Krystian ŁYGAS

ALGORYTMY SYSTEMU WIZYJNEGO STANOWISKA BADAWCZEGO APLIKACJI PICK AND PLACE

W artykule przedstawiono algorytmy oparte o bibliotekę funkcji przetwarzania obrazów OpenCV, w celu rozpoznawania oraz określenia lokalizacji oraz orientacji obiektów. Algorytmy zostały użyte na próbnym obrazie. Zadaniem algorytmu było określenie lokalizacji krążków znajdujących się na płycie podajnika, oraz klasyfikacji pod względem koloru. Autor artykułu poszukiwał rozwiązania cechującego się redukcją czasu trwania algorytmu.

WSTĘP

Widzenie maszynowe określa dział techniki, który realizuje różnego rodzaju zadania za pomocą programów analizy wizyjnej [1].

Autor książki [2], w rozdziale poświęconemu automatycznej inspekcji wizyjnej dzieli problem inspekcji na dwa rodzaje: sprawdzenie komponentu pod względem jakości oraz uzyskanie precyzyjnych informacji usytuowania komponentu, Oba rozważane rodzaje wymagają głównie tych samych wizyjnych algorytmów, a rodzaj zastosowanego urządzenia akwizycji obrazu może być zależny od tego czy rozważany obiekt znajduje się w spoczynku, czy w ruchu. Zastosowanie kamery liniowej może być bardziej użyteczne do inspekcji komponentów znajdujących się na transporterze, a zastosowanie kamery matrycowej w przypadku statycznym. Dobór urządzenia akwizycji obrazu ma wpływ na wielkość mierzonych obiektów oraz ich tolerancji wymiarowych. Widzenie maszynowe jest pojęciem ogólnym. W warunkach przemysłowych celem zautomatyzowanej inspekcji wizyjnej zwykle jest: weryfikacja jakości komponentów oraz wyrobów, sterowanie procesem, wspomaganie planowania produkcji.

Według autora książki [3] podstawowymi zadaniami systemów wizyjnych są: identyfikacja (weryfikacja) obiektów - rozpoznanie, jaki obiekt lub elementy składają się na oglądaną przez robota scenę.- lokalizacja obiektów i ich orientacja; zlokalizowanie obiektu pozwala robotowi na prawidłowe przechwycenie, oraz wygenerowanie trajektorii ruchu dojazdu do poszukiwanego obiektu. Bez lokalizacji za pomocą technik wizyjnych obiekty musiałyby być odpowiednio usytuowane w specjalnie zaprojektowanych gniazdach transportera w ściśle zdeterminowanej orientacji przestrzennej, a naruszenie tych warunków prowadziło by do błędnego działania systemu manipulacji; - określenie własności obiektów - przykładowo określenie kompletności części lub ich deformacji prowadzi do podstawowych operacji w procesie zautomatyzowanej kontroli jakości; -nawigacja i sterowanie robotem - na podstawie analizy obrazu uzyskanego z systemu wizyjnego umożliwia manipulatorowi ominięcie potencjalnych przeszkód i uniknięcie kolizji.

1. LOKALIZACJA OBIEKTÓW

Funkcją systemu wizyjnego jest rozpoznanie obiektów znajdujących się w przestrzeni roboczej. Wybór algorytmu lokalizacji na podstawie obrazu zależy od poszukiwanych własności obiektów [4].

W przypadku obiektów znajdujących się w spoczynku wyodrębnienie poszukiwanych własności, parametrów, cech zazwyczaj odbywa się poprzez zastosowanie technik przetwarzania obrazu w celu uzyskania odpowiednika w formie binarnej. Istnieje ku temu wiele metod, takich jak: progowanie, odfiltrowanie kolorów, wyodrębnienie krawędzi itp. Operacje zastosowane na uzyskanym obrazie takie jak: wyznaczenie środków ciężkości, dopasowanie minimalnego okręgu, dopasowanie minimalnego prostokąta, pozwalają na określenie regionów występowania obiektów. W dalszej analizie pozwala to na określenie własności takich jak kolor obiektu, orientacja itp., lub na dopasowanie wzorca.

W przypadku obiektów znajdujących się w ruchu istnieją techniki wśród których można wyróżnić: meanshift (przesunięcie średniej), optical flow (przepływu optycznego), background substraction (różnoicowania tła) [5]. Wymienione metody posiadają ograniczenia: metody background substraction oraz optical flow można stosować gdy obiekt porusza się na nieruchomym tle, metodę meanshift stosuje się w celu analizy ruchu wcześniej zlokalizowanego obiektu.

Zastosowanie metod lokalizacji obiektów znajdujących się w ruchu wymusza analizę obrazów niewiele się różniących, czyli próbkowanych w krótkim czasie. Okres próbkowania uzależniony jest od szybkości poruszającego się obiektu.

