Krystian ŁYGAS, Wojciech DANILCZUK

STANOWISKO DO APLIKACJI PICK'N'PLACE Z WYKORZYSTANIEM SYSTEMU WIZYJNEGO I AUTORSKIEGO INTERFEJSU KOMUNIKACJI

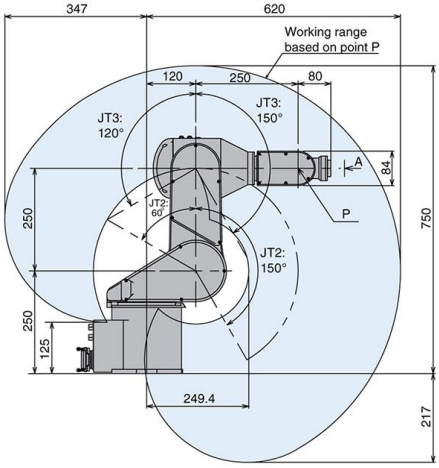
W artykule przedstawiono zaprojektowany oraz wykonany system wizyjny do aplikacji pick and place zgodny z kontrolerem robota Kawasaki. Aplikacja polega na wykonaniu zdjęcia przestrzeni w której znajdują się losowo rozłożone elementy do paletyzacji. Algorytm systemu wizyjnego (przygotowany w środowisku Adaptive Vision) określa współrzędne rozważanych elementów. Aplikacja napisana w języku Python ma na celu komunikację systemu wizyjnego z kontrolerem Kawasaki po interfejsie TCP/IP. Kontroler robota odpowiada za wykonanie operacji pobrania detali i paletyzacji oraz komunikację z komputerem PC. Wymiana informacji odbywa się po standardowych wejściach/wyjściach IO (wykorzystując protokół TCP/IP). Dzięki opracowaniu własnego interfejsu komunikacji autorom pracy udało się uzyskać kompatybilność między niezależnie działającym algorytmem systemu wizyjnego realizowanym na komputerze klasy PC a kontrolerem robota. Dodatkowo przygotowany przez autorów program typu pick'n'place oraz algorytm systemu wizyjnego cechują się dużym poziomem uniwersalności. Zastosowanie tej metody umożliwia wykorzystania technik wizyjnych i połączenie ich z niezależnie sterowanym robotem przy relatywnie niskiej cenie.

WSTĘP

Pick and place polega na przemieszczaniu przedmiotów spoczywających lub poruszających się przez automatyczny/zrobotyzowany system [1]. Położenia początkowe oraz końcowe przemieszczanego elementu są z góry znane lub określane przy pomocy systemu pomocniczego. Konfiguracja takiego systemu może przybierać różne formy, w zależności od użytej technologii. Jedną z form jest użycie obrazu cyfrowego, powstałego z obserwacji przestrzeni roboczej/ ruchu przemieszczanych przedmiotów. Wykorzystanie komputerowego przetwarzania obrazu, umożliwia dostosowanie uniwersalnego sprzętowego systemu wizyjnego do wszelakich aplikacji. Podejście takie umożliwia również przeprowadzenie wstępnej kontroli jakości poprzez odrzucenie obiektów nie spełniających wymagania specyfikacji. Autorzy opracowali autorski system wizyjny dla kontrolera E firmy Kawasaki. Algorytm systemu wizyjnego został przygotowany w środowisku Adaptive Vision [5]. Elementem wykonawczym systemu wizyjnego jest kamera internetowa co znacznie zmniejsza koszty stanowiska. Rozwiązanie takie wymagało opracowania autorskiego interfejsu do komunikacji pomiędzy komputerem PC przetwarzającym dane (zdjęcia) z kamery a kontrolerem robota. W tym celu wykorzystano język programowania Python. Komunikację uzyskano przez interfejs TCP/IP przy użyciu biblioteki dynamicznej (dll) KCT- dostarczonej przez producenta kontrolera robota.

