Projekt oraz oprogramowanie zrobotyzowanego stanowiska do gratowania felg samochodowych

Paulina Pietruś, Magdalena Muszyńska
Politechnika Rzeszowska im. Ignacego Łukasiewicza, Katedra Mechaniki Stosowanej i Robotyki, Al. Powstańców Warszawy 8, 35-959 Rzeszów

Streszczenie: Coraz częściej w procesach produkcyjnych wykorzystuje się rozwiązania, w których roboty współpracują z systemami wizyjnymi. Wiąże się to z realizacją zadań typu ,,pick and place" lub korekcją ścieżki narzędzia w trakcie procesu obróbki. Systemy wizyjne wymieniają informacje z kontrolerami robotów, co umożliwia wykrycie określonego obiektu, uzyskanie informacji o jego lokalizacji i orientacji. W ramach artykułu zdecydowano się zaprojektować oraz zbudować zrobotyzowane stanowisko w środowisku RobotStudio przeznaczone do gratowania felg samochodowych. Przedstawiono proces projektowania algorytmu w środowisku MATLAB pozwalającego określić położenie i orientację obrabianego detalu. Komunikacja obu środowisk MATLAB oraz RobotStudio odbywa się przez protokół TCP/IP. Zaprezentowano także weryfikację działania oraz symulację zbudowanego stanowiska.

Słowa kluczowe: systemy wizyjne, MATLAB, protokół TCP/IP, RobotStudio, stacja zrobotyzowana, manipulatory

1. Wprowadzenie

Producenci robotów wprowadzają coraz nowsze rozwiązania sprzętowe, a także ulepszenia w obszarze oprogramowania. W oferowanych przez nich wirtualnych środowiskach istnieje możliwość między innymi zaprogramowania stanowisk zrobotyzowanych, symulacje uwzględniające cykle pracy oraz zaawansowaną logikę stanowisk. Środowiska są stale doposażane w kolejne moduły ułatwiające programowanie określonych procesów przemysłowych, takich jak spawanie, malowanie, zgrzewanie itd. Ponadto rozbudowywane są funkcje pozwalające na obsługę systemów wizyjnych oraz optymalizację programów sterujących. Niezbędną częścią w środowiskach programowania jest możliwość importowania plików CAD, dzięki czemu można odwzorować stanowisko rzeczywiste, znając odpowiednie wymiary komponentów [6].

Coraz częściej w procesach produkcyjnych wykorzystuje się rozwiązania, w których roboty współpracują z systemami wizyjnymi. Wiąże się to z realizacją zadań typu ,,pick and place" lub korekcją ścieżki narzędzia w trakcie procesu obróbki. Systemy wizyjne wymieniają informacje z kontrolerami robotów, co umożliwia wykrycie określonego obiektu, uzyskanie informacji o jego lokalizacji i orientacji.

Zgodnie z założeniami niniejszego artykułu obiektem poddawanym analizie jest felga aluminiowa, która powstaje w procesie odlewania, jako monolit. Ze względu na fakt wykorzystania wieloczęściowych form odlewniczych konieczna jest późniejsza obróbka ubytkowa w celu wykonania powierzchni montażowych, czy też usunięcia powstałych wypływek. Powstające wypływki mają nieregularny, niepowtarzalny kształt, co stanowi trudność tego procesu. Biorąc pod uwagę złożoną konstrukcję felgi, nie zawsze istnieje możliwość wykorzystania w tym celu obrabiarek sterowanych numerycznie, z kolei obróbka ręczna nie zapewnia wymaganej powtarzalności. Zastosowanie stacji zrobotyzowanej z systemem wizyjnym do procesu gratowania felg spełni swoją rolę, osiągając wymagane parametry procesu [11].

W artykule został również przedstawiony proces projektowania algorytmu pozwalającego określić położenie i orientację obrabianego detalu.

Autor korespondujący:
Paulina Pietruś, p.pietrus@prz.edu.pl

Artykuł recenzowany
nadesłany 29.11.2020 r., przyjęty do druku 18.02.2021 r.

2. Analiza istniejących rozwiązań

Gratowanie jest procesem, w wyniku którego zostaje określony ostateczny kształt oraz powierzchnia obiektu uzyskiwanego podczas obróbki tworzyw sztucznych bądź metalu. W wyniku tego procesu ostre krawędzie zostają zatępione, a zadziory i nierówności powierzchni usunięte. Ten typ obróbki mechanicznej jest szczególnie uciążliwy dla pracowników, ponieważ wymaga wysokiej jakości oraz powtarzalności [13, 17–19]. Gratowanie zaraz obok takich procesów jak polerowanie, cięcie oraz frezowanie to procesy podlegające szeroko pojętej robotyzacji. Większość firm produkujących roboty zapewnia klientom rozwiązania dedykowane właśnie do obróbki wykańczającej.

Obecnie technologia wizyjna nieustannie ewoluuje, a różnego typu zintegrowane systemy wizyjne stają się standardem. Systemy wizyjne dostarczają informacji dotyczącej otoczenia w postaci obrazu, dlatego często można spotkać się z porównaniem ich do „zmysłu wzroku". Za pomocą odpowiednich narzędzi

pozyskany obraz jest przetwarzany i wystawiana jest informacja zwrotna, przydatna w dalszych operacjach. Systemy wizyjne są stosowane w przemyśle między innymi w celu rozpoznawania części, identyfikacji wad oraz monitorowania danego procesu produkcyjnego. Zastosowanie wizji w procesie niesie ze sobą zalety, główne z nich to poprawa jakości produktu, usuwanie błędów, a także redukcja kosztów [9, 10].