To że poszukiwany obiekt porusza się, wcale nie oznacza że do zlokalizowania go nie można zastosować metod statycznych.

Częstotliwość próbkowania (ilość obrazów na sekundę; fps) można dobrać na podstawie obserwacji procesu. W przypadku poszukiwania współrzędnych obiektów poruszających się na transporterze, częstotliwość próbkowania można dobrać na podstawie pomiaru prędkości ruchu obiektów oraz wielkości obserwowanej przestrzeni roboczej. Odpowiedni dobór częstotliwości próbkowania pozwala na redukcję przetwarzanych informacji, odciążając jednostkę obliczeniową [2].

1.1. Lokalizacja obiektów statycznych

Lokalizację obiektu można określić wyznaczając:

- środek ciężkości konturu (reprezentacji binarnej),
- położenie i orientację dopasowanego wzorca,
- położenie krawędzi obiektu.

We wszystkich wspomnianych przypadkach przed użyciem funkcji lokalizacji należy przetworzyć obraz w celu wyodrębnienia poszukiwanych parametrów, własności, cech[6].

1.2. Lokalizacja obiektów będących w ruchu

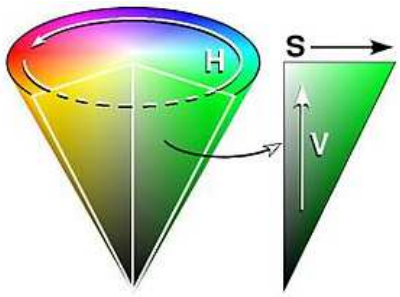
Optical Flow jest to termin który oznacza ekstrakcję ruchu pikseli na płaszczyźnie obrazu [5],[7]- tworzenie mapy złożonej z wektorów przyporządkowanych poszczególnym pikselom. Metoda rejestruje przesunięcia pikseli na kolejnych klatkach obrazu. Poprzez porównywanie kolejnych ramek obrazu wideo znajdywana jest korelacja pomiędzy nimi. Następnie tworzona jest tablica wektorów nazywana polem przepływu optycznego (optical flow field). Wektory te określają przesunięcie pikseli lub regionów, które dokonało się na skutek względnego ruchu obiektów i kamery.

Background substraction jest to metoda śledzenia ruchomych obiektów która wymaga statycznie umocowanej kamery [5]. Algorytm polega na różnicowaniu klatki obrazu z klatką poprzednią poprzednio z progowaną i odfiltrowaną. W wyniku czego otrzymuje się obraz wynikowy przedstawiający obszar przesuniętego obiektu.

2. METODY UZYSKANIA OBRAZU BINARNEGO

Metody uzyskania obrazu binarnego wyróżnione w podrozdziale:

- progowanie proste- najprostszy rodzaj polegajacy na przetworzeniu obrazu szarego w binarny poprzez przypisanie wartości intensywności do dwóch klas (np. 0 lub 255), na podstawie założonej wartości progu, dla której intensywność występowania odcieni szarości znajdująca się powyżej progu przyjmowała wartość 255, a poniżej wartość 0. Istnieją również typy jednostronne binearyzujące obraz powyżej lub poniżej wartości założonego progu.
- progowanie adaptacyjne- podobnie jak progowanie proste, wynikiem zastosowania procesu jest obraz binarny z różnicą taką że wartość progu wyznaczana jest na podstawie intensywności sąsiednich pól obrazu, dzięki czemu uwzględniane jest potencjalne występowanie obszarów poddanych zmiennym oświetleniu,
- zamiana na przestrzeń HSV- jest to metoda polegająca na konwersji obrazu kolorowego na obraz opisany modelem HSV (ang.Hue Saturation Value), który nawiązuje do tego jak widzi ludzkie oko. Zasadę działania przedstawia (Rys. 1)[8],



Rys. 1. Stożek modelu HSV [8]
gdzie: H- odcień wyrażony kątem na kole barw (0-360stopni),
S- nasycenie koloru – opisany jako promień podstawy,
V- moc światła jako wysokość stożka.

Przyporządkowanie częstotliwości fal świetlnych na kole barw w modelu HSV jest takie samo jak w modelach HSL, tzn. centrum barwy czerwonej odpowiada kąt 0° lub 360°. Centrum barwy zielonej odpowiada kąt 120°. Centrum barwy niebieskiej odpowiada kąt 240°. Pozostałebarwy pośrednie dla składowej Hue są odpowiednio rozłożone pomiędzy kolorami czerwonym, zielonym i niebieskim.

Wyodrębnienie koloru z obrazu polega na porównaniu każdego punktu macierzy obrazu z zakresami występowania rozważanego koloru w zapisie HSV. W przypadku gdy rozpatrywany punkt mieści się we wszystkich zakresach to do macierzy nowego obrazu przypisywana jest wartość 255, a w przeciwnym razie wartość 0. W taki sposób dobierając odpowiednio zakresy tablic, można wykryć dowolny kolor odzwierciedlony na obrazie cyfrowym.