1. KWASASKI RS003N

Robot Kawasaki RS003N, jest robotem z serii R, przeznaczonej do małych i średnich pod względem obciążeń aplikacji przemysłowych. Kształt oraz struktura kinematyczna została zaprojektowana w taki sposób aby robot ten znalazł zastosowanie w jak najszerszej gamie aplikacji. Cechuje się następującymi parametrami [7]: 6 stopni swobody, udźwig 3 kg, zasięg poziomy 620 mm, zasięg w pionie 967 mm, powtarzalność +-0.02 mm, maksymalna prędkość 6000 mm/s. Aplikacje do których wykorzystywany jest ten manipulator to najczęściej: montażu, paletyzacja, operacje przenoszenia (np. na stanowiskach obróbki skrawaniem), montaż uszczelnień.



Rys. 1. *Zasięg robota Kawasaki RS003N [7]*

Wymiary poszczególnych ramion oraz zakres pracy poszczególnych członów napędowych zostały przedstawione na Rys. 1. Kolorem ciemniejszym na rysunku przedstawiono przestrzeń roboczą robota, czyli zbiór wszystkich punktów w których możliwe jest umieszczenie końcówki roboczej manipulatora.

2. KONTROLER E70

Kontroler robota to sterownik który przeznaczony jest do:
- sterowania napędami (serwo sterowniki silników)
- odczytywania wartości z enkoderów,
- obsługi wejść/wyjść (IO)
- obsługi komunikacji w typowych dla przemysłu standardach (PROFINET, ETHERCAT, ETHERNET itp.)

Roboty stosowane w przemyśle opierają się zazwyczaj na tym samym modelu kinematycznym. Często zasięg robotów, udźwig i ich dynamika jest taka sama. O zastosowaniu robota do konkretnej

aplikacji decyduje natomiast kontroler. Inne sterowniki przeznaczone są do przenoszenia części (ang. *handling*) - operacji dla których ważne jest zatrzymanie końcówki roboczej robota (o dużej dynamice) w odpowiednim miejscu z zamierzoną dokładnością oraz powtarzalnością, zaś inne do aplikacji spawalniczych (ang. *welding*), gdzie ważne jest pod względem technologicznym odzwierciedlenie zadanej trajektorii ruchu.

Kontrolery E70, przeznaczone są dla małych robotów (RS03N, 05N, 05L, 06L, 10N). Cechują się dużym odzwierciedleniem zadanej trajektorii, szybkim wykonywaniem programu, jak również szybkim ładowaniem i zapisywaniem plików. Kontroler obsługuje pełne cyfrowe serwo systemy, język programowania AS, dotykowy Teach Pendant, 96 wejść/ wyjść IO. Waży 30 kg [7].

AS to system dzięki któremu programista może się komunikować z robotem lub tworzyć programy za pomocą języka o tej samej nazwie. [3].

System AS znajduje się w pamięci trwałej modułu sterującego robota. Umożliwia on sterowanie i kontrolę robota na dwa różne sposoby:

- poprzez polecenia wprowadzane bezpośrednio z poziomu teach pendant'a które są wykorzystywane do pisania, edycji i wykonywania programów,
- poprzez wykonywanie instrukcji programów zapisanych w pamięci robota które są wykorzystywane do realizacji zaprogramowanej trajektorii ruchu, nadzoru i kontroli zewnętrznych sygnałów. Ważne jest to, że program jest zbiorem instrukcji np. rozkazami ruchu poszczególnych osi.

Programowanie robota można przeprowadzić w dwojaki sposób. Pierwszy polega na generowaniu programu przy użycia teach pendanta (tzw. programowanie on-line). Druga metoda to przygotowanie programu robota na komputerze PC (np. w środowisku symulacyjnym K-ROSET dedykowanemu robotom Kawasaki) a następnie transfer programu do sterownika robota (tzw. programowanie off-line). Może być ono (przekazanie programu do pamięci kontrolera) przeprowadzane poprzez połączenie komputera PC z kontrolerem robota złączem ETHERNET i komunikacja wykorzystując protokół TCP/IP. Innym sposobem skopiowania przygotowanego off-line programu jest przeniesienie go poprzez nośnik pamięci USB (pen-drive'a).

