestawu zamierzonych działań robota, należy nie tylko rozszerzać zbiór funkcji sensorycznych i aktorycznych; trzeba również mieć na uwadze środki, które mogą wiązać różne funkcje w taki sposób, aby podczas ich realizacji zapewnione było elastyczne współdziałanie tych funkcji. W robotyce jednym z interesujących zagadnień tego rodzaju jest współdziałanie aktoryki i sensoryki. Realizacja działań czuciowych (sensorycznych) i działań ruchowych (aktorycznych) powinna przebiegać równolegle. Do zbudowania systemu, w którym takie współdziałanie będzie mieć miejsce, nie wystarczy zwykłe, równoległe działanie funkcji. Trzeba widzieć możliwości istnienia więzów, które potrzebne są do odpowiedniej zgodności i synchronizacji. Podczas rozpatrywania elastycznego połączenia elementów funkcjonalnych dobrym przykładem może być system operacyjny komputera. W obszarze systemów operacyjnych wprowadza się pojęcie *procesu* i szczegółowo bada się sposoby sterowania przy równoległym zachodzeniu kilku procesów. W wykładzie tym zajmiemy się takim sterowaniem procesami równoległymi, które pozwoli zjednoczyć elementy funkcjonalne robota. Systemy operacyjne, wykorzystywane podczas funkcjonowania robota, powinny pracować w realnej skali czasu – w tak zwanym czasie rzeczywistym. Aby to zapewnić, systemy te powinny mieć wyższą szybkość działania niż typowe systemy operacyjne komputerów.

Proces przedstawia sobą pojęcie, w którego granicach wykonywany program można rozpatrywać jako czynność abstrakcyjną. Przy jednoczesnym występowaniu kilku czynności można przyjąć, że mamy do czynienia z procesami równoległymi. Jeżeli uwzględni się wzajemne powiązania procesów zachodzących równolegle, to można rozpatrywać dwa przypadki. W pierwszym procesy są wzajemnie niezależne. W drugim, dzięki synchronizacji i wymianie sygnałów, zachodzi wzajemne dopasowanie i współpraca procesów. Podczas przebiegu procesów równoległych trudno jest powiedzieć, w jakim stanie znajduje się proces rozpatrywany w stosunku do innych procesów. Jeżeli procesy równoległe są niezależne i nie oddziałują wzajemnie na siebie, to problemy specyficzne nie pojawiają się. Jednak przy równoczesnym wykorzystywaniu danych i urządzeń przez wszystkie procesy, a także podczas pracy z zapewnieniem wzajemnej synchronizacji (jeżeli rozpatrywane są procesy równoległe i wzajemnie wpływają na siebie) mogą pojawić się problemy spowodowane np. różnymi względnymi prędkościami przebiegu procesów. Typowymi problemami tego rodzaju są problemy wzajemnego wykluczenia (mutual exlusion) i synchronizacji.

Problem wzajemnego wykluczenia

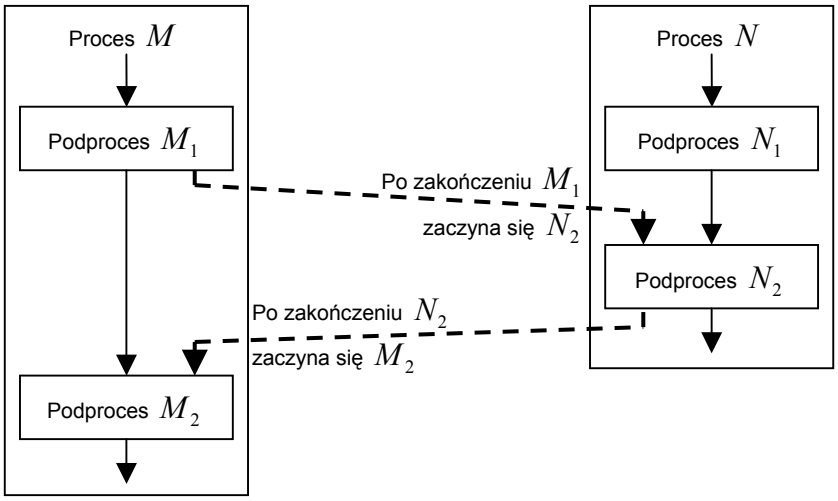
Rozpatrzmy przypadek, w którym dwa procesy, A i B, przebiegające równolegle, powinny mieć dostęp do obszaru danych D (rysunek 21). proces A odnawia treść tego obszaru, a proces B odczytuje (próbkuje, wybiera) dane z tego obszaru i wykonuje odpowiednie wyliczenia. W każdym momencie obszar danych D przedstawia sobą zbiór stanów. Jeżeli proces A odnawia obszar danych D jednocześnie z przebieganiem innego procesu B i odczytywaniem danych z obszaru D, to może pojawić się sytuacja, że otrzymywane w różnych momentach próbkowania wyniki odczytu będą pozbawione sensu. A to dlatego, że w obszarze D mogą zaistnieć zarówno dane odnowione, jak i nieodnowione. Aby do tego nie dopuścić, należy, od chwili gdy proces A zaczyna odnawiać obszar D do chwili zakończenia odnawiania, wykluczyć dostęp procesu B w obszar D. I na odwrót, przy odczytywaniu przez proces B danych z obszaru D, do zakończenia procesu odczytywania należy wykluczyć możliwość dostępu procesu A w obszar D. Taki obszar nazywa się obszarem krytycznym, a problemy tego typu – wykluczeniem wzajemnym. Aby zagwarantować wykluczenie wzajemne, należy przygotować taki mechanizm, który pozwali otwierać obszar krytyczny tylko dla jednego procesu.



Rys. 21 Wykorzystywanie obszaru danych przez kilka procesów

Problem synchronizacji

Niech dwa procesy równoległe, M i N, składają się z następujących po sobie podprocesów M_1, M_2 i N_1, N_2. Istnieje przy tym ograniczenie, zgodnie z którym podproces N_2 nie powinien zaczynać się przed zakończeniem podprocesu M_2, a podproces M_1 nie powinien zaczynać się przed zakończeniem podprocesu N_2 (rysunek 22). Poza tym podprocesy M_1 i N_1 mogą zaczynać się równolegle i niezależnie jeden od drugiego. Załóżmy teraz, że podproces N_1 zakończył się nieco wcześniej niż podproces M_1. Następujący po nim podproces N_2 nie może rozpocząć się do chwili, gdy nie pojawi się sygnał czasowy o zakończeniu podprocesu M_1. Do pojawienia się tego sygnału podproces N_2 powinien czekać, znajdować się w stanie oczekiwania. Ponadto, gdy zakończy się podproces M_1, to podproces M_2 nie może rozpocząć się do chwili zakończenia podprocesu N_2. Stoi więc przed nami zadanie synchronizacji procesów. Podczas rozwiązania tego rodzaju zadań synchronizacji procesów równoległych należy otrzymać odpowiednie informacje od mechanizmu, który steruje początkiem procesów. Wykorzystuje się do tego wymianę sygnałów czasowych.



Rys. 22 Synchronizacja procesów

Podstawowe rozkazy synchronizacji

Do rozwiązywania problemów wykluczenia wzajemnego i synchronizacji procesów równoległych stosuje się rozkazy z semaforem, tak zwane rozkazy P i V. W powszechnym rozumieniu semafor przedstawia sobą urządzenie sygnałowe ze wychylnym ramieniem, stosowane dawniej na kolei. W naszym przypadku semaforem jest ogólna zmienna całoliczbowa, stosowana w celu zapewnienia synchronizacji. Rozkazy P i V są podstawowymi rozkazami, których wartości sprawdza semafor. Przy jednym i tym samym semaforze można wykonać tylko jeden rozkaz. Jeżeli przy jakimś semaforze s zaczyna być wykonywany rozkaz P albo V, to w innym procesie rozkazy P i V, odpowiadające temu samemu semaforowi, czekają na zakończenie wykonywanego rozkazu P albo V. Rozkazy P i V, odpowiadające semaforowi s, oznacza się wielkościami $P(s)$ i $V(s)$, które można określić w następujący sposób:

- $P(s)$. Wartość semafora s jest zmniejszana o jeden. Jeżeli w rezultacie tego wielkość s stanie się ujemna, to proces wykonujący ten rozkaz zostaje zatrzymany.
- $V(s)$. Wartość semafora s jest zwiększana o jeden. Jeżeli istniał proces zatrzymany po wykonaniu rozkazu P w stosunku do s, to wybierany jest jeden z rozkazów i odnowione zostaje wykonanie procesu. Jeżeli nie ma procesów zatrzymanych, to uważa się, że wykonanie rozkazu jest zakończone.

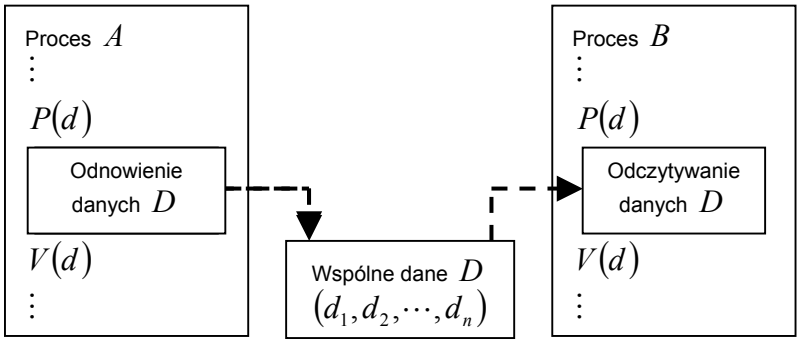
Dzięki stosowaniu rozkazów P i V można dość prosto rozwiązywać zadania wykluczenia wzajemnego i zadania synchronizacji.

Wykorzystanie w zadaniach wykluczania wzajemnego

Dla przytoczonego wcześniej przykładu z rysunku 1 rozwiążemy teraz zadanie wykluczenia wzajemnego za pomocą rozkazów P i V (rysunek 23). W tym celu dla obszaru krytycznego D wydzielimy semafor d. Procesy, przebiegające równolegle w obszarze D, mają odcinki, które są ograniczone wielkościami $P(d)$ i $V(d)$. Dla procesów A i B program może mieć postać następującą:

Proces A: ……; $P(d)$; odnowienie danych w obszarze D; $V(d)$; ……

Proces B: ……; $P(d)$; odczytywanie danych z obszaru D; $V(d)$; ……



Rys. 23 Wykluczenie wzajemne z wykorzystaniem rozkazów P i V.

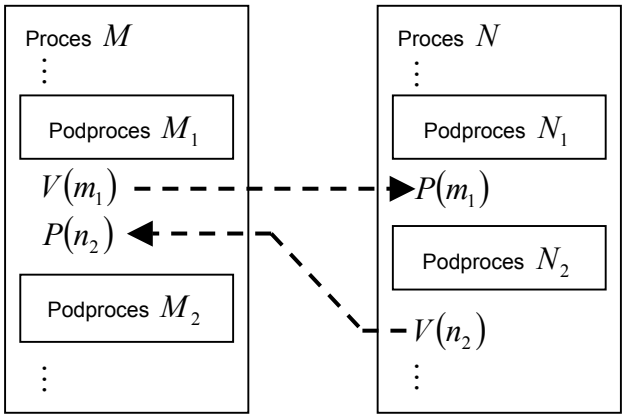
Jako początkową wartość semafora d dla obszaru krytycznego D, można przyjąć 1 i przeanalizować realizację pokazanego wyżej programu wykluczania wzajemnego. Załóżmy, że proces A wcześniej wykonał rozkaz $P(d)$. Z określenia rozkazu P wartość $d = 1-1 = 0$. Ponieważ wartość d jest nieujemna, wykonanie procesu A będzie trwać nadal z odnawianiem danych w obszarze D. Załóżmy teraz, że proces B wykonał rozkaz P. Wskutek wykonania rozkazu P pojawi się sytuacja, kiedy $d = 0-1 = -1$. Ponieważ wartość d stała się ujemna, to proces B, wykonujący rozkaz P, należy zatrzymać. Jeżeli proces A zakończy odnawianie danych i będzie wykonany rozkaz $V(d)$, to zgodnie z określeniem $d = -1+1 = 0$. W zgodzie z semaforem d proces B został zatrzymany, ale teraz można go odnowić i zacząć odczytywanie danych z obszaru D. Po zakończeniu odczytywania, gdy zostanie wykonany rozkaz $d = 1-1 = 0$, wartość $d = 0+1 = 1$. Ponieważ nie ma procesów, które, zgodnie z semaforem d, są zatrzymane, to następuje w końcu rachunku zakończenie rozkazu V bez żadnych następstw. Przed zakończeniem odczytywania danych w procesie B ponownie zostaje uaktywniony proces A i, jak można założyć, wykonywany jest rozkaz $P(d)$. W takim przypadku $d = 0-1 = -1$, to znaczy wartość d jest ujemna. Po wykonaniu rozkazu P proces A zostaje zatrzymany. Przy tym odnowienie danych odkładane jest do wykonania rozkazu V procesu B. W ten sposób, podczas wspólnego wykorzystywania obszaru krytycznego D, zapewnia się dostęp tylko jednego procesu do tego obszaru przy jednoczesnym zachodzeniu kilku procesów.

Wykorzystanie w zadaniach synchronizacji

Rozpatrzmy teraz rozwiązanie zadania synchronizacji procesów równoległych przy wykorzystaniu rozkazów semaforów (rysunek 24). Przykładem niech będzie zadanie z rysunku 2 a semaforami wielkości m_1 i n_2, które pokazują, że procesy M_1 i N_2 są zakończone. Aby zsynchronizować procesy M i N, należy wstawić następujące rozkazy P i V:

Proces M:……; podproces M_1; $V(m_1)$; $P(n_2)$; podproces M_2……

Proces N:……; podproces N_1; $P(m_1)$; podproces N_2, $V(n_2)$;……



Rys. 24 Synchronizacja procesów przez wykorzystanie rozkazów P i V .

Procesy M i N zachodzą równolegle. Podproces N_1 kończy się wcześniej niż podproces M_1, a więc wykonany jest rozkaz $P(m_1)$. Na podstawie rozkazu $P_1(m_1)$ wartość $m_1 = 0-1<0$. Podczas wykonywania rozkazu P proces N zatrzymuje się. Gdy kończy się podproces M_1 i wykonywany jest rozkaz $V(m_1)$, wartość $m_1 = -1+1=0$, to znaczy odnawiane jest wykonanie procesu, który został zatrzymany semaforem m_1. Proces M trwa nadal i wykonywany jest rozkaz $P(n_2)$. Jeżeli proces ten przebiega szybciej od procesu N, to $n_2 = 0-1=-1$ i wykonywanie procesu M zostaje zatrzymane. Po odnowieniu procesu N następuje wykonanie podprocesu N_2 i realizowany jest rozkaz $V(n_2)$. W rezultacie otrzymujemy $n_2 = -1+1=0$ i znowu zaczyna się wykonywanie procesu M, zatrzymanego semaforem n_2. Za pomocą rozkazów P i V, dotyczących semaforów m_1 i n_2, synchronizuje się w ten sposób procesy M i N, które działają równolegle i jednocześnie są wzajemnie powiązane.

Zamiast rozkazu $P(s)$ stosuje się również rozkaz $wait(s)$ (oczekiwanie), a zamiast rozkazu $V(s)$ – rozkaz $signal(s)$ (sygnał). W celu ułatwienia zapisu programu będziemy dalej stosować rozkazy $wait(s)$ i $signal(s)$.