3. METODA UZYSKANIA KRAWĘDZI

Jedną z metod detekcji krawędzi znajdujących się w obrazie jest algorytm Canny. Metoda wykorzystuje wielostopniowy algorytm umożliwiający detekcję wielu różnych krawędzi w obrazie [5].

Na kroki algorytmu składają się:

- redukcja szumu- detektor krawędzi Canny jest czuły na obecność szumu w surowym, nieobrobionym obrazie, więc w pierwszym etapie należy poddać obraz filtrem Gaussa, czego efektem powstaje lekko rozmazany obraz, nie dotknięty pojedynczymi zakłóceniami,
- poszukiwanie natężenia gradientu obrazu- ze względu na dowolny kierunek skierowania krawędzi, algorytm wykorzystuje cztery filtry do detekcji poziomych, pionowych, oraz przekątnych krawędzi na wygładzonym obrazie. Operatory detekcji krawędzi zwracają wartości pierwszej pochodnej dla kierunku poziomego (G_y) i kierunku pionowego (G_x). Nachylenie krawędzi oraz jej kierunek mogą być określone na podstawie zależności (1).

$$G = \sqrt{(G_x^2 + G_y^2)}, \theta = \arctan\left(\frac{G_y}{G_x}\right) \quad (1)$$

- usuwanie niemaksymalnych pikseli- polega na pocienianiu krawędzi, aby powstała ciągła linia o szerokości pojedynczego piksela,
- progowanie z histerezą- progowanie służy do usunięcia nieistotnych krawędzi które mają nachylenie poniżej ustawionego progu.

4. METODA POSZUKIWANIA WZORCA

Template Matching (dopasowanie szablonu) jest to technika cyfrowego przetwarzania obrazu służąca do dopasowania wzorców (szablonów) do większych obrazów. Technika polega na założeniu istnienia dwóch macierzy [9],[10]. Przykładowym może być metoda SAD (suma różnic absolutnych) stosowana dla obrazów zapisanych w odcieniach szarości. Metoda ta wykorzystuje macierz obrazu badanego o wielkości (x_s,y_s) oraz macierz wzorca (x_t,y_t). Obydwie macierze w każdym punkcie posiadają odpowiednią intensywność $I_s(x_s,y_s)$ i $I_t(x_t,y_t)$, odzwierciedlającą ich stopień szarości.

Różnica intensywności każdego punktu macierzy definiowana jest jako:

$$Diff(x_s, y_s, x_t, y_t) = |I_s(x_s, y_s) - I_t(x_t, y_t)| \quad (2)$$

$$SAD(x, y) = \sum_{i=0}^{T_{rows}} \sum_{j=0}^{T_{cols}} Diff(x+i, y+j, i, j) \quad (3)$$

Równanie (2) odnosi się do przeszukiwania każdego punktu macierzy w celu wykrycia najlepszego dopasowania, co można zapisać przez sformułowanie:

$$\sum_{x=0}^{S_{rows}} \sum_{y=0}^{S_{cols}} SAD(x,y) \quad (4)$$

gdzie: S_{rows} oraz S_{cols} są to rzędy oraz kolumny macierzy badanej, a T_{rows} oraz T_{cols} macierzy wzorcowej. W tej metodzie najmniejsza suma absolutnych różnic (SAD) daje najlepsze oszacowanie zbieżności pozycji badanego obrazu ze wzorcem.

5. POMIAR CZASU

W przypadku gdy określenie nowej pozycji obiektu znajdującego się w ruchu ze stałą prędkością polega na zmierzeniu czasu ruchu, dokładny pomiar czasu odgrywa ważną role.

Interpreter kodu Python posiada wbudowaną bibliotekę time [11], pomiar czasu polega na próbkowaniu kolejno w chwili początkowej oraz końcowej (Rys. 2). Przyrost czasu jest różnicą wcześniej próbkowanych zmiennych.

```
>>> # -*- coding: cp1250 -*-
>>> import time
>>> Poczatek=time.clock()
>>> print 'Pomiar czasu'
Pomiar czasu
>>> Koniec=time.clock()
>>> przyrostCzasu=abs(Koniec-Poczatek)
>>> print 'Mineło ' , przyrostCzasu, ' sekund'
Mineło  13.4689744387  sekund
>>>
```

Rys. 2. *Skrypt napisany w języku Python służący do określenia czasu wykonania programu*

6. OPENCV- BIBLIOTEKA FUNKCJI PRZETWARZANIA OBRAZÓW

Jak przedstawia oficjalna strona projektu [12], OpenCV jest biblioteką funkcji przetwarzania obrazu wydaną na licencji BSD, a więc darmową do użytku domowego jak i komercyjnego. Posiada interfejs C ++, C, Java i Python oraz obsługuje system Windows, Linux, Mac OS, iOS i Android. Biblioteka OpenCV została zaprojektowana dla dużych wydajności obliczeniowych oraz do pracy w systemach czasu rzeczywistego. Posiada możliwość programowania wielowątkowego. Z bibilioteki korzysta szacunkowo 47 tysięcy osób. Bibilioteka znalazła zastosowanie w zaawansowanej robotyce, jest jednym z modułów ROS (systemu operacyjnego robotów).