3. BIBLIOTEKA KCT

3.1. OPIS FUNKCJI

Większość komend języka AS można wywołać przy pomocy terminala kontrolera, wykorzystując w tym celu aplikacje KCWINTCP. Istnieje możliwość opracowania własnego oprogramowania komunikacyjnego, współpracującego z kontrolerem robota, przy wykorzystaniu biblioteki dll. Biblioteka ta jest zbiorem funkcji które zostały opisane poniżej:

- AsKCTInit() - funkcja ta generuje proces systemu Windows o nazwie AsKCTCommu. Umożliwia on późniejsze nawiązanie komunikacji komputera klasy PC z systemem AS,
- AsKCTDestroy() - analogicznie jak AsKCTInit() z tym że funkcja ta zabija proces systemu Windows służący do komunikacji,
- AsKCTConnect() - funkcja służąca do nawiązania komunikacji z kontrolerem robota, wśród argumentów posiada adres IP kontrolera robota. Aby użyć tę funkcje wymagane jest wcześniejsze zainicjowanie funkcji AsKCTInit()
- AsKCTLoad() - wykonuje funkcję języka AS o nazwie load, która służy do wczytania programu z pamięci kontrolera,
- AsKCTSave() - wykonuje funkcję języka AS o nazwie save, która służy do zapisu programu do pamięci kontrolera,
- AsKCTCommand() - służy do zadania podstawowych funkcji języka AS, lub oddziaływania na wejścia wyjścia urządzenia,
- AsKCTCmdCheck() - służy do sprawdzenia rezultatu użytych komend, zwraca ewentualne błędy,
- AsKCTReply() - służy do wysłania wiadomości zwrotnej na zapytanie systemu AS,
- AsKCTVersion() - służy do sprawdzenia wersji KCT.

Pełną dokumentacja znajduje się w dokumentacji producenta [4].

Odpowiednie użycie funkcji, umożliwia interakcję z kontrolerem robota w trakcie pracy (wykonywania programu).

3.2. IMPLEMENTACJA BIBLIOTEKI

Python jest językiem programowania wysokiego poziomu, o rozbudowanym pakiecie bibliotek standardowych [6]. Głównie cechuje się tym że jego składnia jest przejrzysta i zwięzła. Jest rozwijany jako projekt Open Source zarządzany przez Python Software Foundation. Wspiera następujące paradygmaty programowania: obiektowego , imperatywnego oraz funkcyjnego. Posiada również zbiór funkcji i bibliotek numerycznych i technicznych, dzięki czemu zyskał popularność w środowiskach naukowych.

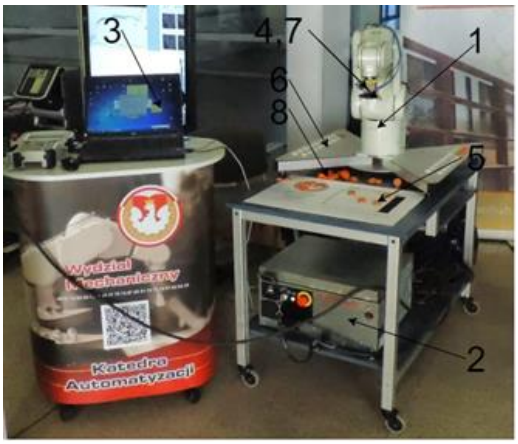
Implementację biblioteki ASCKT.dll do języka python uzyskano poprzez użycie funkcji ctypes.WinDLL. Jest to zewnętrzna funkcja biblioteki dla języka Python. Zapewnia kompatybilność typów danych C i umożliwia wywoływanie funkcji DLL lub bibliotek współdzielonych. Może być używana do wywoływania bibliotek dll w czystym Pythonie.

Kod programu napisany w języku python polega na wczytaniu biblioteki dll, a następnie odwołaniu się do nazw funkcji zawartych w bibliotece, jednocześnie definiując typ danych powrotnych, definiując atrybut restype. Podejście takie umożliwia zastosowanie funkcji biblioteki dll, które nominalnie przeznaczone były do języka Viusal Basic, w języku Python.