W pracach [12, 13, 15] przedstawiono przykłady stacji zrobotyzowanych, które współpracują z systemami wizyjnymi oraz znajdują zastosowanie w procesie gratowania. Rozwiązania takie wykorzystują m.in. producenci robotów: FANUC, KUKA oraz ABB. Firma FANUC prezentuje pakiet Deburring Package we współpracy z robotem FANUC LR Mate 200iD. Deburring Package jest to połączenie systemu wizyjnego (FANUC iRVision) oraz czujnika siły (FANUC Force Sensor) [16]. System ten pozwala, w oparciu o zebrane informacje, na automatyzację obróbki detali bez konieczności programowania robota dla każdego z detali osobno. Drugim elementem pakietu Deburring Package jest czujnik siły, który daje możliwość kontroli siły w czasie operacji, chroniąc tym samym detal przed zniszczeniem. System pozwala na pomiar detalu przez wykrycie kontaktu pomiędzy narzędziem robota, a mierzonym detalem. Szerokie zastosowanie czujnika ogranicza również udział zewnętrznej aparatury kontrolno-pomiarowej [13, 16]. Pakiet Deburring Package może być wykorzystany m.in. w procesach gratowania.

Kolejny przykład przedstawia firma KUKA. Barriquan Robotic Deburring Cell firmy SUGINO to zautomatyzowana cela wyposażona w wrzeciono umieszczone na ostatnim ramieniu robota firmy KUKA. Oprócz 6-osiowego robota cela ma obrotowy stół pozycjonujący, który daje dodatkową siódmą oś, co jest przydatne do usuwania zadziorów najbardziej złożonych części [15]. Barriquan to uchwyt narzędziowy z wbudowanym mechanizmem ruchomym. Utrzymuje narzędzie pod stałym naciskiem na powierzchnie obrabiane, aby umożliwić profilowanie. Ciśnienie docisku ma regulację pozwalającą dopasować siłę docisku tak, aby spełnić wymagania i zlikwidować zadziory. Maszyna nadaje się do gratowania otworów w rurach, otworów lub krawędzi na powierzchni i od środka bez zmiany narzędzia [15].

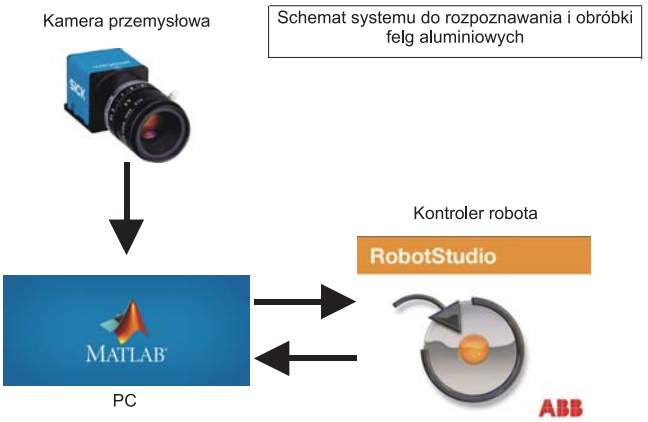
Następny przykład prezentuje firma ABB. Flex Finishing Cell firmy ABB to zrobotyzowana cela wyposażona w robota IRB 140 wraz z pakietem Force Control Machining. Czujnik siły zapewnia sprzężenie zwrotne podczas procesu obróbki, pozwala to na uzyskanie wysokiej jakości oraz powtarzalności. Cela wyposażona jest również w czujnik pozwalający na kalibrację punktu TCP [12]. Stanowisko przeznaczone jest do obróbki takich materiałów jak stal, stal nierdzewna, aluminium, magnez, tworzywa sztuczne oraz szkło. W zależności od konfiguracji mogą być wykonywane operacje polerowania, szlifowania, gratowania oraz frezowania. W pracach [1–4, 7, 8] przedstawiono przykłady zastosowania pakietu Force Control Machining do zatępiania krawędzi odlewów, usuwania naddatków oraz szlifowania łopatek. Celem sprawdzenia wykonanej obróbki detale są skanowane przez manipulator ze skanerem GOM.

3. Projekt oprogramowania w środowisku MATLAB

W celu budowy stanowiska zrobotyzowanego oraz jego oprogramowania wykorzystano algorytmy przetwarzania obrazu w programie MATLAB oraz język RAPID w środowisku RobotStudio.

3.1. Algorytm działania programu

Zaprojektowane stanowisko zrobotyzowane łączy współpracę dwóch środowisk – MATLAB/Simulink, pełniącego rolę systemu przetwarzania obrazu, oraz RobotStudio, w którym zamodelowano stację. Zastosowanie systemu przetwarzania obrazu umożliwia korekcję odpowiednich ścieżek ruchu robota. Na rys. 1

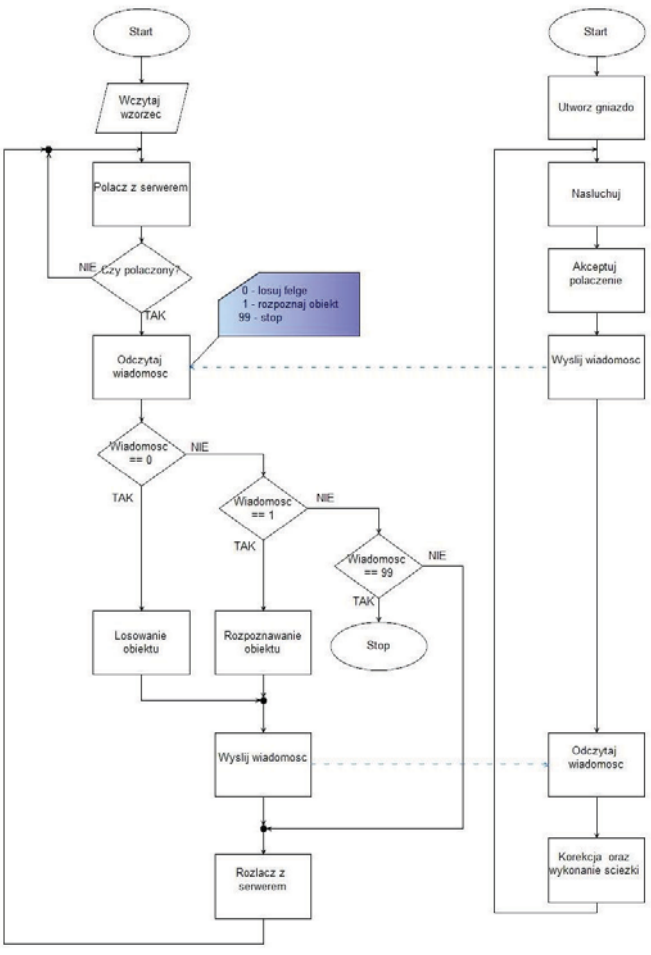


Rys. 1. Ogólny schemat komunikacji między środowiskami MATLAB oraz RobotStudio
Fig. 1. General diagram of communication between MATLAB and RobotStudio environments

przedstawiono schemat wymiany informacji między oprogramowaniem MATLAB oraz środowiskiem RobotStudio.