6.1. Progowanie proste przy pomocy funkcji z biblioteki Opencv.

Bibilioteka Opencv posiada funkcję progowania o nazwie cv2.Threshold()[13][14].

cv2.threshold(src,thresh, maxval, type[,dst]) → retval, dst (5)

gdzie: src - obraz źródłowy zapisany w odcieniach szarości,

- thresh- wartość progowa intensywności w zakresie [0-255],
- maxval- maksymalna wartość tablicy,
- type - metoda progowania,
- retval- status wykonania operacji,
- dst- obraz po konwersji.

W zależności od użytej metody progowania uzyskuje się inne rezultaty. Poniżej przedstawiono typy progowań oraz ich modele matematyczne:

- CV_THRESH_BINARY – binaryzacja z progiem thresh,

$$dst(x,y)=\begin{cases} maxVal & if\ src(x,y)>thresh \\ 0 & otherwise \end{cases} \quad (6)$$

- opis matematyczny przedstawia wyrażenie (6), zasadę działania przedstawia (Rys. 3.b),
- CV_THRESH_BINARY_INV negacja metody CV_THRESH_BINARY.

$$dst(x,y)=\begin{cases} 0 & otherwise \\ maxVal & if\ src(x,y)>thresh \end{cases} \quad (7)$$

- opis matematyczny przedstawia wyrażenie (7), zasadę działania przedstawia (Rys. 3.c)
- CV_THRESH_TRUNC – ograniczenie od gory od Threshold,

$$dst(x,y)=\begin{cases} threshold & if\ src(x,y)>thresh \\ src(x,y) & otherwise \end{cases} \quad (8)$$

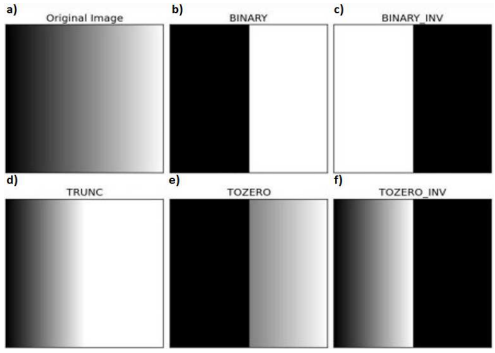
- opis matematyczny przedstawia wyrażenie (8), zasadę działania przedstawia (Rys. 3.d)
- CV_ THRESH_TOZERO – zerowanie poniżej Threshold

$$dst(x,y)=\begin{cases} src(x,y) & if\ src(x,y)>thresh \\ 0 & otherwise \end{cases} \quad (9)$$

- opis matematyczny przedstawia wyrażenie (9), zasadę działania przedstawia (Rys. 3.e)
- CV_THRRESH_TOZERO_INV- zerowanie powyżej Threshold

$$dst(x,y)=\begin{cases} 0 & otherwise \\ src(x,y) & if\ src(x,y)>thresh \end{cases} \quad 10$$

- opis matematyczny przedstawia wyrażenie (10), zasadę działania przedstawia (Rys. 3.f)



Rys. 3. *Graficzne przedstawienie działania metod funkcji cv2.Threshold() bibliotegki OpenCV; a)oryginalny obraz, b) metodą binary, c) odwrotną do metody binary, d) metodą Trunc, e) progowaniu metodą To Zero, f)metodą odwrotną do metody To Zero [13]*

6.2. Progowanie adaptacyjne przy pomocy funkcji z biblioteki Opencv

Bibilioteka Opencv posiada funkcję progowania adaptacyjnego o nazwie cv2.adaptive.Threshold() (11) [13], [14]

cv2.adaptiveThreshold(src,maxval,adaptiveMethod,threshold Type, bloksize,C[,dst]) →,dst (11)

gdzie:-src – obraz źródłowy zapisany w odcieniach szarości,

- thresh- wartość progowa intensywności w zakresie [0-255],
- maxval- parametr obrazu wyjściowego [0,maxval],
- adaptiveMethod- metoda adaptacyjnego progowania,
- thresholdtype – metoda progowania prostego,
- dst- obraz po konwersji,
- blokSize- macierz określająca rozważane sąsiedztwo pól,
- C- jest to stała która jest odejmowana od średniej lub obliczonej średniej ważonej.