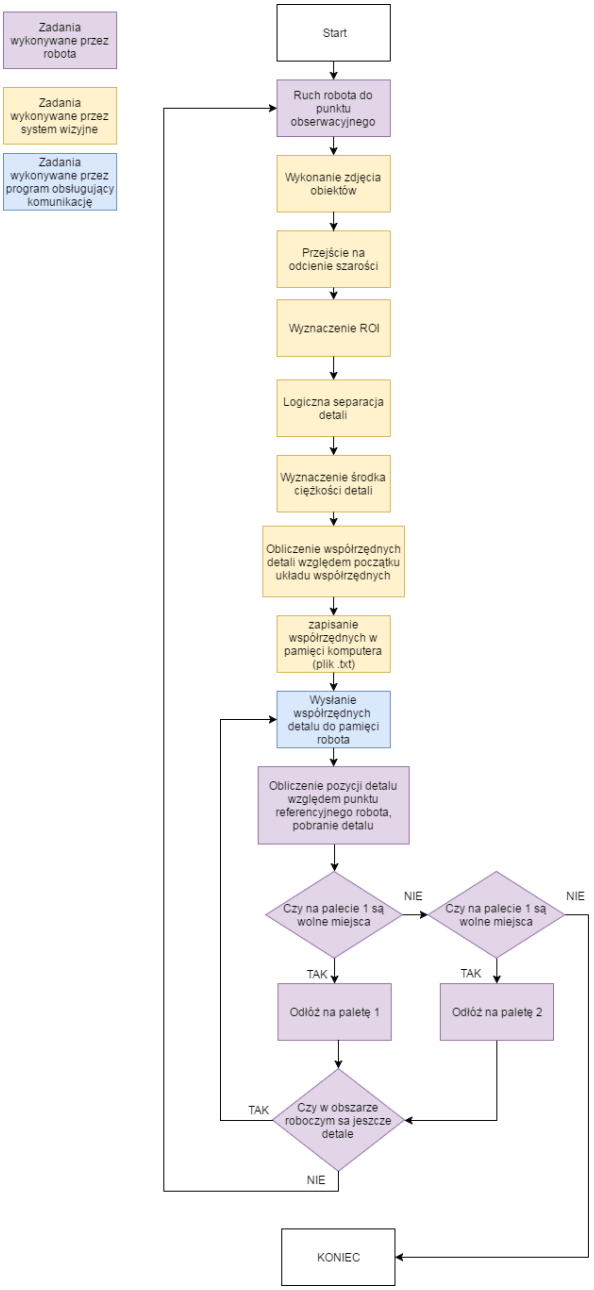
4. STANOWISKO DO PALETYZACJI DETALI

Koncepcję stanowiska oraz problematykę inspekcji przedstawia (Rys. 2). Stanowisko składa się z robota szeregowego Kawasaki RS003N (1), kontrolera Kawasaki E70 (2), komputera (3), kamery cyfrowej zamontowanej na korpusie robota (4), strefy akwizycji danych (obserwowany obszar z którego pobierane są detale) (5), strefy paletyzacji (6), chwytaka robota (7), oraz z pobieranych detali (8).

Założeniem projektowym było przygotowanie statycznej aplikacji pick'n'place (rozpoznającej, pobierającej i paletyzującej elementy będące w spoczynku). Zadanie stanowiska polega na rozpoznawaniu elementów znajdujących się w strefie akwizycji, obliczeniu ich współrzędnych, pobraniu, a następnie paletyzacji. Warto zwrócić uwagę, że rozpoznawane elementy dostarczane są w sposób losowy, a system ma za zadanie samodzielnie określić ich pozycję by móc dokonać operacji pobrania i paletyzacji.



Rys. 2. *Stanowisko badawcze*



Rys. 3. *Schemat algorytmu aplikacji*

Realizacja założonego zadania składała się z dwóch głównych etapów (Rys. 3) Pierwszym etapem było wykonanie procedury systemu wizyjnego. Algorytm rozpoznawania obrazu składał się z następujących kroków: wykonanie zdjęcia obszaru akwizycji, przejście na odcienie szarości, wyznaczenie ROI (ang. *Region-of-interest*), wyznaczenie początku układu współrzędnych na podstawie znacznika, logiczna separacja elementów, wyznaczenie środka ciężkości detali, obliczenie współrzędnych detali względem początku układu współrzędnych, zapisanie pozycji detali w pamięci komputera. Szczegółowy opis tych funkcji można znaleźć w [2] i [5]. Drugim etapem było przesłanie do kontrolera robota wyznaczonych pozycji detali oraz ruch manipulatora. Program obsługujący komunikację wysyłał współrzędne pierwszego detalu do pamięci robota. Po otrzymaniu informacji zwrotnej od robota o odłożeniu detalu na paletę program wysyłał kolejną współrzędną. Gdy robot pobrał wszystkie detale znajdujące się w obszarze akwizycji lub zapełnił obie palety wysyłał informację do komputera PC. W takiej konfiguracji nadrzędnym układem kontrolującym prace całego stanowiska był komputer, pełnił on rolę *master'a*, robot natomiast jako układ wykonawczy był *slave'm*.