Aplikacja została zaprojektowana celem wyszukiwania, rozpoznawania i klasyfikacji przedmiotu na podstawie informacji odczytanych z obrazu. Kolejnym etapem jest wykonywanie obróbki elementu bazując na pozyskanych danych. Komunikacja między środowiskami odbywa się za pomocą protokołu TCP/IP. Na rys. 2 przedstawiono algorytm działania stacji.

Przedstawiony na rys. 2 algorytm dzieli się na dwie części. Lewa strona algorytmu dotyczy programu MATLAB, prawa strona – programu RobotStudio. Dokładny opis działania zaprezentowanego algorytmu zostanie przedstawiony w niniejszym oraz kolejnym rozdziale.



Rys. 2. Algorytm działania aplikacji
Fig. 2. Application operation algorithm

3.2. Opis działania programu

Program opracowany w ramach artykułu został zaprojektowany w środowisku MATLAB, zawiera algorytmy umożliwiające przetwarzanie obrazu, wykrywanie na nim obiektów oraz ich dopasowywanie. Przesyłanie informacji dotyczących między innymi orientacji oraz rodzaju przedmiotu zostało zrealizowane przy użyciu protokołu TCP/IP [5]. Odpowiednie dane zostają przesłane do kontrolera, który na ich podstawie aktualizuje ścieżkę ruchu robota. W m-pliku Main.m znajduje się kod programu odpowiadający za nawiązanie połączenia między środowiskami MATLAB – RobotStudio.

```
client = tcpip('127.0.0.1', 50001);
client.Timeout = 100;

while 1
  try
    fopen(client);
    flag = fread(client, 1);

    if flag == '0'
     Image = randi([1, 6]);
      switch Image
        case 1
          value = num2str(Image);
          [actualFeatures, actualPoints, originalImgAct,
                              newImgAct] = readPattern('Zdj/
                              audi_0.png');
          disp('Audi')
.
.
.
        case 6
          value = num2str(Image);
          [actualFeatures, actualPoints, originalImgAct,
                              newImgAct] = readPattern('Zdj/
                              honda_obr.png');
           disp('Honda_obr')
      end
      fprintf(client, value);
      fclose(client);

    elseif flag == '1'

      objectRecognition

      fprintf(client, ramka);
      fclose(client);
    end
  catch
    disp(,Oczekiwanie na połączenie...');
  end
end
```

W zaprezentowanym algorytmie działania program MATLAB pełni funkcję klienta. Posługując się metodą tcpip został utworzony obiekt o nazwie client klasy TCPIP, który przyjmuje adres IP 127.0.0.1 oraz numer portu 50001. Polecenie Timeout definiuje maksymalny czas oczekiwania na zakończenie operacji odczytu i zapisu, wyrażony w sekundach. Zastosowane dane są istotne do poprawnego działania komunikacji bazującej na protokole TCP/IP.

Następnie zastosowano nieskończoną pętlę while, która zapewnia ciągłość działania programu. Instrukcja fopen w pętli odpowiada za otwarcie połączenia z serwerem. Kolejnym krokiem jest odczytanie pojedynczego znaku, który przypisywany jest do zmiennej flag determinującej działanie dalszej części programu. W instrukcji warunkowej if obsługiwane są dwie wartości zmiennej flag. W przypadku, gdy jest równa zeru, uruchamiana jest metoda randi, która wraz z instrukcją switch odpowiada za losowanie jednego z sześciu modeli felg aluminiowych. Procedura ta została wprowadzona na potrzeby symulacji. Natomiast w sytuacji, gdy jest równa jeden zostaje wywołany m-plik objectRecognition.m. Zadaniem metody fprintf jest przesłanie wylosowanej wcześniej wartości do programu RobotStudio, gdzie odpowiedni model felgi zostanie umieszczony na przenośniku. W końcowej części programu metoda ta odpowiada za przesłanie ramki danych. Instrukcja fclose zamyka połączenie z serwerem.

Obsługa komunikacji między programami zawiera się w bloku instrukcji try...catch. Jeśli dowolna instrukcja w bloku try zwróci błąd, wykonywanie programu przechodzi natychmiast do bloku catch, który zawiera instrukcje obsługi błędów.

W m-pliku Main.m znajduje się również procedura wczytywania trzech różnych zdjęć wzorców w formacie .png. Na rys. 3 przedstawiono zdjęcia wzorcowe felg uzyskane na podstawie idealnych modeli 3D.

Rys. 3. Zdjęcia wzorcowe felg
Fig. 3. Model photos of rims

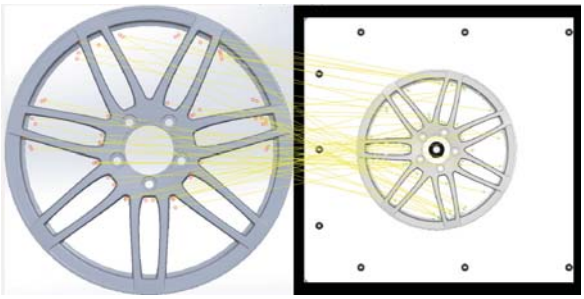
Do zdefiniowanych wzorców będą porównywane zdjęcia wykonane przez kamerę umieszczoną w stacji i na tej podstawie będzie rozpoznawany typ felgi, jej orientacja oraz położenie. Przesłanie informacji do kontrolera umożliwi rozpoczęcie procesu obróbki przez manipulator.

Po wczytaniu zdjęcia do pamięci programu przy użyciu metody imread, zostaje ono przekształcone z obrazu kolorowego na obraz w odcieniach szarości za pomocą funkcji rgb2gray. Zastosowane metody detectSURFFeatures oraz extractFeatures umożliwiają wykrycie oraz wyodrębnienie charakterystycznych punktów na zdjęciu. Na rys. 4 przedstawiono najmocniejsze charakterystyczne punkty każdego ze wzorców.