Metody progowania adaptacyjnego:

- CV2.ADAPTIVE_THRESH_MEAN_C – wartość progowania jest średnią wartością sąsiednich pól,
- CV2.ADAPTIVE_THRESH_GAUSSIAN_C – wartość progowania jest ważoną sumą sąsiednich pól gdzie wartość wagi określana jest gęstością prawdopodobieństwa,

Zapis metod progowania prostego użytych w metodzie adaptacyjnej:

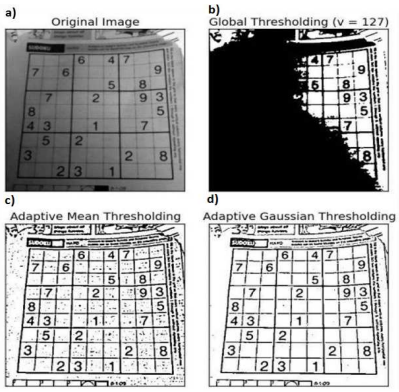
- CV2.THRESH_BINARY

$$dst(x,y)=\begin{cases} maxValue & if\ src(x,y)>T(x,y) \\ 0 & otherwise \end{cases} \quad (12)$$

gdzie: T(x,y) jest średnią powstałą z użycia metod adaptacyjnych,
- opis matematyczny przedstawia wyrażenie (3.8), zasadę działania przedstawia (Rys. 3.6.c)
- CV2.THRESH_BINARY_INV

$$dst(x,y)=\begin{cases}0 & if\ src(x,y)>T(x,y)\\ maxValue & otherwise\end{cases} \quad (13)$$

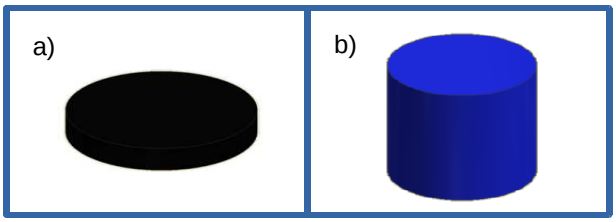
- opis matematyczny przedstawia wyrażenie (3.9), zasadę działania przedstawia (Rys. 3.6.d)



Rys. 4. *Graficzne przedstawienie działania metod progowania adaptacyjnego; a)obraz oryginalny, b)metoda Mean thresholding, c)metoda adaptive mean thresholding, metoda adaptive gaussian thresholding [13]*

7. PRZEDMIOT BADA

Przedmiotem badań jest opracowanie algorytmu wizyjnego służącego do określenia pozycji jak również rozróżniania kolorów krążków przedstawionych na (Rys. 5)



Rys. 5. *Obiekty detekcji; a) czarny krążek, b) niebieski walec*

Charakterystyka obiektów detekcji

a) czarny krążek (Rys. 5.a),

Parametry geometryczne:
- średnica podstawy 25 mm,
- wysokość 5 mm,
- kolor – czarny.

Wyróżnione cechy obiektu:
- lokalizacje: -(x_{cn}, y_{cn})- współrzędne środka ciężkości obiektu,
- wielkości:-r_{cn} – promień obiektu,-właściwości:
- właściwości: -kolor,

b) niebieski walec (Rys. 5.a),

Parametry geometryczne:
- średnica podstawy 30 mm,
- wysokość 30 mm,
- kolor – niebieski.

Wyróżnione cechy obiektu:
- lokalizacje: -(x_{cn}, y_{cn})- współrzędne środka ciężkości obiektu,
- wielkości: -r_{cn} – promień obiektu,
- właściwości: -kolor.

Problem inspekcji został przedstawiony na (Rys. 5.a,b). Próbka testowa składa się z niejednorodnych krążków różniących się kolorem. Zgodnie z wytycznymi poszczególnych próbek (Rys. 5.a,b), algorytm powinien wyznaczyć współrzędne krążków znajdujących się na transporterze (x_{cn}, y_{cn}), promieniem r_{cn} oraz znacznik koloru.

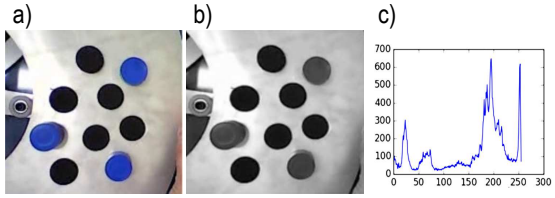
W celu uzyskania powyższego, postanowiono opracować algorytmy oparte na:
- progowaniu prostym jednostronnym,
- progowaniu obrazu HSV,
- poszukiwania wzorca .

Opracowano różne algorytmy (Rys. 8) w celu określenia najbardziej korzystnych pod względem minimalizacji ilości obliczeń, a więc czasu ich wykonania.