Program sterujący

Program sterujący (ang. *supervisor*, program nadzorczy, monitor) zapewnia takie warunki przebiegu procesów równoległych, że, przy obecności ograniczonych środków obliczeniowych (procesora centralnego, obszaru danych, sprzętu) powstaje wrażenie jednoczesnego przebiegu kilku procesów. Nawet wtedy, gdy procesy równoległe przebiegają z synchronizacją i wzajemnym wykluczaniem, nie wpływają one niepożądanie na siebie i powstaje wrażenie, że procesy w rzeczywistości przebiegają równolegle.

Odstępy czasowe (interwały czasowe)

Aby zrealizować fikcyjne równoległe zachodzenie procesów, program sterujący rozbija czas procesora centralnego na małe odstępy czasu. Odstępy te nazywa się nazywa się *interwałami*, *lub kwantami czasowymi*. Program sterujący rozdziela wykonanie procesów na pojedyncze interwały. Czas pracy procesora centralnego jest, przez program sterujący, rozdzielany parytetowo (w odpowiednim udziale) na wiele procesów. W trakcie wykorzystywania interwału przez jakiś proces wykonywanie tego procesu zostaje przerywane, gdy wydzielany jest kolejny interwał jest dla innego procesu, mimo że działania, związane z przebiegiem poprzedniego procesu nie zostały jeszcze zakończone. Pozwala to zrealizować fikcyjne, równoległe działania procesów.

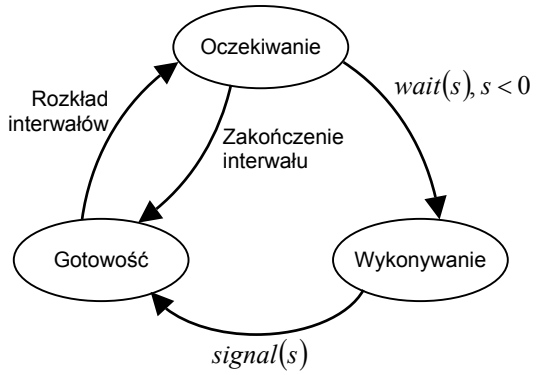
Przejście procesu z jednego stanu w drugi

Działania, związane z przebiegiem procesu, określane są programem. Proces można stworzyć wtedy, gdy w pamięci komputera umieścimy odpowiedni program. Samo tako umieszczenie oznacza jednak, że proces znajduje się w stanie nieaktywnym. Kiedy, zgodnie z programem sterującym, zaczyna występować wymaganie wykonania procesu, to następuje jego „obudzenie" i przejście do stanu aktywnego. „Obudzony" proces może przyjmować jeden z następujących stanów:

ready (gotowość do wykonania);
running (wykonywanie), oraz
waiting (oczekiwanie).

Program sterujący kontroluje obecność tych stanów w różnych procesach i przez to realizuje sterowanie procesami równoległymi.

Przejście procesu z jednego stanu w drugi pokazuje rysunek 25. Jeżeli należy wykonać proces, to program sterujący rejestruje go w stanie ready (gotowość do wykonania). Zwykle w tym stanie rejestrowanych jest kilka procesów. Program sterujący rozdziela po równo interwały między te procesy i powoduje wykonywanie każdego z procesów. Gdy interwał zostanie procesowi wydzielony, to przechodzi on w stan running (wykonywanie). Kod, określający proces, może być realizowany za pomocą procesora centralnego. Podczas wykonywania procesu podawany jest rozkaz $wait(s)$ (czekaj). Zadana wartość zmiennej s jest wtedy zmniejszana o jeden. Jeżeli wynik staje się ujemny, to rozpatrywany proces przechodzi w stan waiting (oczekiwanie) i przebieg procesu jest zatrzymywany. Kiedy proces, znajdujący się w stanie wykonywania, kończy interwał dozwolony przez program sterujący, wykonanie procesu jest przerywane i przechodzi on w stan ready. Jeżeli w stanie running zakończona zostanie realizacja kodu określającego proces, to proces przechodzi w stan nieaktywny. Innymi słowy, wróci on do stanu, w którym jest tylko przechowywany w pamięci i gdzie będzie znajdować się dopóty, dopóki nie pojawi się wymaganie wykonania zgodne z rozkazem startprocess. Przebieg procesu znajdującego się w stanie waiting jest zatrzymywany, gdy z innego procesu, znajdującego się w stanie wykonywania, podany zostanie rozkaz $signal(s)$. Zatrzymany proces przechodzi znowu w stan ready. Opis powyższy dotyczy tylko jednego procesu. Rozpatrzone są tylko te powiązania, które mają miejsce podczas przejścia z jednego stanu w drugi.



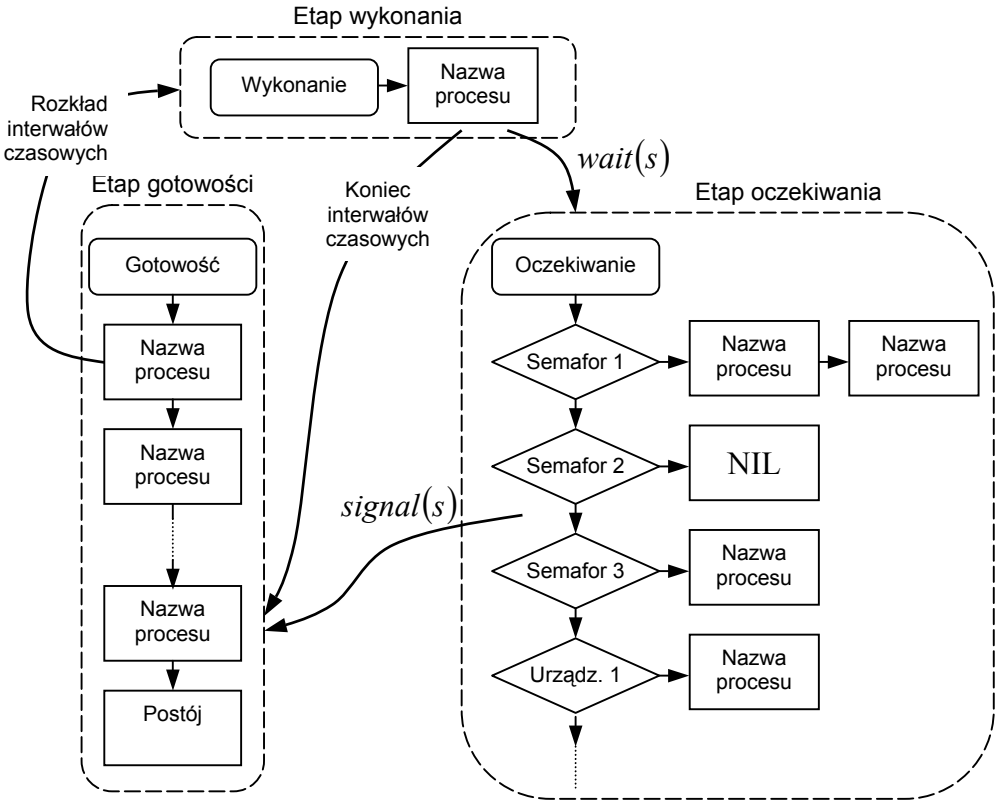
Rys. 25 Przejście procesu z jednego stanu w drugi

Struktura danych dla sterowania stanami

Każdy z procesów przebiegających równolegle znajduje się w dowolnym z trzech stanów przedstawionych wyżej. Gdy wykorzystywany jest tylko jeden procesor centralny, w stanie wykonywania znajduje się tylko jeden proces. Inne procesy mogą znajdować się w stanie ready lub w stanie waiting. Aby program sterujący mógł rozróżniać oddzielne procesy, każdy proces ma przypisaną nazwę. Najprostszym elementem wskazującym, że kilka procesów znajduje się w stanie ready, jest macierz oczekiwania. Przeprowadzanie procesów prze program sterujący w stan ready oznacza, że one ustawiają się w kolejkę ready. Rozpatrzmy kolejkę typu FIFO (First In First Out, pierwszy przyszedł – pierwszy obsłużony) ustaloną dla stanu ready. Program sterujący ustawia nazwy procesów w kolejkę FIFO. Nazwa procesu stojącego na przedzie kolejki FIFO przechodzi w stan running. Na końcu kolejki FIFO znajduje się stan idle (stan jałowy, przestój – swego rodzaju oczekiwanie). W przypadku, gdy nie

ma procesów mających stan, istnieje proces fikcyjny, przeznaczony do tego, aby przeprowadzić procesor centralny w stan idle.

Kolejkę można rozpatrywać jak model przedstawiający kilka procesów w stanie waiting. Proces w stanie waiting ma sygnał warunkowy, zadawany przerwaniem zewnętrznym, oraz rozkazowy $signal(s)$, który może istnieć wewnątrz innego procesu. Zmienną, rozpoznającą ten sygnał, jest semafor. Można przyjąć, że semafor jest warunkiem oczekiwania. Dla każdego warunku można rozpatrzyć kolejkę FIFO i ustawić procesy w tej kolejce w porządku oczekiwania według warunków. Rysunek 26 pokazuje strukturę danych zapewniającą powiązane sterowanie przejściami stanu procesów równoległych. Prostokąty ukazują nazwy procesów. W prostokącie związanym z semaforem 2 NIL oznacza, że nie istnieje proces, który oczekuje przy tym semaforze. NIL jest szczególną wartością wskaźnika, która nie wskazuje na żaden dalszy proces.



Rys. 26 Struktura danych podczas sterowania przejściami stanu procesów

Jądro programu

Centralną częścią programu sterującego jest jądro. Jest to program, który zawiera podstawowe funkcje do sterowania procesami równoległymi (przejście stanów procesu, sterowanie kolejnością wykonania procesów, wykluczanie wzajemne, podstawowe rozkazy synchronizacji itd.). Pracę jądra można organizować tylko trzema środkami: (1) przerwaniem według interwałów czasowych, (2) zakleszczeniem (przerwaniem przy pojawieniu się sytuacji nieprzewidzianej) i (3) przerwaniem sygnałem zewnętrznym.

Przerwanie według interwałów czasowych. Aby kilka procesów przebiegało jakby równolegle, czas pracy procesora centralnego rozbija się na małe interwały, które przypisuje się kolejno do procesów. W ten sposób zapewnia się parytetowe wykorzystywanie procesora centralnego. Przerwania następują w stałych odcinkach czasu, według zegara. W efekcie jądro przerywa wykonywany proces i daje możliwość innemu procesowi wykorzystać procesor centralny. Przełączenie z wykonywania jednego procesu na wykonywanie drugiego odbywa się według poniższego algorytmu.

Algorytm 1 (przerwanie według interwałów czasowych)

1. Wprowadza się zakaz pojawiania się innych możliwych przerwań.

2. Zachowuje się informację o stanie procesu bieżącego, którego nazwę posyła się na koniec kolejki FIFO oczekiwania na wykonanie.
3. Nazwa procesu, pierwszego w kolejce FIFO, przenoszona jest w stan wykonywania.
4. Odnawiana jest informacja o stanie procesu oczekującego, przeprowadzonego w stan running. Odwołuje się zakaz przerwania. Zaczyna się wykonywanie tego nowego procesu.

Zakleszczenie. Zakleszczenie (deadlock) jest przerwaniem podczas pojawienia się sytuacji nieprzewidzianej. Jest ono uwarunkowane oprogramowaniem, to znaczy przerwaniem systemowym. Można przyjąć, że zakleszczenie pojawia się w rozkazach wykorzystujących funkcje programu sterującego, to znaczy podczas odwoływania się do programu sterującego.

Algorytm 2 (zakleszczenie)

1. Wprowadza się zakaz przerwania.
2. Zachowuje się informację o stanie procesu bieżącego.
3. Przywołuje się program sterujący, podwodujący pojawienie się zakleszczenia.
4. Odnawiana jest informacja o stanie procesu przerwanego zgodnie z pkt. 2. Odwołuje się zakaz przerwania. Odnawiane jest wykonywanie procesu przerwanego.

Rozpatrzymy teraz pięć rodzajów odwoływania się do programu sterującego. Za ich pomocą organizuje się wykonanie procesów równoległych. Stosuje się tu następujące odwołania:

startprocess (p)	Proces p umieszczany jest na końcu kolejki FIFO oczekiwania na wykonanie;
stopprocess (p)	Proces p przymusowo wraca w stan nieaktywny;
signal (s)	Rozkaz V odnoszący się do semafora s ;
wait (s)	Rozkaz P odnoszący się do semafora s ;
startIO $c(a)$	Zapewniany jest początek wykonywania operacji wejścia-wyjścia (rozkaz c , spis argumentów a).

Odwołanie się do programu sterującego startIO, po przekazaniu spisu argumentów i rozkazów odpowiedniemu procesorowi wejścia-wyjścia, znajduje się w stanie oczekiwania do chwili, kiedy będzie otrzymany sygnał zakończenia tego działania. Aby przy tym czas procesora centralnego nie był jałowy, proces, przy którym pojawiło się odwołanie do programu sterującego, czeka podczas posłania rozkazu, dopóki w kolejce FIFO nie zakończy się wejście-wyjście (c jest tu identyfikatorem). Wiadomość o zakończeniu wejścia-wyjścia jest posyłana do programu sterującego za pomocą przerywania zewnętrznego, podczas którego można posługiwać się procedurą jak niżej.

Przerwanie zewnętrzne. Po zakończeniu wejścia-wyjścia, którego początkiem był rozkaz startIO, powstaje przerwanie zewnętrzne. W wyniku tego przerywania znowu odnawiane jest wykonywanie procesu, który był przerwany przez pojawienie się rozkazu startIO. Przetwarzanie, odpowiadające przerwaniu zewnętrznemu, schematycznie można przedstawić następującym algorytmem.

Algorytm 3 (przetwarzanie przy przerwaniu zewnętrznym)

1. Wprowadza się zakaz przerywania.
2. Zachowuje się informację o stanie procesu bieżącego.
3. Przy identyfikatorze operatora wejścia-wyjścia, przy którym pojawiło się przerywanie, zatrzymany proces przechodzi na koniec kolejki FIFO oczekiwania na wykonanie.
4. Odnawiana jest informacja o stanie przerwanego procesu, zgodnie z pkt. 2, i zniesiony zostaje zakaz przerywania.

Aktywacja zestawu funkcji

Zestaw funkcji odnoszących się do „zmysłów” i działań robota można uruchamiać specjalnymi centralnymi procesorami różnych podsystemów: sensoryki wizyjnej, sensoryki taktylnej (dotykowej) i sensoryki siłowej. Przez funkcje należy tu rozumieć rozkazy, które pozwalają zrealizować zamierzone działania robota, odpowiadające wyżej wymienionym podsystemom. Zestawowi funkcji – z punktu widzenia programu sterującego procesami równoległymi – przypada rola makrorozkazów wejścia-wyjścia

dla podsystemów. Każda funkcja uruchamiana jest przez odwołanie się do programu sterującego w postaci startIO, to znaczy startIO function (argument). Przy takim odwołaniu się do programu sterującego argumenty i rozkaz uruchomienia funkcji sterującej przekazywane są do podsystemu. Proces, w którym pojawią się te instrukcje, przechodzi w stan waiting (oczekiwanie) i zapewnia się oczekiwanie na zakończenie wykonywania funkcji. Gdy w podsystemie zakończy się obróbka wskazanej funkcji, pojawia się przerwanie zewnętrzne dotyczące programu sterującego. Po otrzymaniu tego przerwania program sterujący przeprowadza proces, znajdujący się w stanie waiting, w stan ready.