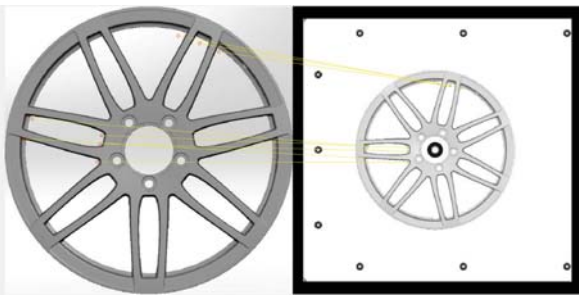
Rys. 4. Wykrycie stu charakterystycznych punktów dla każdego wzorca
Fig. 4. Detection of one hundred characteristic points for each pattern

W celu dopasowania obrazu wykonanego przez kamerę w stacji do jednego z wzorców napisano m-plik objectRecognition.m. Kod m-pliku zaprezentowano poniżej.

```
while 1
  idx = idx + 1;
  if idx > length (allPatterns(:,1))  %jeżeli nie dopasowano żadnego
                                      z 3 wzorców
    type = num2str(99);
    ramka = strcat(type, '|0|0|0')
    return;
  end
  pairs = matchFeatures(allPatterns{idx, 1}, actualFeatures,'Method',
        'nearestneighborsymmetric');
  if length(pairs) > 40
    type = idx;
    break;
  end
end
```



Rys. 5. Początkowe dopasowanie par punktów – po lewej wzorzec, po prawej zdjęcie z kamery
Fig. 5. Initial matching of pairs of points - on the left a pattern, on the right a photo from the camera



Rys. 6. Dopasowanie par punktów z usuniętymi wartościami odstającymi – po lewej wzorzec, po prawej zdjęcie z kamery
Fig. 6. Matching pairs of points with removed outliers – on the left a pattern, on the right a photo from the camera

Program rozpoczyna nieskończona pętla while. Pierwsza instrukcja warunkowa if definiuje obsługę błędu. Instrukcja zostanie uruchomiona, gdy algorytm nie dopasuje obrazu wykonanego przez kamerę do żadnego wzorca. Wówczas do programu RobotStudio zostanie wysłana ramka w postaci ‚99|0|0|0'.

Kolejnym poleceniem występującym w przedstawionym m-pliku jest metoda matchFeatures, która dopasowuje punkty charakterystyczne wykryte na wykonanym zdjęciu do punktów wykrytych na zdjęciu wzorca. Warunkiem koniecznym w celu doboru odpowiedniego wzorca, jest liczba dopasowanych par większa od czterdziestu. Na rys. 5 przedstawiono początkowe dopasowanie par punktów.

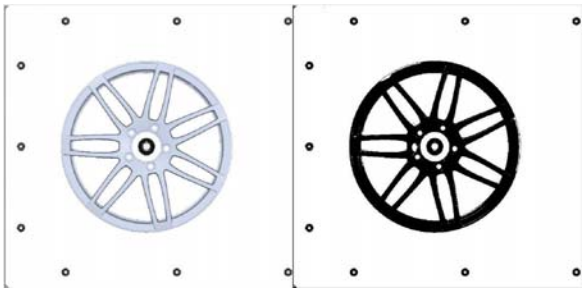
Kolejnym poleceniem w *m*-pliku objectRecognition.m jest instrukcja estimateGeometricTransform. Przedstawiona funkcja eliminuje wartości odstające od wcześniej dopasowanych par punktów. Funkcja przyjmuje następujące argumenty:

— similarity – typ transformacji, który przyjmuje minimalną liczbę dopasowanych par punktów równą dwa;
— Confidence – pewność znalezienia maksymalnej liczby par; zwiększenie tej wartości poprawia działanie algorytmu kosztem dodatkowych obliczeń;
— MaxDistance – maksymalna odległość w pikselach między punktami pary.

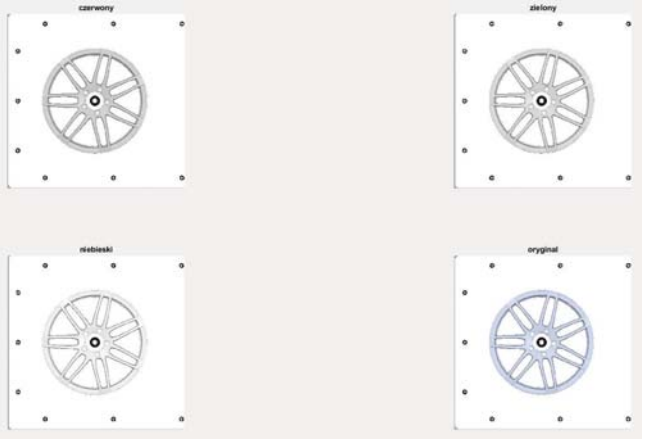
Wynikiem zastosowania polecenia estimateGeometricTransform dla początkowo dopasowanej pary punktów przedstawiono na rys. 6.

W celu poprawy jakości uzyskanego obrazu zastosowano przetwarzanie progowe. Przetestowano dwa podejścia, pierwsze z nich dotyczy konwersji zdjęcia do obrazu monochromatycznego, drugie – konwersji zdjęcia do palety barw RGB. Polecenie im2bw odpowiada za konwersję obrazu monochromatycznego na obraz binarny, zastępując wszystkie piksele macierzy wejściowej o wartości nasycenia większej niż ustalony poziom (zmienna level), wartością 1 (biały) oraz zastępując pozostałe piksele wartością 0 (czarny). Efekt progowania przedstawiono na rys. 7.