7.1. Implementacja alogrytmu progowania prostego

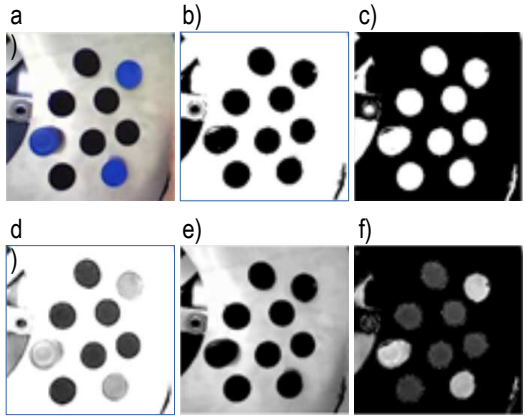
Implementowany algorytm przedstawia (Rys. 8), który został wykonany kolorem niebieskim.

Pierwszym etapem było oszacowanie wartości progowej. Dokonano tego poprzez wykreślenie histogramu intensywności obrazu po uprzedniej konwersji do odcieni szarości (Rys. 6.).



Rys. 6. *Rozpatrywany obraz; a)oryginał, b) w odcieniach szarości, c) histogram*

Na podstawie histogramu dobrano wartość progu intensywności 90 oraz dokonano progowania proste (Rys. 7).

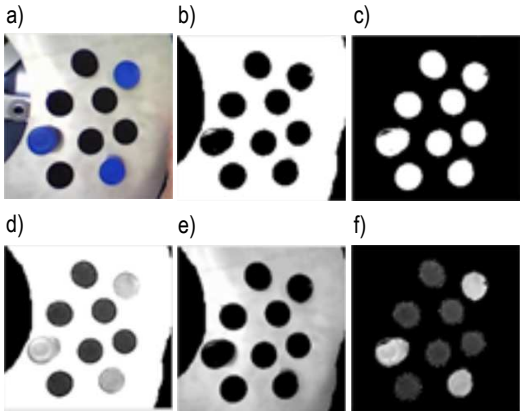


Rys. 7. *Progowanie proste dla wartości 90; a) obraz oryginalny, b) metodą „threshold binary”, c) metodą „threshold binaryinvert”, d) metodą „threshold trunc”, e) metodą „threshold to zero”, f) metodą ‘threshold to zero invert”*

Z powodu trudności usunięcia zakłóceń wynikających z obserwacji otoczenia transportera postanowiono na uzyskane wyniki (Rys. 7) nałożyć maskę, przez co uzyskano obraz bez zakłóceń wynikających z istnienia tła transportera (Rys. 9).

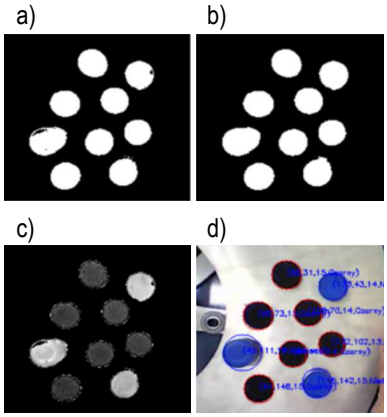
Wyniki zostały opracowane dla pięciu metod progowania w celu dobrania najlepszej. W rozważanym przypadku postanowiono poddać dalszej analizie obraz poddany metodzie Threshold Binary-Invert oraz Treshold To Zero Invert.

Rys. 8.*Zbiorcze zestawienie algorytmów wizyjnych. Kolorem niebieskim- algorytm oparty o progowanie proste, kolorem pomarańczowym - algorytm oparty o progowanie obrazu HSV, kolorem zielonym- algorytm oparty o poszukiwanie wzorca*



Rys. 9. *Progowanie proste z maską dla wartości 90; a) obraz oryginalny, b) metodą „threshold binary", c) metodą „threshold binaryinvert", d) metodą „threshold trunc", e) metodą „threshold to zero", f) metodą 'threshold to zero invert"*

Następnie użyto funkcję morfologiczną cv2.morphologyEx() do wyeliminowania czarnych obszarów znajdujących się wewnątrz wyodrębnionych obiektów (Rys. 10.a). Wynik zastosowania tej metody przedstawia (Rys. 10.b).



Rys. 10. *Przedstawienie wyników algorytmu; a) obraz po progowanie, b) obraz po operacji zamknięcia, c) obraz po progowaniu jednostronnym, d) obraz z nałożonymi wynikami*

W celu wyodrębnienia współrzędnych poszukiwanych obiektów postanowiono użyć funkcji cv2.findCounturs() wykrywającej kontury, a następnie cv2.minEnclosingCircle() dopasowującą najmniejszy okrąg do rozważanego obiektu zwracającą współrzędne, oraz promień obiektu. Na tym etapie uzyskano współrzędne (x_{cn}, y_{cn}) oraz wartość promieni r_{cn} bez rozpoznania własności ich koloru.