Interwały czasu od rozpoczęcia do zakończenia wykonywania funkcji są zwykle różne dla różnych funkcji. I tak, dla funkcji put_arm_reference zadawaną wartość przemieszczenia ręki wprowadza się tylko w rejestr ustawienia podsystemu. Wykonanie tej funkcji odbywa się w bardzo krótkim interwale czasu. Natomiast podczas wykonywania funkcji template_matching, z którą mamy do czynienia w sensoryce wizyjnej, od chwili uruchomienia systemu należy wprowadzić dane obrazu, wykonać porównanie ze wzorcami, przeprowadzić odpowiednie obliczenia, związane z rodzajem wzorca (z którym następuje pokrycie), jego współrzędnymi i kątem. Dopiero po tym pojawia się końcowe przerwanie dla programu sterującego. Czasy trwania obróbki w tych dwóch przykładach różnią się znacznie. W przypadku funkcji put_arm_reference, po odwołaniu się rozkazu startIO do programu sterującego, jednorazowe przeprowadzenie w stan waiting jest zbyteczne. System można uprościć kosztem organizacji sterowania za pomocą sposobu, podobnego do sposobów wykorzystywanych dla innych funkcji.

Niektórym procesom, działającym równolegle, stawia się wymaganie, aby jedne i te same funkcje były wykonywane jakby równocześnie. W takim przypadku należy organizować wzajemne wykluczanie podczas uruchamiania funkcji wykorzystujących jedne i te same urządzenia wejścia-wyjścia. W robocie hipotetycznym mamy pięć niezależnych urządzeń wejścia-wyjścia: rękę, kiść (dłoń), sensory dotyku, sensory siły i sensory optyczne. Jeżeli nazwy tych urządzeń uczyni się semaforami i wykorzysta się odwołania signal i wait, to łatwo można zrealizować wykluczenie wzajemne. Na przykład dla funkcji put_arm_reference nazwę urządzenia arm można wykorzystać jako semafor i zrealizować wzajemne wykluczenie w następującym szeregu odwołań do programu sterującego:

wait (arm)

start IO put_arm_reference (X_r, Y_r, Z_r, α_r)

signal (arm)

Jeżeli we wszystkich miejscach, gdzie wykorzystywana jest funkcja put_arm_reference, zrobi się odpowiednią „ochronę" za pomocą wait (arm) i signal (arm), to takie urządzenie, jak arm, jest w stanie wykorzystywać jednocześnie tylko jeden proces. Ffunkcja wait (arm) zablokuje możliwość powtórnego wykorzystania funkcji put_arm_reference przez inny proces; funkcja signal (arm) usunie zaś tę blokadę). W następnym wykładzie pokażemy przykład programu wzajemnego oddziaływania różnych sensorów. Aby przedstawić istotę programu w postaci uproszczonej, należy z rozpatrywania wykluczyć opis pokazanego wyżej wykluczenia wzajemnego. W warunkach rzeczywistych, w celu zapewnienia bezwarunkowego wykonania programu, przeprowadzenie wykluczenia wzajemnego jest swego rodzaju działaniem ochronnym.

W programach, związanych z sensoryką i działaniami robotów, należy koniecznie uwzględnić warunek, żeby prędkość ruchu programu odpowiadała czasowi rzeczywistemu. Dlatego dodaje się odwołanie do programu sterującego, nazywane delay (time). Odwołanie to, jak sama nazwa wskazuje (delay *ang.* opóźnienie) zatrzymuje wykonywanie programu tylko na czas ukazany symbolem time. Przy pojawieniu się tego odwołania program sterujący przeprowadza proces, znajdujący się w stanie wykonywania, w stan waiting. Po upływie zadanego czasu następuje przerwanie według zegara i rozpatrywany proces przechodzi ze stanu waiting w stan ready. W następnym wykładzie pokażemy wykorzystanie odwołania delay do sterowania dynamiką odnawiania działań zadawanych ręce i dłoni.

Sterowanie torem ruchu (trajektorią)

Rozpatrzyliśmy zbiór podstawowych funkcji robotów oraz program sterujący, który zapewnia wzajemne powiązanie tych funkcji i ich sterowanie. Teraz pokażemy kilka typowych operacji i zrealizowanie w konkretnych warunkach takiego programowania, które pozwoli zjednoczyć czucie (sensorykę) i działania (aktorykę) robota.

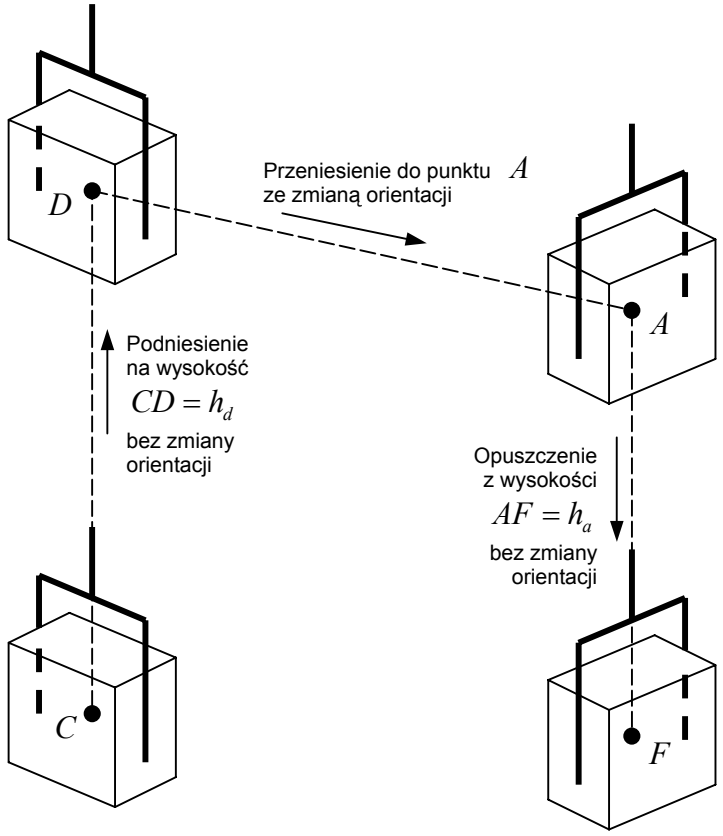
Ruch ręki robota realizowany jest w przestrzeni roboczej po zadanej trajektorii łamanej. Połączmy dwa punkty przestrzeni roboczej linią prostą i przeanalizujmy metodę interpolacyjną, która zapewnia ruch wzdłuż tej trajektorii. Przyjmijmy, że znane jest położenie (pozycja + orientacja) zarówno w punkcie wyjściowym $C(x_c, y_c, z_c, \alpha_c)$, jak i w punkcie końcowym R, do którego realizowane jest przemiesz-

czenie $(x_r, y_r, z_r, \alpha_r)$. Jeżeli założymy, że robot przemieszcza się z prędkością v, to dzieląc odległość między punktami C i R przez tę prędkość otrzymamy czas, potrzebny na to przemieszczenie. Czas ten wygodnie jest przedstawić w postaci iloczynu NT_0, gdzie T_0 jest jednostką czasu. Przyrosty pozycji i orientacji w jednostce czasu można przedstawić następująco:

$$dx = \frac{x_r - x_c}{N}, \quad dy = \frac{y_r - y_c}{N}, \quad dz = \frac{z_r - z_c}{N}, \quad d\alpha = \frac{\alpha_r - \alpha_c}{N}.$$

Można więc przyjąć, że zadawane wartości x, y, z, α doznają w jednostce czasu przyrostów $dx, dy, dz, d\alpha$ i zachodzi przy tym wykonanie funkcji put_arm_reference. Program przemieszczania się robota po linii prostej wygląda w języku Pascal następująco:

```
process move_direct ( x_r , y_r , z_r , α_r );
begin
        {Wyliczane są wartości  N, dx, dy, dz, dα }
        for I : = 1 to  N do begin
                x_c := x_c + dx;  y_c := y_c + dy; z_c := z_c + dz;   α_c := α_c + dα ;
                startIO put_arm_reference ( x_c , y_c , z_c , α_c ) ;
                delay ( T_0 ) ;
        end ;
end;
```



Rys. 27 Ruch obiektu przez punkty odprawienia D i przybycia A

Rysunek 27 pokazuje przykład typowej trajektorii. Robot chwyta obiekt w punkcie C i przenosi go do punktu F. Podczas podnoszenia obiektu do punktu D orientacja obiektu nie zmienia się a podnoszenie odbywa się po prostej. Również podczas opuszczania obiektu w okolicach stołu roboczego dobrze jest nie zmieniać orientacji, aby zmniejszyć prawdopodobieństwo styku z innymi obiektami. Zwykle punkt D nosi nazwę *punktu odprawienia*, a punkt A – *punktu przybycia*. W punkcie F pozy-

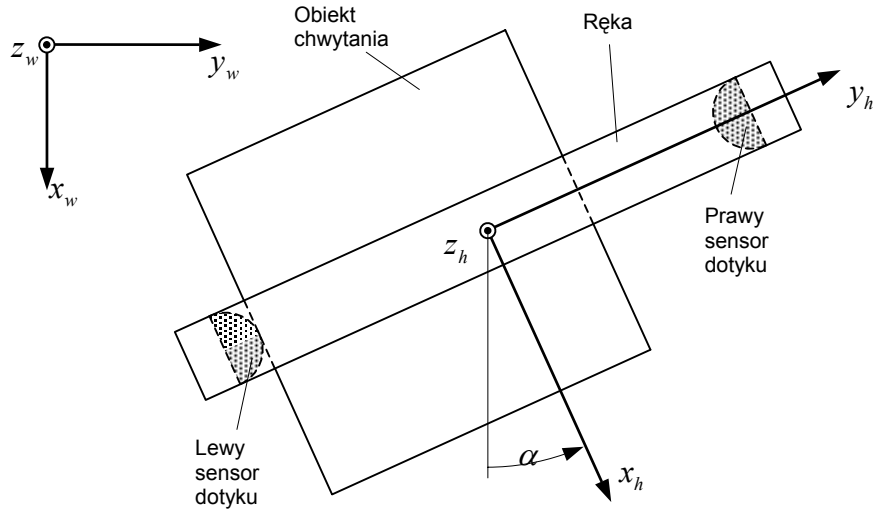
cja i orientacja opisane są parametrami x_f, y_f, z_f, α_f. Oprócz tego istnieją parametry h_d i h_a. Pierwszy jest odległością odprawienia, drugi odległością przybycia. Program, opisujący trajektorię z rysunku 27, można zapisać następująco:

process move_deproach (x_f, y_f, z_f, α_f) ;
begin
move_direct ($x_c, y_c, z_c + h_d, \alpha_c$) ;
move_direct ($x_f, y_f, z_f + h_a, \alpha_f$) ;
move_direct (x_f, y_f, z_f, α_f) ;
end ;

Chwytanie obiektu z wykorzystaniem sensoryki dotykowej

W robocie hipotetycznym chwytanie obiektu odbywa się jest przez jednoczesne zamykanie się dwóch palców. Chwytanie powinno być płynne, aby robot nie przesuwał obiektu. Jeżeli zapewnimy dokładne pozycjonowanie robota i nauczymy go dokładnego trafiania w miejsce, gdzie obiekt się znajduje, to robot chwyci obiekt poprawnie, nie przesuwając go. Brak odpowiedniej dokładności chwytania, obojętnie z jakiej przyczyny, może spowodować przemieszczenie obiektu. Idealnym jest taki proces chwytania, w którym centralne położenia palców pokrywają się z centralną pozycją obiektu. Chwytanie takie można nazwać *centrowanym*. Aby je zrealizować, niezbędna jest sensoryka dotykowa, zwana w terminologii międzynarodowej taktylną (łac. *tactilis* = dotykalny). W rozpatrywanym robocie sensoryka taktylna (czujniki dotyku) istnieje na wewnętrznych stronach obu palców. Zapoznajmy się więc z programem chwytania centrowanego.

Chwytanie centrowane można realizować za pomocą dwóch równolegle przebiegających procesów. W pierwszym następuje zachodzi powolne zbliżenie palców. Ponieważ podczas zbliżania palców zawsze jeden z nich dotknie obiektu chwytanego jako pierwszy, to w chwili dotknięcia następuje oddalenie całej ręki od obiektu chwytanego. I to jest drugi proces. Innymi słowy ręka robota dopasowuje swoją pozycję do pozycji obiektu i przez to centruje chwytak (rysunek 28).



Rys. 28 Adaptacja położenia ręki podczas centrowania

Rozpatrzmy przypadek kiedy styk następuje tylko lewym palcem, jak na rysunku 2. Aby podczas stykania obiekt nie został przesunięty ze swej pierwotnej pozycji, należy zmienić pozycję ręki w kierunku y_h. Ale o ile? Jeżeli odległość między palcami skraca się w jednostce czasu o 2δ, to pozycję ręki w kierunku y_h należy zmienić o δ. Ponieważ rozkaz do takiego działania przedstawia się w globalnym układzie współrzędnych (x_w, y_w, z_w) a orientacja scharakteryzowana jest parametrem α, oznacza to, że w kierunku x_w należy zrealizować przemieszczenie o $\delta \sin\alpha$ zaś w kierunku y_w przemieszczenie o $\delta \cos\alpha$. Dopasowując odpowiednim sposobem te dwa procesy można zacisnąć palce na

obiekcie nie przesuwając go. Obiekt jest uchwycony wtedy, gdy na obu palcach zadziałają sensory dotyku. Teraz można przyłożyć odpowiednią siłę, aby obiekt uchwycić wystarczająco silnie do przeniesienia go. Taki właśnie program można wykorzystać do realizacji działania centrującego obiekt w chwytaku. Obie operacje, closing (zbliżanie) i adapting (dopasowywanie), dwóch równolegle przebiegających procesów kończą się wtedy, gdy zadziałają czujniki dotyku na obu palcach. Do oceny szerokości rozwarcia palców W_r przyjmuje się kryterium $W_r < \varepsilon$, gdzie ε jest wartością progową, przy której palce zamkną się i zadziałają oba czujniki dotyku.

```
process center (grasp_force) ;
begin
        startIO get_touch_sensor ( L, R ) ;
        cobegin closing ; adapting ; coend ;
        if W_r < ε then error ;
        startIO put_grasp_force ;
end ;
process closing ;
begin
        while ( R = 0 ) or ( L = 0 ) do begin
                W_r := W_r − δ ; startIO put_hand_opening ( W_r ) ; delay ( T_0 ) ;
                startIO get_touch_sensor ( L, R ) ;
        end ;
end ;
process adapting;
begin
        while ( R = 0 ) or ( L = 0 ) do
                startIO get_touch_sensor ( L, R ) ;
                if ( R = 1 ) and ( L = 0 ) then begin
                        X_r := X_r + δ * sin(α); Y_r := Y_r − δ * cos(α); end ;
                else if ( R = 0 ) and ( L = 1 ) then begin
                        X_r := X_r − δ * sin(α); Y_r := Y_r + δ * cos(α); end ;
                startIO put_arm_reference ( X_r, Y_r, Z_r, α_r ) ; delay ( T_0 ) ;
        end ;
end ;
```

Umieszczanie obiektu z wykorzystaniem sensoryki siłowej

Gdy człowiek umieszcza (kładzie) jeden obiekt na drugi, chwyta go zwykle ręką, podnosi i przenosi do miejsca, gdzie należy obiekt umieścić, a potem uwalnia go od ręki (albo rękę od obiektu). Nie będziemy tu rozpatrywać nakładania jednego obiektu na drugi. Przyjrzymy się tylko temu, jak robot nie pozwala, aby obiekt upadł swobodnie w dół po uwolnieniu zacisku palców chwytaka. Innymi słowy chodzi o to, jak robot ostrożnie kładzie czy stawia przedmiot. Aby robot to zrobił, niezbędna jest sensoryka siły reakcji podłoża na przedmiot stawiany. Sensoryka ta pozwala ustalić obecność styku obiektu z podłożem lub innym obiektem.