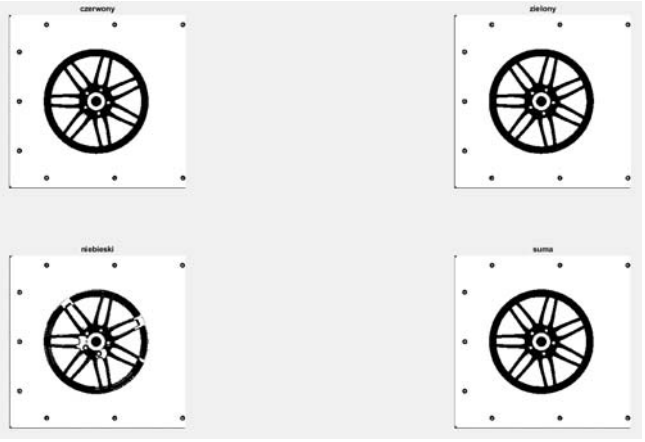
Zastosowanie tej metody zaowocowało nieco krótszym czasem obliczeń. Okazało się jednak, że użyty typ progowania nie zapewnia zadowalających rezultatów, co powoduje niedogodności w dalszej analizie obrazu. Z tego względu skorzystano z progowania, które przekształca zdjęcie kolorowe, biorąc pod uwagę nasycenie trzema kolorami RGB, na zdjęcie binarne. Na rys. 8. przedstawiono podział na składowe palety barw RGB.



Rys. 7. Obraz monochromatyczny przed i po progowaniu
Fig. 7. Monochrome image before and after thresholding



Rys. 8. Oddzielenie kolorów RGB
Fig. 8. RGB color separation



Rys. 9. Przekształcenie progowe RGB
Fig. 9. RGB threshold transform

Do przetwarzania progowego zostało użyte polecenie identyczne jak przy progowaniu obrazu w odcieniach szarości. Rysunek 9 przedstawia efekt progowania z użyciem palety barw RGB.

Poziomy doboru nasycenia palety barw zostały dobrane eksperymentalnie. W rozważanym przypadku zauważa się dokładniejsze wykrycie krawędzi przedmiotu w porównaniu z przekształcaniem obrazu monochromatycznego. Kolejnym krokiem jest wypełnienie luk wykrytych na otrzymanym powyżej obrazie binarnym. Na rys. 10 zaprezentowano efekt operacji wypełnienia ubytków.

Polecenie imcomplement odpowiada za uzupełnienie luk obrazu binarnego, zera zamieniają się na jedynki i odwrotnie. W celu wypełnienia ubytków znalezionych na zdjęciu zastosowano metodę imfill. Umożliwia ona dokładne zlokalizowanie krawędzi obiektu. Filtrację na podstawie kształtu obiektu przedstawiono na rys. 11.

Funkcja strel wykonuje podstawowe przekształcenia morfologiczne obrazu, takie jak erozja i dylatacja, wyszukując obiekty o promieniu większym od 30 pikseli, które następnie zostają wyeliminowane.

Kolejnym etapem było określenie środka wykrytego obiektu. Listing programu przedstawiający instrukcje pozwalające na wyznaczenie współrzędnych środka wykrytego obiektu zaprezentowano poniżej.

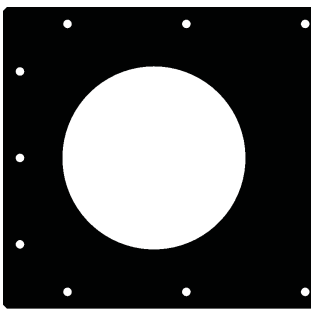
```
stats = regionprops(Iopenned, 'Centroid', 'MajorAxisLength',
        'MinorAxisLength');
diameters = mean([stats.MajorAxisLength stats.
        MinorAxisLength], 2)
radii = diameters/2;
```

Na rys. 12 zaprezentowano, w jaki sposób algorytm wykrywa kąt obrotu detalu względem wzorca.

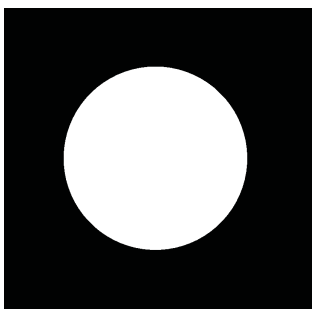
Za pomocą funkcji regionprops mierzone są właściwości danego obszaru obrazu. Jednym z jej argumentów jest 'Centroid', który odpowiada za pomiar współrzędnych środka ciężkości. Kolejne to 'MajorAxisLength' oraz 'MinorAxisLength', które oznaczają długość (w pikselach) większej i mniejszej osi elipsy. Za pomocą metody mean wyliczana jest średnica z wcześniej utworzonej macierzy i na tej podstawie wyliczany jest promień. Na rys. 13. przedstawiono poprawnie wykryty kształt przedmiotu.

Następnym krokiem było obliczanie kąta obrotu oraz skali przetwarzanego obrazu względem wzorca.

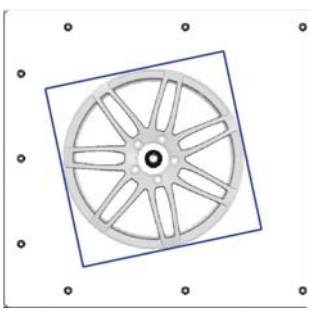
```
Tinv  = tform.invert.T;
ss = Tinv(2,1);
sc = Tinv(1,1);

wspSkal_px = sqrt(ss*ss + sc*sc);
wspSkal_mm = allPatterns{idx, 5} / size(allPatterns{idx,4}, 1);
wspSkal = wspSkal_px *  wspSkal_mm;

theta = atan2(ss,sc)*180/pi
```



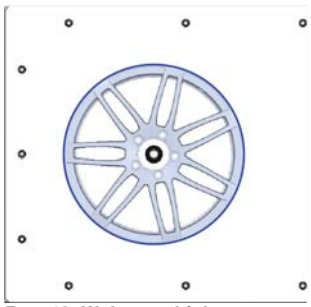
Rys. 10. Wypełnienie wykrytych luk na obrazie
Fig. 10. Filling detected gaps in the image



Rys. 11. Filtracja obrazu na podstawie kształtu
Fig. 11. Image filtering based on shape



Rys. 12. Wykryty obiekt – widać obrót obiektu o kąt theta
Fig. 12. Object detected - you can see the object's rotation by theta angle



Rys. 13. Wykryty obiekt
Fig. 13. Object detected

Wcześniej obliczona została macierz transformacji geometrycznej tform, którą należy przekształcić na macierz odwrotną, w celu obliczenia skali i kąta obiektu na obrazie [14].