W celu rozpoznania koloru postanowiono obliczyć średnią wartość sąsiadujących punktów obrazu poddanego progowaniu metodą threshold To Zero Invert (), wykorzystując do tego wyznaczone współrzędne. Uzyskaną wartość poddano progowej klasyfikacji, dzięki czemu otrzymano znacznik koloru . W celu wizualizacji otrzymanych wyników () wykorzystano funkcję cv2.circle() oraz cv2.putTest() służące do rysowania odpowiednio okręgów oraz tekstu. Przy pomocy funkcji time.clock(), dokonano pomiaru czasu wykonania algorytmu (Tab. 1).

Tab. 1. *Wynik pomiaru czasu wykonania algorytmu*

Czas wykonania algorytmu	0.012 s

7.2. Implementacja algorytmu progowania obrazu HSV

Implementowany algorytm przedstawia (Rys. 8), który został wykonany kolorem pomarańczowym. W pierwszym etapie dokonano konwersji obrazu do przestrzeni koloru HSV (Rys. 11.b).

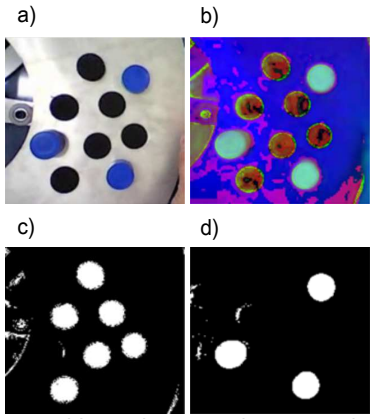
W celu wykrycia koloru niebieskiego oraz czarnego opracowano w sposób eksperymentalny zakresy występowania tych barw. Dla koloru czarnego :

$$H_{cz}[0:179]S_{cz}[0:200]V_{cz}[0:73] \tag{13}$$

Dla koloru niebieskiego:

$$H_{n}[0:179]S_{n}[125:255]V_{n}[52:255] \tag{14}$$

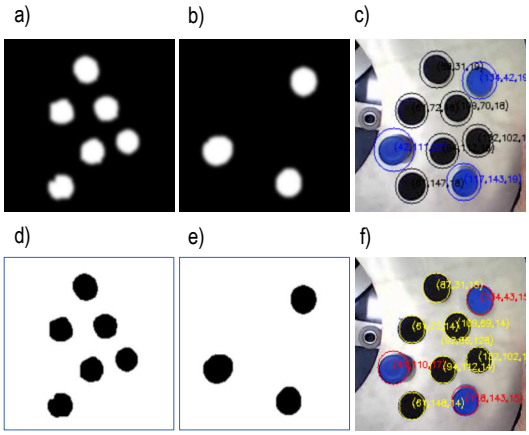
Przy pomocy funkcji cv2.inrange poddano obrazy binearyzacji o wytyczne zakresów (13), (14) (Rys. 11.c,d).



Rys. 11: *Obrazy poddane algorytmowi progowania HSV; a) obraz oryginalny, b) obraz w przestrzeni barw HSV, c) obraz uzyskany przez progowanie koloru czarnego, d) obraz uzyskany przez progowanie koloru niebieskiego*

Wynik progowania (Rys. 11.c,d) posiada zakłócenia, wynikające z istnienia przestrzeni o zbliżonym kolorze do poszukiwanego, co w rozważanym przypadku wiąże się z nieuniknioną obserwacją otoczenia transportera.

Postanowiono uzyskany obraz poddać operacjom erozji, dylatacji oraz funkcji cv2.GaussianBlur() polegającej na rozmyciu (uśrednieniu) wartości sąsiadujących pikseli przedstawia (Rys. 12.a,b).



Rys. 12.*Obrazy poddane funkcjom wizyjnym; a,b) po podwójnym zastosowaniu operacji erozji oraz rozmyciu, c) wynika na podstawie algorytmów dopasowania okręgów, d,e) po dokonaniu progowania obrazów (a-b) metodą „thresh binary invert", f) wynik końcowy algorytmu.*

Użyto funkcję do wykrycia konturów cv2.findContours() oraz funkcję cv2.minEnclosingCircle() służącą do dopasowania do uzyskanych konturów najmniejszych okręgów. Wynik algorytmu przedstawia (Rys 12.c), dla którego przy pomocy funkcji rysujących wskazano wyszukane cechy. Jak można zauważyć wskazane okręgi posiadają znacznie większą średnicę od rzeczywistej. Stwierdzono że uzyskany błąd jest wynikiem zastosowania funkcji rozmycia, dlatego zastosowano progowanie proste metodą Thresh Binary_Invert w celu uzyskania reprezentacji binarnej obrazu (Rys. 12.d,e). Ponownie opracowano wyniki oparte na nowym obrazie (Rys. 12.f). Aproksymacja okręgów poprawiła się, aczkolwiek zauważalne są odchyłki aproksymacji w miejscach gdzie występuje cień rzucany przez obiekt rozpatrywany. Wartości znajdujące się obok sklasyfikowanego obiektu to kolejno (x_{cn},y_{cn},r_{cn}) . Znacznik koloru został zastąpiony reprezentacją koloru na obrazie.