Działanie robota, oparte na wykorzystaniu sensora siły reakcji, gdy jeden obiekt kładziemy ostrożnie na drugi, nosi nazwę **puton**. Rozpatrzmy program tego działania. Najpierw należy zmierzyć siłę F_{z0}, działającą w nadgarstku w osi z. Można założyć, że początkowa wartość tej siły zależy od ciężaru trzymanego obiektu. Mierząc ciągle siłę w osi z należy powoli opuszczać rękę (dłoń) o wartość δ. Jeżeli siła reakcji f_z, mierzona w osi z, osiągnie progową wartość ε, to można przyjąć, że uchwycony obiekt zetknął się z podłożem (innym obiektem). Innymi słowy, przy $F_{z0} - f_z \geq \varepsilon$ można uważać, że trzymany w ręce obiekt jest ostrożnie postawiony na wymaganym miejscu. Wykorzystując odpowiednią informację o sile reakcji i upewniwszy się, że obiekt jest postawiony poprawnie, należy teraz rozsunąć palce i uwolnić trzymany obiekt. Program w języku PASCAL można zapisać następująco:

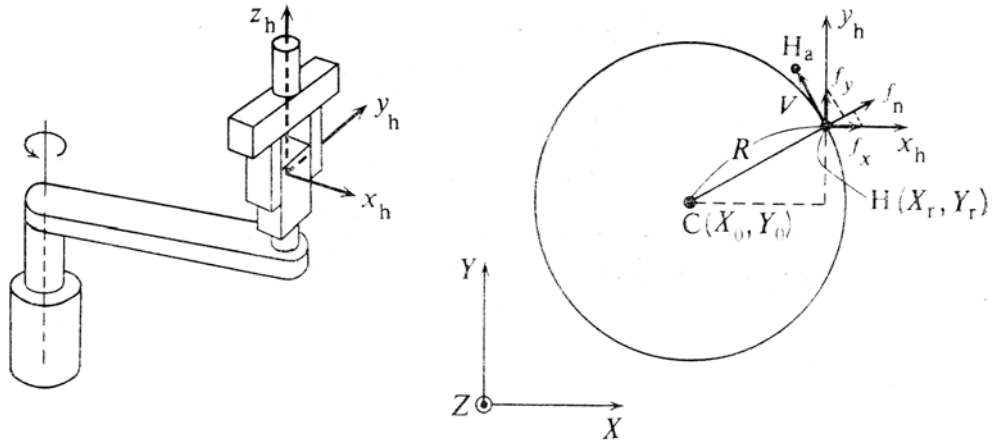
```
process puton ;
```

```
begin
        startIO get_wrist_force (f_x, f_y, f_z, m_x, m_y, m_z) ;
        F_z0 := f_z ;
        while (F_z0 − f_z) < ε do
                Z_r := Z_r − δ ;
                startIO put_arm_reference (X_r, Y_r, Z_r, α_r) ; delay (T_0) ;
                startIO get_wrist_force (f_x, f_y, f_z, m_x, m_y, m_z) ;
        end ;
                W_r := full_opening ;
                startIO put_hand_opening (full_opening) ; delay (T_0) ;
        end ;
```

Obracanie korbą z wykorzystaniem sensoryki siłowej

Dla człowieka obracanie korbą jest operacją bardzo prostą, dla robota – jedną z najtrudniejszych. Korba (dźwignia) porusza się po torze kołowym zadanym w przestrzeni. Ten tor kołowy jest dużym ograniczeniem i zapewnienie ruchu korby problemem dość złożonym. Dzisiejsze roboty mogą prawie dokładnie opisywać tory kołowe (okręgi). Jeżeli jednak okrąg, po którym następuje ruch korby, choćby tylko niewiele odbiega od okręgu zadanego robotowi, to współpraca robota z obracającą się korbą może zakończyć się połamaniem korby lub samego robota. Chodzi o to, aby podczas pracy nie pojawiały się nadmierne siły a obrót korby następował właściwie, stosując się do ograniczeń narzuconych przez korbę. Ważne jest określanie siły reakcji, działającej na rękę w procesie pracy, oraz sterowanie ruchem ręki w odpowiedni sposób. Aby więc zapewnić ruch ręki robota wzdłuż trajektorii, na którą z zewnątrz nałożone są ograniczenia fizyczne, należy stosować pozycjonowanie z siłowym sprzężeniem zwrotnym.

Załóżmy, że mamy korbę o promieniu R ze współrzędnymi środka X_0, Y_0, Z_0 (rysunek 29). Należy zbudować program, zgodnie z którym korba będzie obracana przez robot ze stałą prędkością. Obrót korby można zrealizować przez dopasowanie dwóch procesów. Pierwszy jest siłowym oddziaływaniem na korbę po stycznej do okręgu opisywanego przez korbę; drugi proces polega na zrealizowaniu siłowego sprzężenia zwrotnego, podczas którego reakcja w kierunku normalnym przechodzi w zero.



Rys. 29 Obracanie korbą w płaszczyźnie

Warunek stałości prędkości w kierunku stycznym do okręgu można zamienić na odpowiednie pozycjonowanie, podczas którego w każdym interwale czasu punkt, przemieszczający się o stałą odległość w kierunku stycznym, rozpatrywany jest jako (fikcyjny) cel czasowy. Załóżmy, że korba znajduje się w punkcie $H(X_r, Y_r, Z_r, \alpha_r)$. Jako położenie celu, odpowiadające upływowi jednostki czasu, można przyjąć punkt H_a, odpowiadający przemieszczeniu po stycznej na odległość V, która jest określana z wartości prędkości obrotu. Składowe wektora $\boldsymbol{V}$ na osi X i Y, można przedstawić następująco:

$$V_{\mathrm{X}} = -V\frac{Y_r - Y_0}{R},$$
$$V_{\mathrm{Y}} = V\frac{X_r - X_0}{R}.$$

Aby przemieścić rękę robota do punktu H_a, należy otrzymać nowe oddziaływanie wielkości zadawanej. Odnawianie tego oddziaływania można dokonywać na podstawie wzorów:

$$X_r \leftarrow X_r - V\frac{Y_r - Y_0}{R},$$
$$Y_r \leftarrow Y_r + V\frac{X_r - X_0}{R}.$$

Oddziaływanie to nie musi pokrywać się z okręgiem, po którym obraca się korba. Posłużymy się bowiem sensorem siły, wbudowanym w nadgarstek ręki, oraz zrealizujemy równanie w taki sposób, aby siła reakcji w kierunku normalnym sprowadzała się do zera. Efekt: dokonamy odpowiedniej adaptację położenia ręki do kołowej trajektorii korby

Załóżmy, że kąt orientacji dłoni (chwytaka) $\alpha = 0$. Jeżeli wyjdziemy teraz z wartości składowych sił f_x i f_y, mierzonych sensorem siły w układzie współrzędnych dłoni, to siłę reakcji w kierunku normalnym można przedstawić następująco:

$$f_{\mathrm{n}} = f_x\frac{X_r - X_0}{R} + f_y\frac{Y_r - Y_0}{R}.$$

Aby siła reakcji sprowadzała się do zera, rękę należy przemieszczać w tym kierunku, w którym ta siła działa. Wygodnie jest tu posługiwać się współczynnikiem proporcjonalności K i skorygować zadawane oddziaływanie ręki o wielkość Kf_n. W efekcie oddziaływanie zadawane w celu adaptacji ręki można odnawiać przez wykorzystanie wzorów:

$$X_r \leftarrow X_r + Kf_{\mathrm{n}}\frac{X_r - X_0}{R},$$
$$Y_r \leftarrow Y_r + Kf_{\mathrm{n}}\frac{Y_r - Y_0}{R}.$$

Jeżeli na podstawie powyższych rozważań zbudujemy program obracania korby, to będzie on mieć następującą postać:

```
process cranking ;
begin
        cobegin turning ; following ; coend ;
end ;
process turning ;
begin
        while stop = 0 do begin
                X_r := X_r - V · (Y_r - Y_0) / R
                Y_r := Y_r + V · (X_r - X_0) / R
                startIO put_arm_reference ( X_r, Y_r, Z_r, α_r ) ; delay ( T_0 ) ;
        end ;
end;
process following ;
begin
        while stop = 0 do begin
                startIO get_wrist_force ( f_x , f_y , f_z , m_x , m_y , m_z ) ;
```

$$
\begin{aligned}
&f_{\mathrm{n}} := f_x \cdot (X_r - X_0)/R + f_y \cdot (Y_r - Y_0)/R;\\
&X_r := X_r + K \cdot f_{\mathrm{n}} \cdot (X_r - X_0)/R;\\
&Y_r := Y_r + K \cdot f_{\mathrm{n}} \cdot (Y_r - Y_0)/R;\\
&Z_r := Z_r + K \cdot f_z;
\end{aligned}
\quad ;
$$

startIO put_arm_reference ($X_r, Y_r, Z_r, 0$) ; delay (T_0) ;

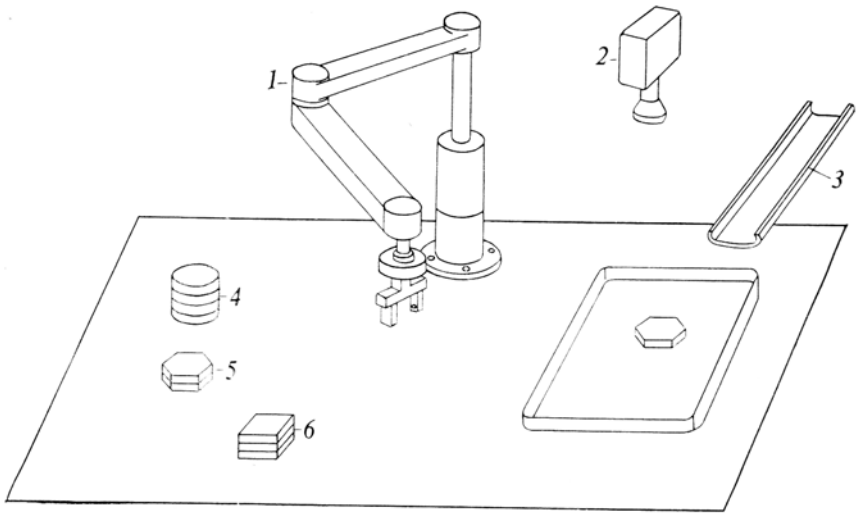
startIO get_wrist_force ($f_x, f_y, f_z, m_x, m_y, m_z$) ;

end ;

end ;

Sortowanie części z wykorzystaniem sensoryki wizyjnej

Na stół montażowy pojedynczo z pochylonego wypadają płaskie części trzech rodzajów: tarcze, kwadraty i sześciokąty. Części te wypadają są w kolejności przypadkowej. Należy je posortować i zgromadzić w wydzielonych miejscach. Rozpatrzymy teraz program pozwalający wykonywać taką pracę. Posłużymy się przy tym sensoryką wizyjną, która pozwala rozpoznawać kształt części płaskich (rysunek 30). Podczas chwytania części rozpoznanej można zastosować działanie centrujące, wykorzystujące sensorykę dotykową. Do zapewnienia dokładnego składowania części, jedna na drugą, wykorzystamy przedstawioną już funkcję (operację) puton. W operacji tej skorzystamy z sensoryki siłowej. Będzie to więc zadanie pokazujące kompleksowo wzajemne oddziaływanie sensorów pozwalających wykonywać funkcje widzenia, dotyku i odczuwania siły.



Rys. 30 Sortowanie części z wykorzystaniem sensoryki wzrokowej: 1 – manipulator; 2 – kamera telewizyjna; 3 – pochylone korytko; 4 – tarcze; 5 – sześciany; 6 – kwadraty.

Do rozpoznawania rodzaju części wykorzystuje się funkcję doboru wzorca. Kształty trzech rodzajów części płaskich można przedstawia się wcześniej następującymi wzorcami do porównywania: disc (tarcza), square (kwadrat), hexagon (sześciokąt). Po tym porównuje się obrazy wejściowe z wzorcami. Funkcja sensoryki wzrokowej template_matching (templatelist, result, x, y, φ) polega na zliczaniu obrazu wejściowego z istniejącymi obrazami wzorców. Nazwa wzorca, do którego dopasowuje się, umieszczana jest w result; współrzędne środka ciężkości – w x i y ; kierunek głównej osi bezwładności – w φ. Ponieważ x i y są współrzędnymi w układzie współrzędnych widzenia, trzeba, posługując się funkcją calculate_world_coord, zrealizować odpowiednie przejście do globalnego układu współrzędnych.

Podczas chwytania rozpoznanego obiektu wykorzystuje się działanie centrujące (center). Aby dokładnie uchwycić tarczę i kwadrat, po przeprowadzeniu jednorazowego działania centrującego (centrowania) należy rozsunąć palce, wykonać obrót o kąt 90° i ponownie przeprowadzić centrowanie. W przypadku sześciokąta powtórne centrowanie wykonuje się jest po obrocie o kąt 60°. Postępowanie takie pozwala uchwycić obiekt w ten sposób, że jego środek pokrywa się ze środkiem palców.