Niech

$$sc = scale \cdot \cos(\theta), \tag{4.1}$$

$$ss = scale \cdot \sin(\theta), \tag{4.2}$$

Wtedy

$$Tinv = \begin{bmatrix} sc & -ss & 0 \\ ss & sc & 0 \\ tx & ty & 1 \end{bmatrix}, \tag{4.3}$$

gdzie tx i ty są odpowiednio przesunięciami w osi x i y układu współrzędnych.

$$wspSkal_{px} = \sqrt{ss^2 + sc^2} \tag{4.4}$$

$$wspSkal_{mm} = \frac{d_{mm}}{d_{px}} \tag{4.5}$$

gdzie d_{mm} oraz d_{px} oznaczają odpowiednio rzeczywistą średnicę felgi oraz wykrytego obiektu.

$$wspSkal = wspSkal_{px} \cdot wspSkal_{mm} \tag{4.6}$$

$$\theta = \operatorname{arctg}\left(\frac{ss}{sc}\right) \cdot \frac{180}{\pi} \tag{4.7}$$

Na tej podstawie został wyznaczony współczynnik skali wyrażony w pikselach oraz kąt θ w stopniach. Współczynnik został określony na podstawie zdjęcia z kamery oraz zdefiniowanego wzorca. Dodatkowo obliczony został współczynnik skali wymiaru rzeczywistego jako stosunek wymiaru rzeczywistego przedmiotu w milimetrach do wymiaru obiektu przedstawionego na wzorcu w pikselach. Całkowity współczynnik skalowania stanowi iloczyn wyżej wymienionych współczynników.

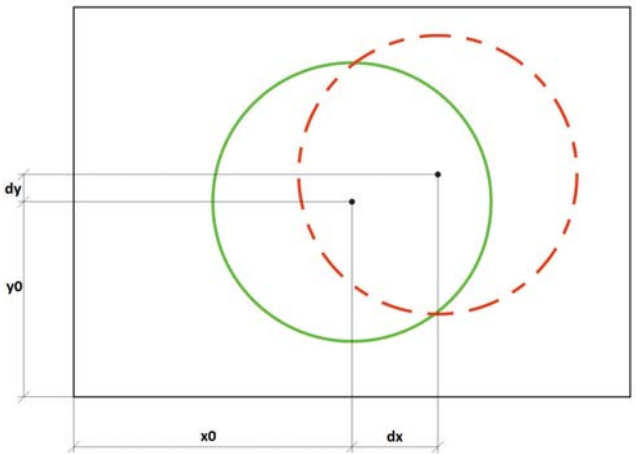
W następnej kolejności wyliczany jest środek obiektu (w mm) z uwzględnieniem całkowitego współczynnika skali.

```
center = stats.Centroid * wspSkal                % mm
x0 = 350; y0 = 350;

dx = (center(1) - x0)                            % mm
dy = (center(2) - y0)
```

Zmienne x0 oraz y0 przechowują wartości współrzędnych środka obszaru wykrywanego przez kamerę, są wyrażone w milimetrach. Na ich podstawie obliczane jest ewentualne przesunięcie obiektu. Na rys. 14 został przedstawiony sposób, w jaki określone zostają współrzędne środka obrabianego detalu. Czarnym kolorem został zaznaczony obszar reprezentujący granicę rejestrowanego przez kamerę obrazu. Kolorem zielonym zaznaczono obiekt w położeniu idealnym, natomiast czerwona linia oznacza rzeczywiste położenie obiektu, wykryte przez zastosowany algorytm. Pozycja jest więc wyznaczana jako różnica środka wykrytego obiektu oraz założonego środka, zarówno w osi x jak i y.

Ostatnim etapem algorytmu w programie MATLAB jest przygotowanie ramki danych tj.: typ felgi, kąt o jaki jest obrócona oraz przesunięta.

```
ramka1 = num2str(type);
ramka2 = num2str(theta);
ramka3 = num2str(dx);
ramka4 = num2str(dy);
ramka = {ramka1,ramka2,ramka3,ramka4};

ramka = strjoin(ramka, '|')
```



Rys. 14. Określenie współrzędnych środka obrabianego przedmiotu
Fig. 14. Determination of the coordinates of the workpiece center

Otrzymana ramka danych zostaje wysłana do kontrolera. Dane przesyłane z wykorzystaniem protokołu TCP/IP konwertowane są na ciąg znaków, w którym każdy z nich jest oddzielany znakiem „ | ”.

4. Projekt zrobotyzowanej stacji

W celu weryfikacji napisanego oprogramowania zostało zaprojektowane oraz zbudowane zrobotyzowane stanowisko w środowisku RobotStudio. Na rys. 15 przedstawiono widok stacji, która przeznaczona jest do gratowania felg samochodowych.

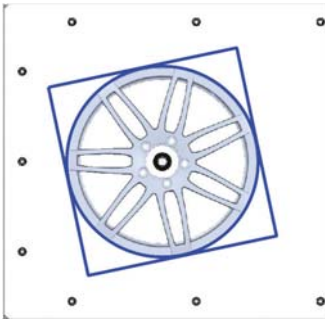
Stacja została wyposażona w trzy przenośniki, robot IRB 2400 oraz IRB 6620, kontroler, wrzeciono pneumatyczne, stół z uchwytem mocującym felgę podczas obróbki oraz systemem wizyjnym. Przenośniki, chwytak zamocowany na robocie IRB 6620, stół z uchwytem oraz stojakiem na kamerę zostały zaprojektowane w środowisku SolidWorks i zaimportowane do środowiska RS. Zastosowane wrzeciono pneumatyczne to narzędzie FDB 150 firmy SHUNK niezbędne do wykonania procesu gratowania felg.