Układ współrzędnych względem którego system wizyjny wyznaczał pozycję detali był również punktem referencyjnym robota Kawasaki. Sekwencja ruchów robota oraz wymiana sygnałów sterownik robota-komputer PC wyglądała w sposób następujący. Robot rozpoczyna ruch w punkcie w którym zamontowana na chwytaku kamera „widzi" cały obszar akwizycji. Następnie po otrzymaniu współrzędnych pierwszego detalu wykonuje operację obliczenia pozycji detalu względem znanego punktu referencyjnego, ruch do obliczonej pozycji, pobranie detalu, paletyzacja, wysłanie informacji zwrotnej do komputera PC z zapytaniem o współrzędne kolejnego detalu. Procedura obliczania i zapamiętywanie dostępnych na palecie miejsc i ich współrzędnych odbywa się w programie robota. Dodatkowo program robota wysyła do komputera informację o zapełnieniu się pierwszej lub drugiej palety oraz o pobraniu wszystkich dostępnych detali.

PODSUMOWANIE

Dzięki wykorzystaniu cyfrowych wejść/ wyjść kontrolera robota Kawasaki oraz protokołu komunikacyjnego TCP/IP udało się przygotować działająca i w pełni kompatybilną z autorskim systemem wizyjnym aplikacje statycznego pick'n'place. Rolą kontrolera, poza komunikacją z nadrzędnym komputerem PC było generowanie trajektorii ruchu robota oraz procedura paletyzacji detali. Ponadto stanowisko i program zostało przygotowane kompaktowo. Tradycyjnie w tego typu rozwiązaniach kamerę umieszcza się na zewnętrznym statywie. Umieszczenie kamery na chwytaku robota zimniejsza potrzebną przestrzeń. Dodatkowo program robota jest w dużej mierze sparametryzowany. Uruchomienie aplikacji wymaga nauczenia go tylko czterech punktów.

BIBLIOGRAFIA

1. Y. Sun, A. Behal, i C.-K. R. Chung, Red., New Development in Robot Vision, 2015 edition. New York: Springer, 2014.
2. L. Wojnar, M. Majorek, „Komputerowa analiza obrazu", Fotobit-Design, Kraków, 1994
3. Kawasaki Heavy Industries,LTD. Kawasaki Robot Controller E Series, Installation and Connection Manual (February 2016: 13th Edition)
4. Kawasaki Heavy Industries,LTD. Robot Division: Controller C Series, KcwinTCP-dll Instruction Manual (March 2003L 1st Edition)
5. Oprogramowanie dla przemysłowych systemów wizyjnych - Adaptive Vision, 2017, http://adaptive-vision.com/pl/
6. The official home of the Python Programming Language . 2017, http://www.python.org/

7. The official home of the Kawasaki Robotics, 2017 http://robotics.kawasaki.com

Pick'n'place application with the use of vision system and own communication interface

The article presents designed and manufactured vision system for pick and place applications compatible with the robot controller Kawasaki. The application procedure made a picture of the space in which are randomly distributed elements to palletizing. The algorithm of vision system (prepared in Adaptive Vision software) determines the coordinates of the component. The application written in Python communicate the vision system and E70 controller using TCP/IP interface . Robot controller is responsible for picking details and palletizing, as well as communication with a PC. Information exchanging is made by standard inputs / outputs (using TCP / IP protocol). By developing their own communication interface authors was achieved compatibility between independently operated vision system algorithm implemented on a PC and the robot controller. In addition, prepared by the authors program type pick'n'place algorithm and video system characterized by a high level of versatility. Using this method allows the use of vision techniques and combining them with independently controlled robot at a relatively low price.

Autorzy:
mgr inż. Krystian Łygas – Politechnika Lubelska, Wydział Mechaniczny, Katedra Automatyzacji, k.lygas@pollub.pl,
mgr inż. Wojciech Danilczuk – Politechnika Lubelska, Wydział Mechaniczny, Katedra Automatyzacji, wojciech.danilczuk@pollub.edu.pl.