Trzymany obiekt przenosi się następnie przenieść w jedno z miejsc o nazwach: disc, square, hexagon. Operację tę można wykonać przez ruch ręki robota po trajektorii jak na rysunku 27. Nałożenie jednego obiektu na drugi można wykonać za pomocą operacji puton, z wykorzystaniem sensoryki siłowej. Pozwala to ostrożnie składować w ustalonych miejscach jeden obiekt na drugim. Po uwolnieniu obiektu przeprowadza się rękę robota w stan „oczekiwanie" i przystępuje się do rozpoznawania kolejnego obiektu za pomocą systemu wizyjnego. Cały program sortowania ma następującą postać:

```
process hand_eye_example ;
begin
        startIO put_hand_opening (full_opening) ;
        while TRUE do begin
                startIO template_matching (templatelist, result, x, y, φ) ;
                startIO caltulate_word_coord ( x, y, φ, h, x_w, y_w, α_w) ;
                case result of
                        disc : begin
                                        move_deproach ( x_w, y_w, , h, α_w ) ; center (grasp_force) ;
                                        startIO put_hand_opening (full_opening) ; delay ( T_0 ) ;
                                                α_w := α_w + 90 ;
                                        startIO put_arm_reference ( x_w, y_w, h, α_w ) ; delay ( T_0 ) ;
                                                center (grasp_force) ;
                                        move_deproach ( X disc, Y disc, Z disc, 0)
                                        puton ;
                   move_deproach ( X home, Y home, Z home, α home) ;
                                end
                        square : begin
                                        move_deproach ( x_w, y_w, , h, α_w ) ; center (grasp_force) ;
                                        startIO put_hand_opening (full_opening) ; delay ( T_0 ) ;
                                                α_w := α_w + 90 ;
                                        startIO put_arm_reference ( x_w, y_w, z_w, h, α_w ) ; delay ( T_0 )
;
                                                center (grasp_force) ;
                                        move_deproach ( X square, Y square, Z square, 0)
                                        puton ;
                                        move_deproach ( X home, Y home, Z home, α home);
                                end
                        hexagon : begin
                                        move_deproach ( x_w, y_w, h, α_w ) ; center (grasp_force) ;
                                        startIO put_hand_opening (full_opening) ; delay ( T_0 ) ;
                                                α_w := α_w + 60 ;
                                        startIO put_arm_reference ( x_w, y_w, h, α_w ) ; delay ( T_0 ) ;
                                                center (grasp_force) ;
                                        move_deproach ( X hex, Y hex, Z hex, 0)
                                        puton ;
                                        move_deproach ( X home, Y home, Z home, α home);
                                end ;
        end ;
end ;
```

Rozdział 4
Programowanie robotów

Według: ISII T., SIMOJAMA. I., INOUE H., HIROSE, M., NAKADZIMA N.: *Mechatronika* (w jęz. ros.). Mir 1988

Na tle innych maszyn roboty wyróżniają się przede wszystkim uniwersalnością. Bierze się ona stąd, że robot jest programowalnym systemem mechanicznym. Przez wykorzystywanie różnych programów (oprogramowania) robot może realizować najróżnorodniejsze prace. Na poprzednich wykładach ustaliliśmy zbiór podstawowych funkcji robota. Poznaliśmy też sterowania procesami równoległymi, które pozwalają równolegle wykorzystywać te funkcje w rzeczywistej skali czasu. Poznaliśmy kilka przykładów budowania programów pracy robotów wyposażonych w „zmysły" czyli sensorykę. W tym wykładzie zajmiemy się budowaniem systemów robotycznych o znacznej uniwersalności i zdolności do rozszerzania zadań. Można to osiągnąć dzięki zastosowaniu do programowania robotów języka ogólnego, rozpatrzonego na przykładach współdziałania z sensorami.

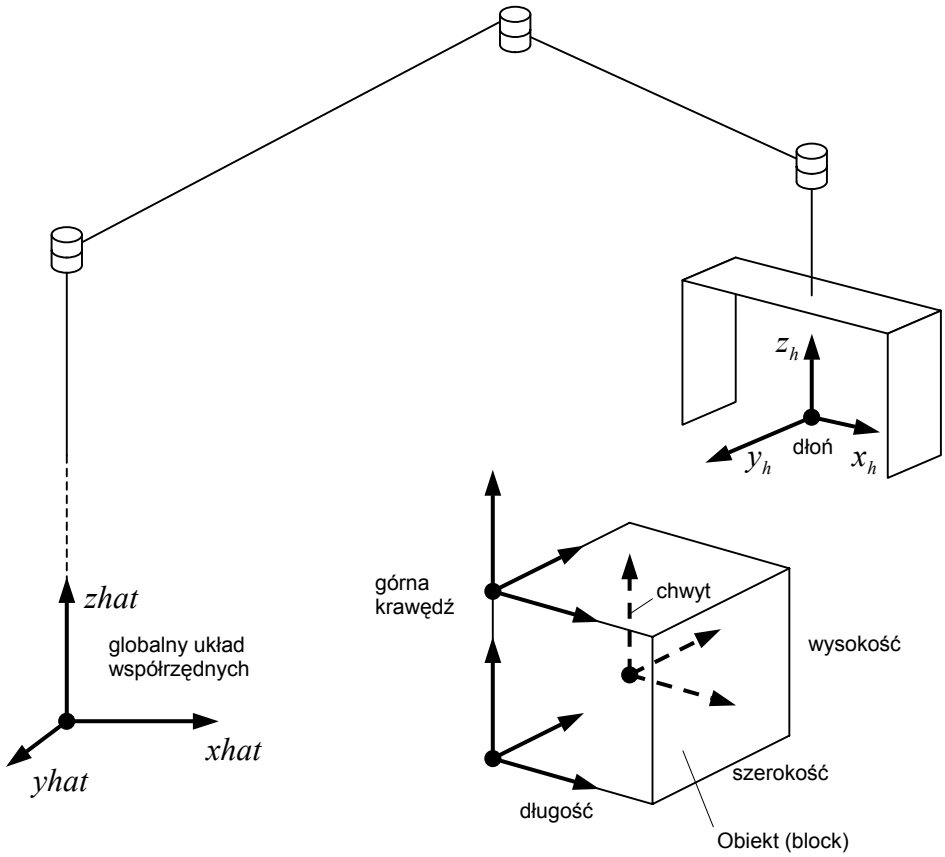
Programy, napisane dla robotów, składają się z dwóch części. W pierwszej opisuje się jest kolejność działań robota, w drugiej – środowisko i obiekt, którym robot manipuluje. Pierwsza część programowana jest jako procedura, w której wykorzystuje się są struktury syntaktyczne (składnię logiczną), dotyczące sensoryki i działań robota. W strukturze sterowania programem, oprócz typowych operatorów sterujących, należy przewidzieć konstrukcje syntaktycznych, które pozwolą wykorzystać sensorykę i działania jako procesy równoległe.

Druga część programu zawiera opis trójwymiarowego środowiska roboczego. Rozpatrywane są pozycje i orientacje obiektów i ręki robota, przytaczana jest niezbędna informacja o obiektach, a także ich powiązania. Aby skutecznie zestawić opisy w języku robota, dobrze jest mieć standardowe przedstawienia (reprezentacje) typów danych (wektorów, układów współrzędnych, przekształceń układów współrzędnych itd.). W wykładzie tym wykorzystamy struktury syntaktyczne języka ASRL (A Simple Robot Language, dosł. prosty język robota)). U jego podstaw leży język AL, opracowany w Uniwersytecie Stanforda oraz język AL/L, opracowany w Uniwersytecie Tokijskim i będący wynikiem wspólnego wykorzystania języków AL i LISP. Za podstawę przyjmiemy opis w języku PASCAL. Niezbędne będzie przy tym wprowadzenie szczegółowych operatorów działania robota, typów danych, struktury sterowania. Dzięki takiemu podejściu powstał rozszerzony wariant języka uniwersalnego. Jeżeli chodzi o system lingwistycznej obróbki informacji, to obróbka programu wyjściowego realizowana jest translatorem. Dla programu sterującego procesami równoległymi, pokazanego w poprzednim wykładzie, tworzy się wyjściowy program stanu realizującego zadaną kolejność funkcji. Poniżej przedstawimy konstrukcje składni i funkcji szczegółowych języka robota.

Typy danych

W języku ASRL środowisko robocze, składające się z robota i obiektu, opisywane jest jednoznacznie przez wykorzystanie kilku prostokątnych układów współrzędnych, związanych z obiektem, oraz wzajemnych związków miedzy tymi układami. Rysunek 31 pokazuje przykład układów współrzędnych hand i block, rozmieszczonych w środowisku roboczym i związanych, odpowiednio, z dłonią robota i obiektem (tu sześcianem). Istnieje również globalny układ współrzędnych (world), w którym początek układu współrzędnych hand określa się wektorem pozycji dłoni robota w przestrzeni roboczej, a orientację samej dłoni – macierzą obrotu układu współrzędnych hand. Wcześniej wprowadziliśmy ograniczenie, że orientację dłoni można zmieniać tylko przez jej obrót wokół osi pionowej. Jednak w poniż-

szych opisach rozpatruje się jest obrót w ogólnym przypadku. Analogicznie można postąpić dla wyrażenia pozycji i orientacji obiektu.



Rys. 31 Układy współrzędnych dłoni i obiektu.

W programie opisu środowiska roboczego, w którym funkcjonuje robot, oprócz danych cyfrowych (liczby całkowite i rzeczywiste) wygodnie jest również wykorzystywać standardowe typy danych, na przykład SCALAR, VECTOR, ROTATION, COORDINATE, TRANSFORM.

Typ danych SCALAR przedstawia sobą typowe wielkości skalarne. W języku AL opisanie danych typu SCALAR prowadzi się z uwzględnieniem wymiaru i jednostki pomiaru, w języku ASRL – bez uwzględnienia wymiaru i jednostki pomiaru.

Danymi typu VECTOR wyraża się są wektory trójwymiarowe. Wykorzystuje się do tego funkcję VECT, której argumentami są trzy typy danych SCALAR. I tak, w globalnym układzie współrzędnych wektor jednostkowy można przestawić następująco (symbol $\leftarrow$ oznacza tu przywłaszczanie): $xhat \leftarrow \mathrm{VECT}(1,0,0);\ \ yhat \leftarrow \mathrm{VECT}(0,1,0);\ zhat \leftarrow \mathrm{VECT}(0,0,1)$.

Dane typu ROTATION wyrażają obrót, który można przedstawić wektorem odpowiadającym osi obrotu i kątowi obrotu względem tej osi. Dane typu ROTATION określa się funkcją ROT o dwóch argumentach: $xhat$ należy obrócić względem osi $zhat$ o kąt 90° aby otrzymać $yhat$. Taką operację można zapisać w postaci

$$yhat \leftarrow \mathrm{ROT}(zhat,90) * xhat;$$

W takim przypadku jako funkcje, określające oś obrotu i kąt obrotu, można wykorzystać funkcje AXIS(r) i ANGLE(r), w których argumentami są dane typu ROTATION.

Typ danych COORDINATE zawiera w sobie dane, za których pomocą można opisać układ współrzędnych. Rozpatrywany układ współrzędnych można przedstawić wektorem pozycji początka współrzędnych i kątem obrotu względem układu globalnego. Początek układu współrzędnych hand (rysunek 31) określony jest punktem x_1, y_1, z_1, a jego orientacja –kątem obrotu α względem osi $zhat$.

Wartość hand można przedstawić wyrażeniem wykorzystując funkcję COORD, której zmiennymi parametrami są obrót i wektor

$$\text{hand} \leftarrow \text{COORD}(\text{ROT}(zhat, \alpha)), \text{VECT}(x_1, y_1, z_1);$$

Typ danych TRANSFORM wiąże (jednoczy) dane, za których pomocą można realizować przekształcenie współrzędnych i wektorów. Dane tego typu, podobnie jak dane układów współrzędnych, mają dwie składowe: kąt obrotu i wektor przeniesienia równoległego. Dane te określane są funkcją TRANS(r, v), której argumentami są składowe wspomniane wyżej

$$t1 \leftarrow \text{TRANS}(\text{ROT}(xhat, 45), 3 * zhat);$$

Przekształcenie współrzędnych wykonuje się przez wymnożenie danych typu TRANSFORM przez wektor czy układ współrzędnych. Na przykład

$$v1 \leftarrow t1 * yhat;$$

Tu, po obrocie $yhat$ o kąt 45° wokół osi $xhat$ w kierunku osi $zhat$, wykonywane jest przeniesienie równoległe o +3. A więc $v1$ (0, 0,71, 3,71). Przekształcenia dotyczące układu współrzędnych można realizować analogicznie. Funkcja $\text{INV}(t)$ oznacza przekształcenie odwrotne po t. Środowisko otaczające, przedstawione na rysunku 1, można opisać następującym programem:

SCALAR width, depth, height, x_1, y_1, z_1, α ;
VECTOR $xhat, yhat, zhat$;
COORDINATE hand, block, block_grasp, block_top ;
$\text{hand} \leftarrow \text{COORD}(\text{ROT}(zhat, \alpha)), \text{VECT}(x_1, y_1, z_1);$
$\text{block} \leftarrow \text{COORD}(\text{ROT}(zhat, 300)), \text{VECT}(200, 40, 0);$
block_grasp ← block + VECT(width/2, depth/2, height/2));
block_top ← block + VECT(0, 0, height);

Operację sumowania, dotyczącą typów danych COORDINATE i VECTOR, można rozpatrywać jak złożenie składowych zmiennych typów COORDINATE i VECTOR.

Powiązanie układów współrzędnych

W zależności od celów rozważań obiekt można rozpatrywać w kilku układach współrzędnych. Sześcian, przedstawiony na rysunku 31, oprócz układu współrzędnych block, opisującego jego pozycję i orientację, opisuje się jest w układach: block_grasp (układ zadawania pozycji uchwycenia) i block_top (układ opisania górnej krawędzi sześcianu). W tym przypadku można uważać, że wzajemne relacje między tymi trzema układami współrzędnych pozostają bez zmian. Gdy przemieszczany jest układ współrzędnych block, związany z sześcianem, należy zrealizować odpowiednie odnowienie bez sprzeczności, tj. zachować stosunki wiążące wartości współrzędnych tych dwóch układów. Stosunek (powiązanie) takiego zjednoczenia układów współrzędnych można zapisać za pomocą operatora AFFIX:

AFFIX block_top TO block RIGIDLY;

AFFIX block_grasp TO block RIGIDLY;

Dwa operatory AFFIX pozwalają rozpatrywać trzy powyższe wyżej układy współrzędnych (block, block_grasp, block_top) jako wzajemnie zjednoczone (powiązane). Sterowanie można prowadzić automatycznie przy zachowaniu określonych stosunków wzajemnych. Oznacza to, że jeżeli przemieści się układ block, to przemieszczą się również układy block_grasp i block_top. Jeżeli przemieści się układ block_grasp, to przemieszczą się układy block i block_top. Takie same stosunki powiązania ustalają się, gdy sześcian trzymany jest w dłoni robota. Tak więc, jeżeli istnieje opis

AFFIX block_grasp TO hand RIGIDLY;

to trzy układy współrzędnych, odnoszące się do dłoni robota i sześcianu, będą rozpatrywane jako jedna całość. Zgodnie z tym podczas przemieszczania układu hand odnawiane są wartości współrzędnych układu block. Jeżeli zapisać prawo, zgodnie z którym przemieszcza się układ współrzędnych

block, to można określić prawo ruchu układu współrzędnych hand, czyli ruchu niezbędnego do osiągnięcia zadanego przemieszczenia układu block. Jeżeli sterowanie powiązaniami układów współrzędnych realizowane jest automatycznie bez sprzeczności, to przy rozpatrzeniu przemieszczenia obiektu można zapisać następujący stosunek:

MOVE block TO final ;

W takim przypadku program zapisywany jest w postaci bardziej naturalnej, łatwo zrozumiałej. Gdy podczas montażu dwóch części tworzy się jedna całość, także należy wykorzystywać opisanie połączenia. Można wydzielić dwa rodzaje połączenia: RIGID (sztywne) i NONRIGID (niesztywne).

Zapis

AFFIX one TO another RIGIDLY;

oznacza połączenie, w którym „one" (jedna część) mocno łączona jest z „another" (drugą częścią) za pomocą śruby lub innego urządzenia. Jeżeli przemieszcza się „one", to przemieszcza się i „another", jeżeli przesunie się „another", to przesunie się i „one".

Zapis

AFFIX cup TO saucer RIGIDLY;

charakteryzuje powiązanie filiżanki z podstawką. Jeżeli przesuniemy podstawkę, to przesunie się także filiżanka. Jednak, jeżeli przesuniemy filiżankę, to podstawka nie przesunie się.