4.1. Symulacja działania zbudowanego stanowiska

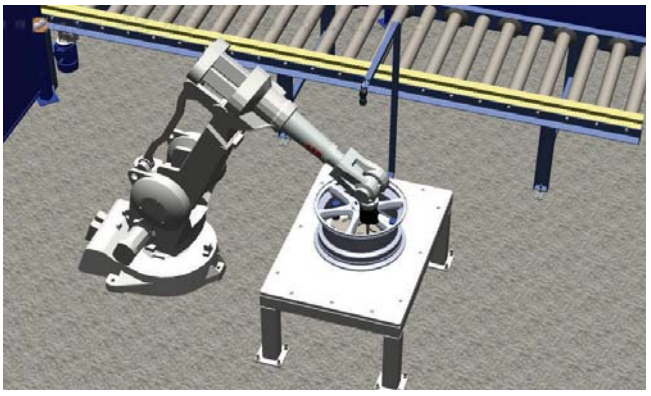
Zadaniem zbudowanej stacji zrobotyzowanej jest gratowanie felg samochodowych. Felgi dostarczane są na stanowisko przez przenośnik rolkowy. Kolejnym krokiem jest pobór felgi z przenośnika przez robota IRB 6620, który umieszcza ją na stole. Następnie jest ona mocowana na stole z wykorzystaniem uchwytu. Zbudowany w środowisku RobotStudio system wizyjny wykrywa detal. Zastosowanie systemu wizyjnego umożliwia uzyskanie informacji, tj.: typ felgi, jej orientacja oraz położenie, które zostają przesłane do kontrolera robota. Na podstawie uzyskanych informacji aktualizuje się ścieżkę ruchu robota. Obraz z kamery przedstawiono na rys. 16.

Uzyskanie informacji o feldze daje możliwość rozpoczęcia procesu gratowania przez robota IRB 2400 (rys. 17) wyposażonego w pneumatyczne, uchylne narzędzie FDB 150 firmy SCHUNK.

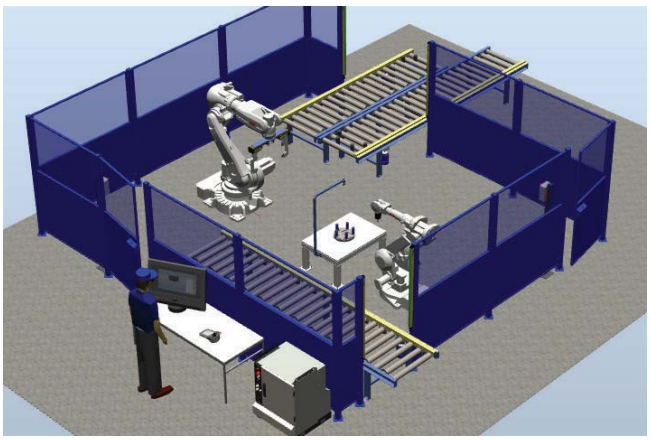
Po zakończonej obróbce przedmiot zostaje przeniesiony na transporter rolkowy, który transportuje ją poza obszar stanowiska. Trzeci przenośnik przeznaczony jest do transportu felg, które nie zostały dopasowane do żadnego ze zdefiniowanych wzorców.



Rys. 16. Obraz z kamery – kształt przedmiotu i kąt obrotu względem wzorca
Fig. 16. Camera image – the shape of the object and the angle of rotation relative to the pattern



Rys. 17. Robot IRB 2400 podczas gratowania jednej z felg
Fig. 17. Robot IRB 2400 deburring one of the rims



Rys. 15. Zamodelowane stanowisko do obróbki felg aluminiowych w programie RobotStudio
Fig. 15. Modeled station for machining aluminum rims in RobotStudio

4.2. Wyniki przeprowadzonych symulacji

Weryfikacja działania zbudowanego stanowiska została przeprowadzona na podstawie symulacji. W Tabeli 1 zaprezentowano wyniki sześciu losowań.

Na podstawie tab. 1 można stwierdzić, że algorytm poprawnie dopasowuje obiekty do wzorców. Zauważyć można trzy przypadki obróconych felg, o kątach 15, 20 i –12 stopni, które również zostały prawidłowo dopasowane. Niewielkie błędy mogą wynikać z niewystarczającej rozdzielczości zdjęć, a także z widocznych w tle cieni.

5. Podsumowanie i wnioski

W ramach artykułu została zaprojektowana oraz zbudowana stacja zrobotyzowana w środowisku RobotStudio, której zadaniem jest obróbka felg samochodowych. Stacja składa się z manipulatora ABB IRB 6620, narzędzia FDB 150, trzech

Tab. 1. Zestawienie wyników dla sześciu wylosowanych felg
Tab. 1. List of results for the six randomly selected rims

Lp.	Wylosowany typ felgi	Dopasowany wzorzec	Orientacja względem wzorca	Położenie x i y	Poprawność działania
1.	3	3	–0,0677	0,018402 -–0,022324	✓
2.	1	1	0,073402	0,0087616 0,0025452	✓
3.	2	2	14,9077	–0,025055 –0,12134	✓
4.	3	3	19,8799	–0,037552 –0,19585	✓
5.	2	2	0,019565	0,13666 –0,048462	
6.	1	1	–11,9341	–0,038156 –0,25451	✓

przenośników rolkowych, stołu z uchwytem mocującym felgę oraz manipulatora ABB IRB 2400 współpracującego z systemem wizyjnym. Zadaniem systemu wizyjnego było rozpoznanie typu felgi, określenie jej położenia oraz orientacji. Uzyskane informacje zostały przesłane do kontrolera robota, na ich podstawie rozpoczął się proces obróbki felgi. W celu obsługi systemu rozpoznawania obrazu w programie RobotStudio napisano program w środowisku MATLAB. Komunikacja między środowiskiem MATLAB i RobotStudio odbyła się przez standard komunikacji TCP/IP. W napisanym programie MATLAB pełnił on rolę klienta natomiast RobotStudio rolę serwera.

Działanie stanowiska zostało zweryfikowane na podstawie symulacji w środowisku RobotStudio. Poprawność działania zaprezentowanego oprogramowania została udowodniona poprzez wykonanie testów.

Zaprojektowane oprogramowanie może zostać rozbudowane o dodatkowe polecenia odpowiedzialne za funkcjonowanie systemu wizyjnego. Mowa tutaj o poszerzeniu asortymentu obrabianych elementów, czy też modyfikacji działania systemu wizyjnego tak aby dostarczał dodatkowych informacji podczas procesów obróbki.