Jeżeli wcześniej określimy cechy (osobliwości) dwóch układów współrzędnych, które mają się jednoczyć za pomocą operatora AFFIX, to na postawie wyrażenia

AFFIX block_grasp TO block RIGIDLY;

można automatycznie wyliczyć względne położenia tych układów współrzędnych. W przypadkach, gdy układ block_grasp nie jest określony, jego i operator AFFIX można określić przez wykorzystanie rezerwowego słowa NILROT, wskazującego na brak obrotu:

AFFIX block_grasp TO block AT ;

TRANS(NILROT,VECT(width/2, depth/2, height/2)) RIGIDLY ;

Do usuwania stosunku połączenia wykorzystuje się operator UNFIX. Kiedy dłoń przestaje chwytać obiekt, następuje wykonanie operacji

UNFIX block_grasp FROM hand ;

Po działaniu operatora UNFIX usuwany jest stosunek połączenia między hand i block, a więc jeżeli przemieści się dłoń, to obiekt pozostanie na miejscu.

Operatory działań

Aby zmusić robot do ruchu, należy wprowadzić operator MOVE, który zapisywany jest następująco:

MOVE frame TO destination ;

Operator frame odpowiada tu sterowanemu układowi współrzędnych. Operator ten jest analogiczny do zmiennej typu COORDINATE, opisującej dłoń i obiekt manipulowany. Operator destination odpowiada układowi współrzędnych opisującemu docelową (końcową) pozycję i orientację. Zmiennymi typu COORDINATE zwykle opisuje się punkt docelowy.

W naszym przypadku operator MOVE oznacza taki ruch robota, po którym nastąpi pokrycie układów współrzędnych frame i destination. Za pomocą standardowego operatora MOVE, jak i za pomocą wprowadzonego wcześniej operatora move_deproach, tworzy się trajektorię przechodzącą przez punkty odprawy i przybycia. Jeżeli nie będziemy rozpatrywać tych punktów a zajmiemy się tylko bezpośrednim przemieszczeniem do destination (dosł. przeznaczenie), to można zapisać

MOVE frame TO destination DIRECTLY;

Dla operatora MOVE można wykorzystać plik VIA, który wskazuje punkty pośrednie na drodze ruchu do punktu docelowego

MOVE hand TO destination VIA $c1, c2, \ldots, cn$;

gdzie $c1, c2, \ldots, cn$ – wzory lub zmienne typu COORDINATE. Operator hand łączy te punkty i tworzy trajektorię w postaci linii łamanej, która dąży do destination. Gdy wskażemy bezpośrednie (proste) przemieszczenie DIRECTLY, punkty odprawy i przybycia można rozpatrywać jako punkty pośrednie.

Podczas realizacji operatora MOVE może pojawić się sytuacja, kiedy, wskutek powstałych warunków, pożądane jest przełączenie na jakiekolwiek inne działanie. W takim przypadku można posłużyć się plikiem ON. Konstrukcja syntaktyczna z plikiem ON ma postać:

ON $\langle condition \rangle$ DO $\langle action \rangle$;

Na ustalony warunek $\langle condition \rangle$, przełącza się działanie $\langle action \rangle$. Przykładem może być działanie operatora MOVE z wykorzystaniem pliku ON

```
MOVE hand TO final
    ON FORCE ⟨(zhat)⟩ treshold DO STOP hand;
```

Operator ten porusza dłoń do położenia końcowego. Jednak jeżeli na drodze ruchu siła reakcji, działająca w kierunku osi z, przewyższy ustalony próg (treshold) wartości, to działanie jest przerywane. Podczas omawiania powiązania układów współrzędnych zobaczyliśmy, że jeżeli obiekt trzymany i dłoń powiązane są operatorem AFFIX, to zapis MOVE block to final odpowiada prostemu przenoszeniu obiektu. Jeżeli uwzględnimy rozpatrzony wcześniej opis działań podczas przemieszczenia obiektu, to działanie, podczas którego trzymany obiekt umieszczany jest w docelowym miejscu (final), można opisać następująco:

```
MOVE block TO final+VECT(0,0,–10)
    ON FORCE ⟨(zhat)⟩ treshold
        DO begin STOP hand; OPEN hand TO full_opening; end;
```

Zwróćmy uwagę, że miejsce, dokąd powinien być obiekt przeniesiony, znajduje się 10 mm poniżej miejsca final. Chodzi o to, aby uniknąć upadku obiektu po tym jak zostanie on postawiony na miejscu (upadek może nastąpić wskutek błędu w nastawie wielkości zadanej).

Dla zadania sprzężenia zwrotnego od siły pojawiającej się w procesie działania wykorzystuje się plik WITH. Jeżeli, na przykład, wymaga się jest lekkiego dociśnięcia ciała do stołu i następnie w takim stanie przesunięcia go z poślizgiem do miejsca przeznaczenia, to można posłużyć się następującym opisem:

```
MOVE block TO destination
    WITH FORCE (zhat)=pressure;
```

Obracanie korby także można opisać za pomocą pliku WITH.

Oprócz pokazanych wyżej istnieją również operatory innych działań: OPEN, CENTER, SQUEEZE, OPERATE. Za pomocą operatora OPEN reguluje się jest szerokość rozwarcia palców dłoni, a operatora SQUEEZE – siłę chwytu. Operator CENTER pozwala uchwycić obiekt w środku, bez spowodowania przesunięcia obiektu.

```
OPEN hand TO finger_opening;
SQUEEZE hand BY grasping_force;
CENTER hand;
```

Istnieją również operatory do sterowania urządzeniami peryferyjnymi, takimi jak narzędzia (np. wkrętak), przenośnik, uchwyt itp. Pracę wkrętaka elektrycznego można opisać następująco:

```
OPERATE driver;
WITH ETORQUE-fastening_torque;
```

gdzie operator fastening_torque charakteryzuje moment sił, jaki pojawia się podczas dokręcania śruby.

Struktura programu sterującego

Podczas budowy programu dobrze jest posługiwać się strukturą blokową, w której bloki odpowiadają operatorom begin i end. Operatorami syntaktycznymi (składniowymi) przeznaczonymi do tworzenia struktury programu sterującego są operatory IF, FOR, WHILE, CASE. Są one takie same jak w języku PASCAL i dlatego nie będziemy ich tu rozpatrywać.

Dla opisania działań równoległych wprowadza się bloki o nazwach COBEGIN, COEND. W poniższym przykładzie równolegle wykonywane są zadania, które znajdują się w bloku ograniczonym przez COBEGIN i COEND.

```
COBEGIN
        MOVE blue_arm TO b_park;
        MOVE red_arm TO r_park;
COEND
```

Zgodnie z tym zapisem, blue_arm (dosł. niebieska ręka) i red_arm (dosł. czerwona ręka) powinny przemieścić się do punktów zatrzymania, odpowiednio, b_park i r_park.

Do opisania synchroniczności działań równoległych istnieją operatory SIGNAL i WAIT, wykorzystujące semafor. Semafor jest zmienną całkowitoliczbową. Jego wartość zmieniana jest operatorami SIGNAL i WAIT. Za wartość początkową przyjmuje się 0.

Operatora WAIT s zmniejsza wartość semafora s. Gdy wartość s staje się ujemna, przerywane jest wykonanie procesu oznaczonego tym operatorem. Przy nieujemnej wartości s wykonanie rozpatrywanego procesu trwa nadal.

Operator SIGNAL s zwiększa wartość semafora s o 1. Jeżeli s dotyczy procesu, który po wykonaniu operatora WAIT został zatrzymany, to wykonanie tego procesu jest odnawiane. Gdy nie ma procesu zatrzymanego, nic się nie dzieje.

Rozpatrzmy teraz przykład programu działań równoległych. Niech lewa ręka bierze przedmiot (kostka) i przekłada ją do ręki prawej. Program związanych z tym działań można przedstawić następująco:

```
COBEGIN
  BEGIN (działanie leftarm)
    MOVE leftarm TO block;          Ręka lewa przemieszcza się do kostki
    CENTER lefthand                 Uchwycenie
    AFFIX block TO leftarm;
    MOVE block TO pass;             Sześcian przemieszczany jest do miejsca przekazania
                                    (pass)
    SIGNAL ready_to_pass;           Zakończenie przygotowania do przekazania
    WAIT caught;                    Oczekiwanie do przyjęcia prawą ręką
    OPEN lefthand TO full_opening;  Przekazanie kostki
    UNFIX block FROM leftarm;
    SIGNAL passed;                  Zakończenie przekazywania
  END;
  BEGIN (działanie rightarm)
    OPEN righthand TO full_opening; Rozwarcie palców
    MOVE rightarm TO catch;         Przemieszczenie do miejsca przyjęcia kostki
    WAIT ready_to_pass;             Oczekiwanie na sygnał zakończenia przygotowania do
                                    przekazania  ręką lewą
    CENTER righthand;               Uchwycenie kostki
    SIGNAL caught;                  Sygnał o otrzymaniu
```

```
    WAIT passed;                    Oczekiwanie na zakończenie działań ręki lewej
    MOVE block TO pallet;
  END;
COEND
```

Makrofunkcje

Przedrostek „makro” oznacza zapis programu w postaci skróconej. Działania typowe, często wykorzystywane i powtarzane, określa się jako makrodziałania, co upraszcza zestawienie programu w języku wejściowym. Rozpatrzymy przykład określenia makrodziałania.

```
DEFINE pick ($object,$graspforce)
  ⊂MOVE hand TO grasp_of($object) ;
       CENTER hand ;
       SQUEEZE hand BY $graspforce ;
       AFFIX $object TO hand RIGIDLY ;⊃
```

pick – nazwa makrookreślenia; $object i $graspforce – argumenty. Część programu, zawarta między $\subset$ i $\supset$, jest centralnym blokiem makrookreślenia. Funkcja graspforce, zapisana w bloku centralnym, jest funkcją, za której pomocą wylicza się punkt uchwycenia obiektu. Jeżeli w programie w języku wejściowym spotyka się opis pick(block, 100), to oznacza to zrealizowanie wywołania makrodziałania jak wyżej. W bloku centralnym następuje wtedy zamiana funkcyjnego argumentu $object na zmienną block, a $graspforce – na liczbę 100. Tekst programu przyjmuje zaś postać

```
MOVE hand TO grasp_of(block) ;
CENTER hand ;
SQUEEZE hand BY 100 ;
AFFIX block TO hand RIGIDLY ;
```

Struktury składniowe

Tabela 2 przedstawia rozpatrzone wyżej podstawowe struktury składni (syntaksu) języka robotów. Język ten jest w zasadzie analogiczny do języków AL i AL/L. Aby nie komplikować objaśnień, postanowiliśmy przedstawić syntaks w postaci skróconej. W przedstawionych konstrukcjach syntaktycznych wykorzystywane jest tylko połączenie RIGID. Dla zmiennych typu VECTOR, ROTATION, COORDINATION, TRANSFORM operatory nie są rozpisane szczegółowo. Dodatkowe objaśnienia zostaną przedstawione w następnym wykładzie, w którym rozpatrzmy przykład programu wykonania operacji montażu. Zapisany w tym języku program wejściowy przekształcany jest w program końcowy odpowiadający zestawowi funkcji, rozpatrzonych w wykładzie o wzajemnym oddziaływaniu sensorów. Przekształcenie to odbywa się za pomocą systemu obróbki informacji lingwistycznej. Ten końcowy program realizowany jest następnie za pomocą programu sterującego i procesów równoległych.

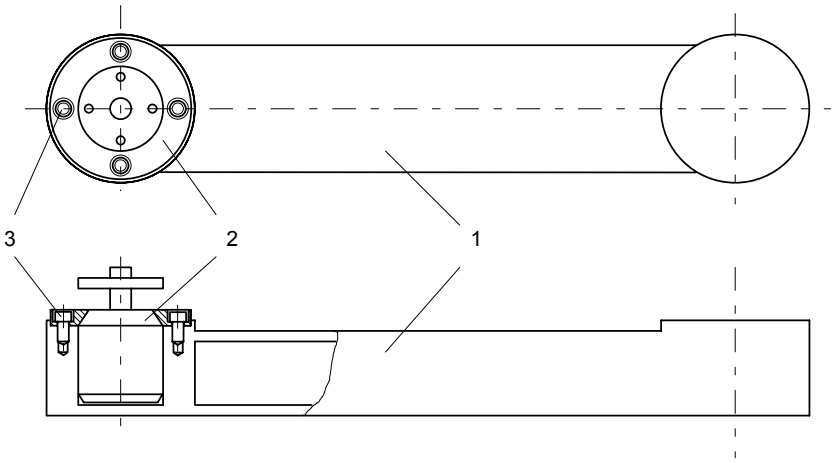
Tabela 2 Podstawowe struktury syntaktyczne języka robotów (s – operator; sem – zmienna semaforowa)

Bloki	BEGIN $s;s;\cdots s;$ END;
Typy danych	SCALAR | VECTOR | ROT | COORD | TRANS
Operator przyswajania	$\langle \text{variable} \rangle \leftarrow \langle \text{expression} \rangle;$
Operatory sterujące	IF $\langle \text{condition} \rangle$ THEN $\langle \text{statement} \rangle$ [ELSE $\langle \text{statement} \rangle$; FOR $\langle \text{var} \rangle = \langle \text{initial} \rangle, \langle \text{final} \rangle$ [STEP $\langle \text{increment} \rangle$]DO $\langle \text{statement} \rangle$; WHILE $\langle \text{condition} \rangle$ DO $\langle \text{statement} \rangle$; CASE $\langle \text{var} \rangle$ OF BEGIN $\langle 11 \rangle : \langle \text{block} \rangle; \langle 12 \rangle : \langle \text{block} \rangle; \cdots;$ END;

Powiązanie układów współrzędnych	AFFIX ⟨coord⟩ TO ⟨coord⟩ [AT ⟨trans⟩]; UNFIX ⟨coord⟩ FROM ⟨coord⟩;
Operatory działań	MOVE ⟨coord⟩ TO ⟨coord⟩ [DIRECTLY] [VIA ⟨coord⟩, ⟨coord⟩, ⋯] [ON ⟨condition⟩ DO ⟨statement⟩] [WITH ⟨sensor_expression⟩]; OPEN ⟨hand⟩ TO ⟨opening⟩; CENTER ⟨hand⟩; SQUEEZE ⟨hand⟩ BY ⟨grasp - force⟩; OPERATE ⟨device⟩ BY ⟨control_parameter⟩; STOP ⟨device⟩;
Procedura	PROCEDURE ⟨pname⟩ [(arg, ⋯, arg)]; BEGIN $s;s;\cdots s;$ END;
Działania równoległe	COBEGIN $s;s;\cdots s;$ COEND;
Synchronizacja	SIGNAL ⟨sem⟩; WAIT ⟨sem⟩;
Makrookreślenie	DEFINE ⟨macroname⟩ [(arg, ⋯, arg)] ⊂ ⟨macro_body⟩ ⊃;

Program montażu robotycznego

Spróbujemy teraz powiązać zagadnienia rozpatrzone dotychczas. Zrobimy to na przykładzie wykonania operacji montażowych. Obiektem montażu będzie zespół ręki robota hipotetycznego (robot będzie montowany przez robot). Zespół ten składa się z przedramienia, napędu i czterech śrub (rysunek 32). Napęd jest blokiem składającym się z silnika elektrycznego, przekładni zębatej i enkodera (urządzenia kodującego kąt obrotu wału silnika). Taki blok realizuje sterowanie dłonią ręki hipotetycznej. Z rysunku 32 widać, że napęd wstawiany jest w gniazdo przedramienia i mocowany czterema śrubami. Rozpatrzmy więc tę nieskomplikowaną operację montażową.



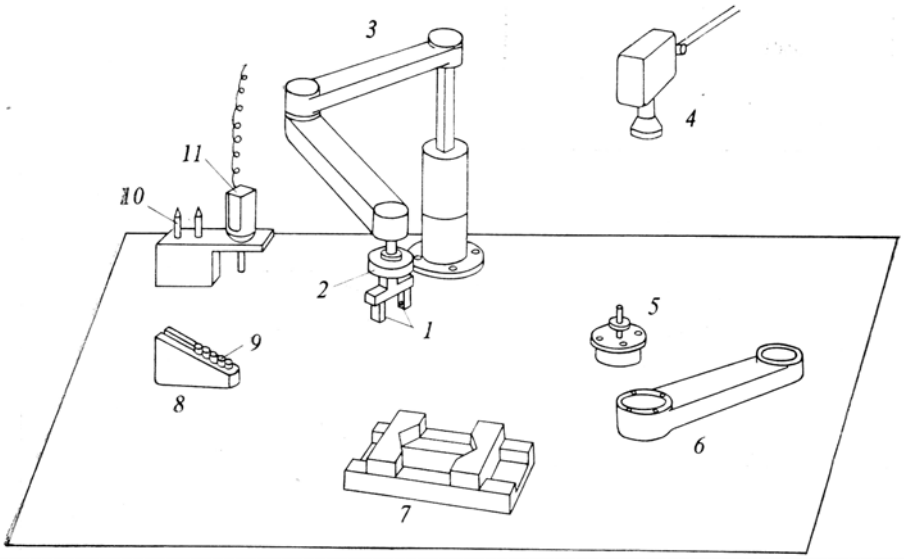
Rys. 32 Obiekt montażu: 1 – przedramię; 2 – napęd; 3 – śruba z gniazdem sześciokątnym

Miejsce wykonania prac montażowych

Dla wyobrażenia sobie miejsca montażu posłużymy się rysunkiem 33. Na tym miejscu roboczym są:

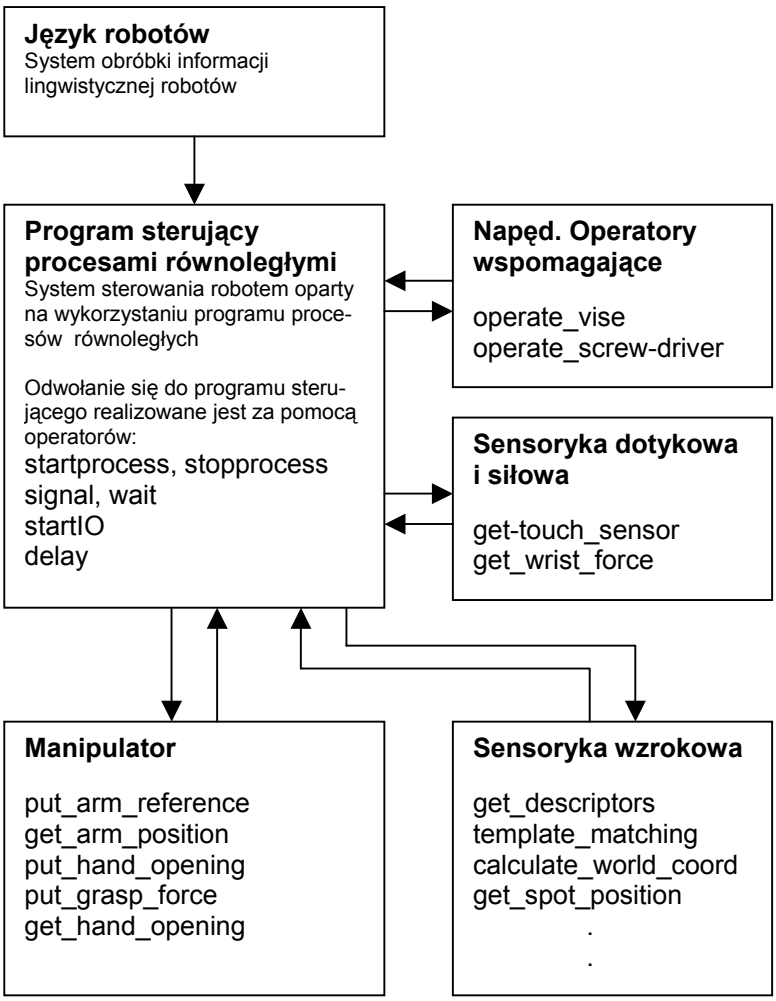
- manipulator (hipotetyczna ręka) wyposażony w sensor siły;
- system widzenia technicznego;
- zacisk (imadło), napędzany elektrycznie, do zamocowania obiektu montowanego;
- stojak z kołkami prowadzącymi i podajnik ze śrubami.

Człon i napęd umieszczone są w ściśle określonych miejscach stołu montażowego. Śruby znajdują się w podajniku i ustawione są w pojedynczym szeregu.



Rys. 33 Miejsce montażu: 1 – sensory styku (dotyku); 2 – blok sensorów siły; 3 –manipulator; 4 – kamera telewizyjna; 5 – blok napędu; 6 – przedramię; 7 – imadło elektryczne; 8 – podajnik śrub; 9 – śruba; 10 – kołki prowadzące; 11 – wkrętak elektryczny.

Rysunek 34 pokazuje strukturę oprogramowania całego systemu montażowego. Każdy blok tego programu ma odpowiedni komentarz wskazujący rozdział, w którym można znaleźć objaśnienie. W blokach, odpowiadających manipulatorowi, widzeniu technicznemu podsystemowi sensoryki dotykowej i sensoryki siłowej, wykorzystany jest zbiór funkcji, które rozpatrywaliśmy w wykładach o ręce i zmysłach robota. Zbiorem tych funkcji steruje monitor (program sterujący) czasu przebiegu procesów równoległych, opisany w wykładzie o sterowaniu procesami równoległymi. W blokach języka robota wykorzystany jest system obróbki informacji lingwistycznej przedstawiony w wykładzie o językach programowania robota. Podczas wykonywania prac montażowych wykorzystuje się imadło elektryczne i silnik elektryczny. Sterowanie tymi urządzeniami opisuje się jest w języku robota operatorem OPERATE.

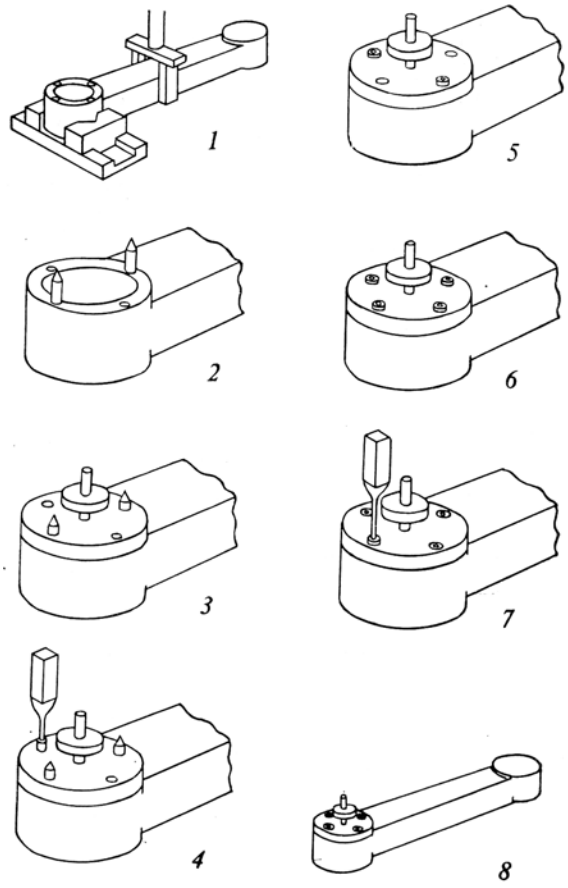


Rys. 34 Struktura oprogramowania systemu montażowego

Kolejność montażu

Zanim zbudujemy program montażu, krótko rozpatrzymy jego kolejność. Możliwe są różne sposoby montażu. Dobrze jest wybrać taki sposób, przy którym można łatwo zrealizować oprogramowanie. Mając to na uwadze wykorzystajmy następującą kolejność montażu (rysunek 35):

1) zacisnąć przedramię w imadle;
2) wstawić dwa kołki prowadzące (ustalające);
3) wykorzystując kołki ustalające, usadowić napęd w gnieździe członu;
4) nałożyć dwie śruby;
5) wyciągnąć kołki ustalające;
6) nałożyć pozostałe dwie śruby;
7) pewnie umocować napęd na miejscu, mocno dokręciwszy cztery śruby;
8) wyjąć zmontowany zespół z imadła i umieścić w przynależnym miejscu.



Rys. 35 Kolejność montażu.

Kołki ustalające są przyrządem wspomagającym, pozwalającym łatwo dopasować cztery otwory w kołnierzu napędu do z czterech nagwintowanych otworów w przedramieniu. Wszystkie te otwory mają wejście stożkowe (są sfazowane), aby kołki i śruby wchodziły bez kłopotu. Końce kołków mają również kształt stożkowy, przez co łatwo wchodzą w otwory. Wykorzystanie kołków z takimi końcami znacznie upraszcza tę część programu, która opisuje proces umieszczania kołka w otworze.

Opisanie środowiska roboczego i części montowanych

Program wykonania montażu składa się z dwóch etapów. W pierwszym opisuje się kolejność operacji montażowych, w drugiej – dane o otoczeniu (środowisku roboczym) i używanych częściach montażowych. Z wykładu o językach robotów dowiedzieliśmy się, że w języku robotów ASRL pozycję i orientację obiektu opisuje się za pomocą zmiennych układów współrzędnych, których początki związane są z ważnymi miejscami obiektu. Rysunek 36 pokazuje układy współrzędnych, którymi posługujemy się podczas wykonania rozpatrywanych operacji montażowych.

W głównej części programu, opisującego operacje montażowe, należy koniecznie wprowadzić odpowiednią informację o wartościach zmiennych, rodzajach układów współrzędnych oraz ich wzajemnych powiązaniach. Rozpatrzmy teraz zespół napędu. Ma on zadanych sześć układów współrzędnych. W języku robotów dane w tych układach współrzędnych i ich wzajemne powiązania można zapisać następująco:

du $\leftarrow$ COORD(ROT($zhat, \alpha_d$),VECT($x_d, y_d, 0$));

AFFIX du_grasp TO du AT TRANS(NILROT,VECT($0,0,h_1$));

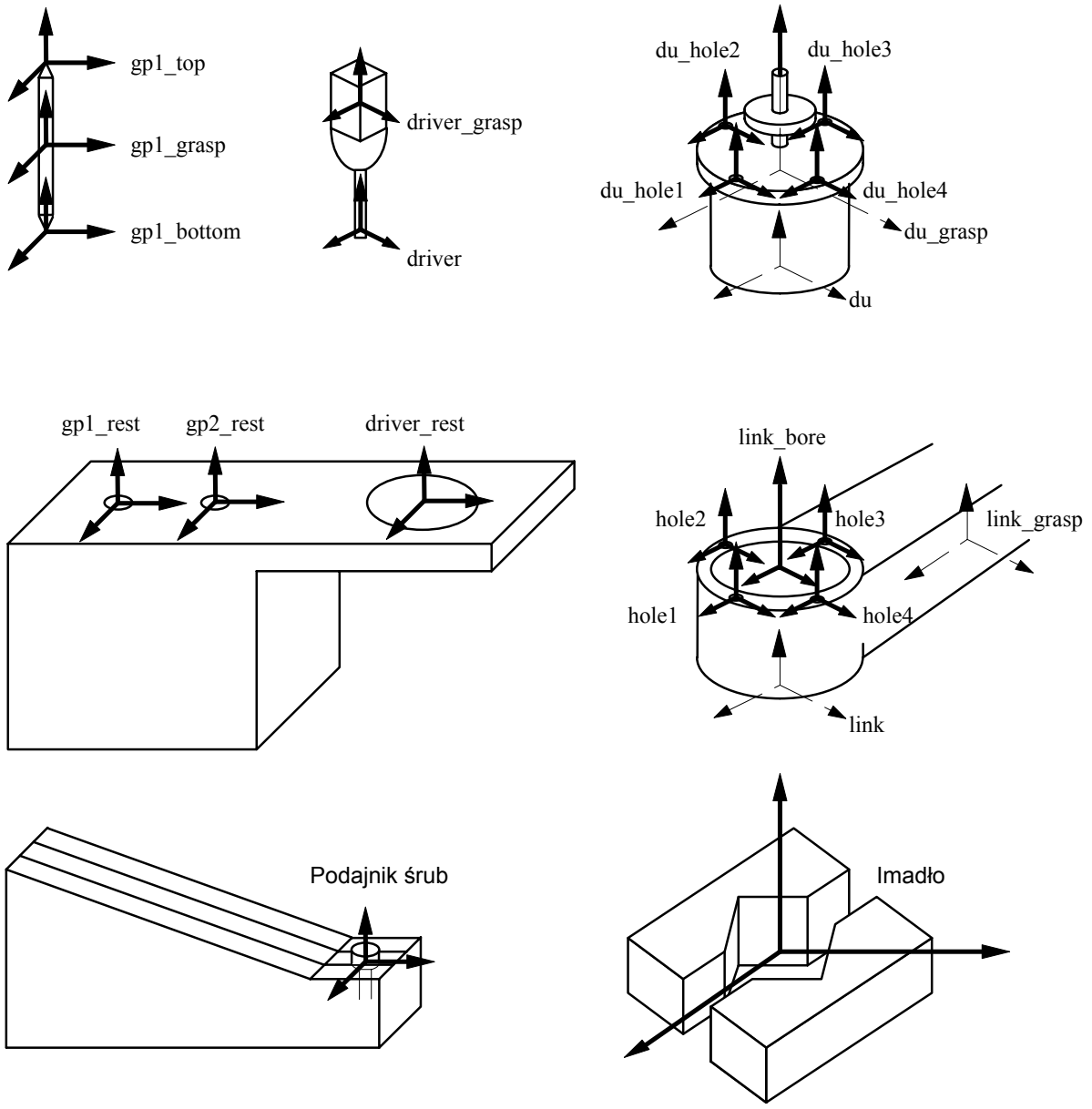
AFFIX du_hole1 TO du AT TRANS(NILROT,VECT($0,r,h$));

AFFIX du_hole2 TO du AT TRANS(NILROT,VECT($r,0,h$));

AFFIX du_hole3 TO du AT TRANS(NILROT,VECT($0,-r,h$));

AFFIX du_hole4 TO du AT TRANS(NILROT,VECT($-r$,0,h);

Układ współrzędnych du, związany sztywno z zespołem napędu, ma w układzie globalnym pozycję opisywaną współrzędnymi $x_d, y_d, 0$ oraz orientację, którą można przedstawić kątem obrotu α_d względem osi z. Z zespołem napędu związany jest również układ współrzędnych du_grasp, który wskazuje na sposób chwytania zespołu napędu, oraz cztery układy współrzędnych du_hole, odpowiadające czterem otworom. Układy te zachowują stałe powiązania z układem du. Zapewnia się to za pomocą operatora AFFIX. Podobnie opisuje się pozostałe układy sztywno powiązane, odpowiednio, z przedramieniem, wkrętakiem, imadłem, kołkami ustalającymi i podajnikiem śrub.