Napisane oprogramowanie oraz jego weryfikacja na stanowisku zrobotyzowanym w środowisku RobotStudio wykazały celowość prowadzonych prac.

Bibliografia

1. Burghardt A., Szybicki D., Kurc K., Muszyńska M., Mucha J., *Experimental study of Inconel 718 surface treatment by edge robotic deburring with force control*. "Strength of Materials", Vol. 49, No. 4, 2017, 594–604. DOI: 10.1007/s11223-017-9903-3.
2. Burghardt A., Kurc K., Muszyńska M., Szybicki D., *Zrobotyzowane stanowisko z kontrolą siły*, „Modelowanie inżynierskie", T. 22, Nr 53, 2014, 30–36.
3. Burghardt A., Kurc K., Szybicki D., *Automatic Detection of Industrial Robot Tool Damage Based on Force Measurement*. "Tehnički vjesnik", Vol. 27, No. 5, 2020, 1385–1393. DOI: 10.17559/TV-20181110163252.
4. Burghardt A., Szybicki D., Kurc K., Muszyńska M., *Optimization of process parameters of edge robotic deburring with force control*. "International Journal of Applied Mechanics and Engineering", Vol. 21, No. 4, 2016, 987–995. DOI: 10.1515/ijame-2016-0060.
5. Burghardt A., Szybicki D., Pietruś P., *Zastosowanie architektury klient-serwer oraz protokołów TCP/IP do sterowania i monitorowania pracy manipulatorów przemysłowych*, „Modelowanie inżynierskie", T. 36, Nr 67, 2018, 16–22.
6. Kaczmarek W., Panasiuk J., Borys S., *Środowiska programowania robotów*, PWN, Warszawa 2017.
7. Kurc K., Burghardt A., Szybicki D., Gierlak P., Łabuński W., Muszyńska M., Giergiel J., *Robotic machining in correlation with a 3D scanner*. "Mechanics and Mechanical Engineering", Vol. 24, No. 1, 2020, 36–41, DOI: 10.2478/mme-2020-0003.
8. Obal P., Burghardt A., Kurc K., Szybicki D., Gierlak P., *Monitoring the parameters of industrial robots*. [In:] International Workshop on Modeling Social Media, Springer, Cham. 2018, 230–238, DOI: 10.1007/978-3-030-11187-8_19.
9. Tadeusiewicz R., *Systemy wizyjne robotów przemysłowych: rola, budowa, zastosowanie*, Zeszyty Naukowe AGH, Kraków 1989.
10. http://alnea.pl/systemy-wizyjne-w-robotyce/
11. http://alucenter.eu/pl/i/Jak-powstaja-alufelgi/
12. https://new.abb.com/products/robotics/pl/
13. https://roboforum.pl/artykul/inteligentne-gratowanie
14. https://www.mathworks.com/discovery/affine-transformation.html
15. https://suginocorp.com/barriquan-machine/
16. https://www.fanuc.eu/pl/pl
17. Islam M.M., Li C.P., Won S.J., Ko T.J., *A deburring strategy in drilled hole of CFRP composites using EDM process*. "Journal of Alloys and Compounds", Vol. 703, 2017, 477–485. DOI: 10.1016/j.jallcom.2017.02.001
18. Niknam S.A., Davoodi B., Davim J.P., Songmene V., *Mechanical deburring and edge-finishing processes for aluminum parts—a review*. "The International Journal of Advanced Manufacturing Technology", Vol. 95, No. 1, 2018, 1101–1125, DOI: 10.1007/s00170-017-1288-8.
19. Gillespie L.K., *Deburring technology for improved manufacturing* (No. GJBX-613-2573). Society of Manufacturing Engineers, Dearborn, MI. 1981.

Design and Software of a Robotic Station for Deburring Car Rims

Abstract: More and more often, production processes use solutions in which robots cooperate with vision systems. This is related to the implementation of “pick and place” tasks or tool path correction during the machining process. Vision systems exchange information with robot controllers, which enables the detection of a specific object, obtaining information about its location and orientation. As part of the article, it was decided to design and build a robotic station in the RobotStudio environment for deburring car rims. The process of designing the algorithm in the MATLAB environment that allows to determine the position and orientation of the processed detail was presented. Both MATLAB and RobotStudio environments communicate via the TCP/IP protocol. The verification of operation and simulation of the constructed station were presented.

Keywords: vision systems, MATLAB, TCP/IP protocol, RobotStudio, robotic station, manipulators

mgr inż. Paulina Pietruś
p.pietrus@prz.edu.pl
ORCID: 0000-0002-6428-0959

Urodziła się w Rzeszowie. Studia wyższe ukończyła na Politechnice Rzeszowskiej im. Ignacego Łukasiewicza. W roku akademickim 2016/2017 podjęła studia doktoranckie na Wydziale Budowy Maszyn i Lotnictwa Politechniki Rzeszowskiej. W tym samym roku rozpoczęła pracę w Katedrze Mechaniki Stosowanej i Robotyki, gdzie obecnie jest asystentem. Jej zainteresowania naukowe – szeroko pojęta mechatronika, programowanie robotów przemysłowych.

.

dr inż. Magdalena Muszyńska
magdaw@prz.edu.pl
ORCID: 0000-0002-0113-6159

Ukończyła studia w zakresie mechatronika na Politechnice Rzeszowskiej im. Ignacego Łukasiewicza w 2005 r. W roku akademickim 2005/2006 podjęła studia doktoranckie na Wydziale Budowy Maszyn i Lotnictwa Politechniki Rzeszowskiej. Od tego też roku jest pracownikiem Katedry Mechaniki Stosowanej i Robotyki, gdzie obecnie pracuje na stanowisku adiunkta. Stopień doktora nauk technicznych uzyskała w 2012 r. Jej zainteresowania naukowe dotyczą m.in. zagadnień sterowania robotami kołowymi z uwzględnieniem metod adaptacyjnych i inteligentnych. Jest współautorem publikacji krajowych i międzynarodowych.