Rys. 36 Układy współrzędnych środowiska prac montażowych

Makrookreślenia podstawowych działań montażu

W wykładzie o wzajemnym oddziaływaniu sensorów przytoczyliśmy kilka przykładów programów takiego oddziaływania. Działania, z którymi mieliśmy do czynienia w tych programach, są działaniami podstawowymi i charakteryzują się dużą uniwersalnością. Dlatego zaliczyliśmy je do działań elementarnych, które często można wykorzystywać do programowania pracy robota. Oprócz tego do waż-

nych podstawowych działań montażu można zaliczyć wstawianie kołka w otwór i wkręcanie śruby. Dla tych podstawowych działań można zrobić makrookreślenia (wyróżnione tu tłustym drukiem).

Wstawianie kołka

Jednym z podstawowych działań montażu jest wstawianie kołka w otwór. Sposobowi wykonania tego działania poświęcono wiele publikacji opracowanych na podstawie wyników prac naukowo-badawczych. Operacje związane z wstawianiem kołka w otwór można z grubsza podzielić na dwa etapy: (1) dostatecznie dokładne umieszczenie czoła kołka względem otworu, (2) dokładne wpychanie kołka w otwór. Jeżeli czoło kołka i skraj otworu mają ścięcia stożkowe (fazki), to wstawienie kołka w otwór nie jest trudne. Jeżeli brakuje tych fazek i luz miedzy kołkiem i otworem jest nieduży, to zachodzi potrzeba wprowadzenia operacje, podczas których, pochylając kołek, próbuje się jego końcem „wymacać" wejście w otwór. Potrzebny jest wtedy sensor siły reakcji. Struktura operacji się wtedy komplikuje i dlatego stosuje się kołki i otwory z fazkami. Jeżeli błąd pozycjonowania ręki jest dostatecznie mały, w porównaniu do średnicy otworu, to nietrudno jest umieścić koniec kołka dokładnie nad otworem. Jeżeli oś kołka jest przesunięta względem środka otworu, to przy próbie wstawienia kołka w otwór pojawi się odpowiednia siła reakcji. W celu płynnego wciskania kołka w otwór należy zapewnić takie warunki, przy których powstająca siła reakcji w płaszczyźnie poziomej staje się równa zeru. W tym celu podczas wstawiania kołka w otwór konieczne są odpowiednie poprawki. O tym, czy kołek dotarł do końca otworu można sądzić na podstawie siły reakcji pojawiającej się w kierunku pionowym. Wtedy można zakończyć wciskanie kołka. Opisane wyżej działania można określić jako makro o nazwie **pin_into_hole** (dosł. kołek do dziury). Program tego makrookreślenia ma postać

```
DEFINE pin_into_hole
   ⊂MOVE hand TO @–VECT( 0,0,d )
         ON FORCE( zhat ) > f1 DO STOP hand;
           MOVE hand TO @–VECT( 0,0,h )
         WITH FORCE( xhat ) = 0;
         WITH FORCE( yhat ) = 0;
         ON FORCE( zhat ) > f2 DO STOP hand;⊃
```

Działania tego makrookreślenia rozpoczynają się od stanu, w którym czoło kołka umieszczone jest nieco wyżej od zakładanego środka otworu. W drugim wierszu jest znak @, przedstawiający początkowy stan ręki. @–VECT($0,0,d$) odpowiada obliczeniu, kiedy stożkowa cześć końca kołka znajdzie się w płaszczyźnie wejścia do otworu. W wierszu czwartym mamy @–VECT($0,0,h$). Według tej różnicy ($d-h$) określana jest pozycja odpowiadająca płaszczyźnie dna otworu. f_1 jest wartością progową siły reakcji powstającej podczas działania walcowej powierzchni kołka na wejściu w otwór, zaś f_2 progową wartością siły reakcji, według której można sądzić o tym, czy wstawiany kołek sięgnął dna otworu.

Wkręcanie śruby

Jeżeli wkręcamy śrubę ze sfazowanym końcem, to operację wkręcania można rozpatrywać tak samo jak **pin_into_hole**. Oznacza to, że można postąpić jak w przypadku kołka, do chwili, gdy koniec śruby znajdzie się w płaszczyźnie wejścia do otworu gwintowanego. Następnie, za pomocą wkrętaka elektrycznego, należy lekko docisnąć śrubę i obracać ją. Podczas wkręcania śruby dobrze jest zapewnić sprzężenia zwrotne od siły. Pozwoli to stworzyć warunki, przy których siła reakcji, działająca na śrubę w kierunku poziomym, okaże się równa zeru. Nastąpi wtedy pożądane zgranie pozycji śruby i otworu gwintowanego. Makrookreślenie do wkręcania śruby może wyglądać nastepująco:

```
DEFINE screw_into
   ⊂MOVE hand TO @–VECT( 0,0,d )
         ON FORCE( zhat ) > f1 DO STOP hand;
COBEGIN
           MOVE hand TO @
            WITH FORCE( zhat ) = – fd;
            WITH FORCE( xhat ) = 0;
```

```
        WITH FORCE( yhat ) = 0;
      OPERATE driver WITH FTORQUE = fastening;
  ON TORQUE(driver) > T_d
      DO begin STOP driver; STOP hand; end;
COEND;⊃
```

Symbol @, znajdujący się w wierszu drugim i piątym, charakteryzuje pozycję i orientację ręki przed początkiem działania operatora MOVE. Za pomocą tego operatora zadaje się pozycję bieżącą jako proponowaną wartość wymaganą. W praktyce realizowane jest sterowanie ze sprzężeniem zwrotnym od siły, przedstawionym plikami WITH. Siły reakcji w kierunkach X i Y powinny przechodzić w zero, a w kierunku Z powinna być przykładana stała siła f_d. Działania te należy wykonywać równolegle z obracaniem wkrętaka. Obracanie wkrętaka kończy się wtedy, gdy moment obrotowy przekroczy wartość T_d.

Nasadzanie śruby na końcówkę wkrętaka

Sześciokątną końcówkę wkrętaka należy powiązać z gniazdem sześciokątnym we łbie śruby i, obracając końcówką, wprowadzić go w gniazdo. Wtedy można podnieść i przenieść śrubę na końcówkę wkrętaka. Operacja nasadzania śruby na końcówkę wkrętaka, nazywa się **bite_screw** (dosł. ugryź śrubę). Dla tej operacji można zrobić makrookreślenie i napisać je w następujący sposób:

```
DEFINE bite_screw
   ⊂COBEGIN
        MOVE hand TO @ WITH FORCE( zhat ) = f_1 ;
                    OPERATE driver WITH TORQUE = τ_1  fastening;
          ON DURATION = 1
             DO begin STOP hand; STOP driver; end;
   COEND;⊃
```

Program wykonania montażu

Zbudujemy teraz program wykonania montażu. Posłużymy się przy tym dotychczasową wiedzą i przedstawionym wcześniej porządkiem operacji montażowych. Aby program ten zapisać w postaci lakonicznej, wygodnie jest przedstawić w postaci makrookreśleń te teksty, które pojawiają się kilka razy.

```
DEFINE grasp($object,$grasp_force)
   ⊂CENTER($grasp_force) ; AFFIX $object TO hand ; ⊃
DEFINE c_grasp($object,$grasp_force)
   ⊂CENTER (10) ; OPEN hand TO 100 ;
          MOVE hand TO @*COORD(ROT( zhat,90 ), VECT(0,0,0)) ;
          CENTER($grasp_force) ; AFFIX $object TO hand ; ⊃
DEFINE release($object,$grasp_force)
   ⊂OPEN hand TO full_opening ; UNFIX $object FROM hand ; ⊃
DEFINE pick_screw($screw)
   ⊂bite_screw ; $screw←screw_feeder–VECT( 0,0,d ) ;
          AFFIX $screw TO driver ; ⊃
```

Z wykładu o wzajemnym oddziaływaniu sensorów dowiedzieliśmy się, że podczas chwytania obiektu i jego przekazywania potrzebne są relacje powiązań między układami współrzędnych. Powyższe makrookreślenia zawierają nie tylko działania lecz również opisy relacji powiązań między układami współrzędnych. Pozwala to zapisać program w postaci lakonicznej. Spotykany w poniższych programach operator MOVE można rozpatrywać jako ruch po odświeżanej trajektorii. Najpierw podniesienie pionowe w górę (bez zmiany orientacji ręki), a następnie przemieszczenie do pewnego punktu końcowego. Po tym następuje opuszczenie w dół, które także odbywa się bez zmiany orientacji do chwili, gdy zostanie osiągnięty punkt docelowy. Poniżej przedstawimy podstawowe elementy programu, który pozwala wykonywać niezbędne prace montażowe:

1. ustalanie przedramienia w imadle

```
MOVE hand TO link_grasp ;
grasp(link, grasp_force) ;
MOVE link TO vise ;
OPERATE vise WITH holding_force ;
release(link)
```

2. wstawienie dwóch kołków prowadzących (gp)

```
MOVE hand TO gp1 ;
c_grasp(gp1, force_p) ;
MOVE gp1_bottom TO hole1 ;
pin_into_hole ;
release(gp1) ;
MOVE hand TO gp2 ;
c_grasp(gp2, force_p) ;
MOVE gp2_bottom TO hole3 ;
pin_into_hole ;
release(gp2) ;
```

3. posadowienie napędu w gnieździe przedramienia za pomocą kołków prowadzących

```
MOVE hand TO du_grasp ;
c_grasp(du, force_du) ;
MOVE du TO link_bore ;
pin_into_hole ;
AFFIX du TO link ;
release(du) ;
```

4. nałożenie dwu śrub

```
MOVE hand TO driver_grasp ;
grasp (driver, force_d) ;
MOVE driver TO screw_feeder ;
pick_screw (bolt2) ;
MOVE bolt2 TO du_hole2 ;
screw_into ;
UNFIX bolt2 FROM driver ;
MOVE driver TO screw_feeder ;
pick_screw (bolt4) ;
MOVE bolt4 TO du_hole4 ;
screw_into ;
UNFIX bolt4 FROM driver ;
MOVE driver TO driver_rest ;
pin_into_hole ;
release (driver) ;
```

5. wyciągnięcie kołków ustalających

```
MOVE hand TO gp1 ;
c_grasp (gp1, force_p) ;
MOVE gp1_bottom TO gp1_rest ;
pin_into_hole ;
release(gp1) ;
MOVE hand TO gp2 ;
c_grasp (gp2, force_p) ;
MOVE gp2_bottom TO gp2_rest ;
pin_into_hole ;
release (gp2) ;
```

6. nałożenie pozostałych dwu śrub

```
MOVE hand TO driver_grasp ;
grasp (driver, force_d) ;
MOVE driver TO screw_feeder ;
```

pick_screw (bolt1) ;
MOVE bolt1 TO du_hole1 ;
screw_into ;
UNFIX bolt1 FROM driver ;
MOVE driver TO screw_feeder ;
pick_screw (bolt3) ;
MOVE bolt3 TO du_hole3 ;
screw_into ;
UNFIX bolt3 FROM driver ;

7. Mocne dokręcenie śrub

MOVE driver TO bolt1 ;
bite_screw ;
OPERATE driver WITH FTORQUE = final_fastening ;
AFFIX bolt1 TO du ;
MOVE driver TO bolt3 ;
bite_screw ;
OPERATE driver WITH FTORQUE = final_fastening ;
AFFIX bolt3 TO du ;
MOVE driver TO bolt2 ;
bite_screw ;
OPERATE driver WITH FTORQUE = final_fastening ;
AFFIX bolt2 TO du ;
MOVE driver TO bolt4 ;
bite_screw ;
OPERATE driver WITH FTORQUE = final_fastening ;
AFFIX bolt4 TO du ;
MOVE driver TO driver_rest ;
pin_into_hole ;
release (driver) ;

8. Wyjęcie zmontowanego zespołu z imadła i umieszczenie w zadanym miejscu

MOVE hand TO link_grasp ;
grasp (link, force_l) ;
OPERATE vise WITH OPENING = full_opening ;
MOVE link TO finished–VECT(0,0,10)
ON FORCE(*zhat*) > landing_force DO STOP hand ;
release (link) ;
MOVE hand TO home.

Podsumowanie wykładów przedstawiających robot jako system komputerowy

W wykładach tych spróbowaliśmy systematycznie wyłożyć podstawową wiedzę, którą można się posługiwać podczas budowy robota jako systemu programowalnego. Rozpatrzyliśmy robot jako system komputerowy, który: (1) może rozpoznawać otoczenie za pomocą „wzroku", „dotyku" i reagować na siły, oraz (2) manipulować obiektem za pomocą ręki mechanicznej. Rozpatrzyliśmy próbę przeistaczania się komputera – z urządzenia, obrabiającego dane abstrakcyjne, w urządzenie, zdolne działać w świecie rzeczywistym; w uniwersalną maszynę posiadającą funkcje rozpoznawania za pomocą takich nowych urządzeń wejścia-wyjścia, jak „oczy" i „ręce". Dzięki wykorzystaniu zbioru dokładnie wybranych rozkazów i systemu oprogramowania, przedstawiającego sobą umiejętne powiązanie tych rozkazów, komputer staje się urządzeniem uniwersalnym.

Podczas analizy i syntezy wiedzy robotycznej dążyliśmy przede wszystkim do uproszczenia wykładu i nadania mu kierunku praktycznego. W tym celu analizowaliśmy robot hipotetyczny. Przyczyną wybrania robota hipotetycznego była chęć stworzenia, na przykładzie prostych funkcji robotycznych, wyobrażenia całościowego, mającego znaczenie praktyczne. Mamy nadzieję, że pozwoliło to wytworzyć u słuchacza (czytelnika) konkretne wyobrażenia. Na przykładzie robota hipotetycznego wydzieliliśmy pewne minimum niezbędnych podstawowych funkcji i przeanalizowaliśmy je jako zintegrowany zbiór. Jest to zbiór podstawowych funkcji dotyczących sterowania ręką i dłonią, sensoryki dotykowej, siłowej i wzrokowej. Z punktu widzenia komputera zbiorowi temu odpowiada zestaw rozkazów. Następnie, w ramach kojarzenia i jednoczenia różnorodnych funkcji przeprowadziliśmy analizę programu sterującego procesami równoległymi. Różnorodne funkcje i zadania, określane jako kombinacje funkcji, można rozpatrywać jako procesy działające równolegle i realizować odpowiednie sterowanie nimi. Wykorzystując takie podejście przeanalizowaliśmy konkretny przykład powiązania czucia (sensoryki) działania (aktoryki). Przytoczyliśmy użyteczne, z punktu widzenia praktyki, przykłady programów z rozkazami sensorów. Wprowadziliśmy język robota w celu programowania pracy robotów. W języku tym zbudowaliśmy program pracy robota wykonującego montaż. Na koniec rozpatrzyliśmy operacje montażowe. Jako obiekt montażu wybraliśmy część ręki (przedramię) analizowanego robota hipotetycznego.