Przy pomocy funkcji time.clock(), dokonano pomiaru czasu wy konania algorytmu (Tab. 2).

Tab. 2. *Wynik pomiaru czasu wykonania algorytmu*

Czas wykonania algorytmu	0.22 s

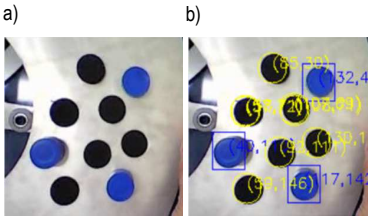
7.3. Implementacja algorytmu poszukiwaniu wzorca

Implementowany algorytm przedstawia (Rys. 8.), który został wykonany kolorem zielonym. W pierwszym kroku opracowano obrazy wzorcowe. Ze względu na mały rozmiar obiektów obrazy wzorców nie zostały ukazane. Następnie użyto funkcji cv2.matchTemplate() do wykrycia poszukiwanych obiektów. Użyta funkcja zapisała dopasowania wzorca do obrazu przeszukiwanego w formie listy. Odnaleziono większą ilość dopasowań niż istnieje obiektów. Opracowano funkcję (Rys. 13.) służącą do filtrowania obiektów podwójnych.

```
for pt in zip(*loc[::-1]):
    if (pt[0]-parametr<pp and pt[0]+parametr>pp and pt[1]-parametr<bb and pt[1]+
parametr>bb):
            continue
    else:
        b.append((pt[0],pt[1],0))
        pp=pt[0]
        bb=pt[1]
```

Rys. 13. *"Wycinek programu" w języku Python, filtrujący dopasowania obrazów*

Funkcja tworzy nową listę odnalezionych obiektów, filtrując występowanie kolejnych w tolerancji 30 pikseli.Problem pojawia się w momencie gdy kolejne dopasowania wędrują pomiędzy dwoma obiektami. Przypadek taki powoduje podwójne zaklasyfikowanie obiektu, co przedstawia wynik algorytmu (Rys. 14). Ze względu na podwójne dopasowanie obiektów, opracowany algorytm nie nadaje się do zastosowania w systemie wizyjnym. Przy pomocy funkcji time.clock(), dokonano pomiaru czasu wykonania algorytmu (Tab. 3).



Rys. 6. *Wynik algorytmu dopasowania wzorca*

Tab. 3. *Wynik pomiaru czasu wykonania algorytmu*

Czas wykonania algorytmu	0.08 s

8. WYNIKI

Podsumowanie zastosowanych metod przedstawia (Tab. 4).

Tab. 1. *Podsumowanie rezultatów zastosowanych algorytmów wizyjnych*

Metoda	Czas wykonania algorytmu [s]	Dopasowanie
Progowania prostego	0.012	Prawidłowe
Progowania HSV	0.022	Prawidłowe
Dopasowania wzorca	0.08	Nieprawidłowe

Biorąc pod uwagę zastosowane metody, algorytm oparty na progowaniu prostym ze względu na uzyskane odpowiednie dopasowanie oraz najniższy czas wykonania algorytmu najlepiej spełnił założenia systemu. Algorytm oparty na progowaniu obrazu HSV ze względu na zastosowanie funkcji erozji oraz rozmyciu powoduje zniekształcenie (zwiększenie) rozpoznawanego elementu, co może się przyczynić do odrzucenia obiektu/produktu pod względem nie spełnienia wymogów specyfikacji. Algorytm oparty na poszukiwaniu wzorca, pod względem nadmiarowości uzyskanych wzorców nie nadaje się do zastosowania. Zauważono również, że czas jego trwania wyniósł niemal ośmiokrotnie więcej niż progowania prostego.

PODSUMOWANIE

Opracowano trzy algorytmy przetwarzania obrazu służace do określenia lokalizacji krążków, oraz ich klasyfikacji pod względem koloru. Wszystkie algorytmy zostały opracowane przy pomocy biblioteki funkcji przetwarzania obrazów OpenCV. Dokonano pomiarów czasów wykonania operacji. Najlepszym dopasowaniem oraz najkrótszym czasem trwania cechowła się metoda progowania prostego. Zastosowanie metody poszukiwania wzorca, ze względu na nadmiarowość uzyskanych zlokalizowanych obiektów nie spełniła wymogów. Zaś metoda progowania obrazu HSV może dawać niedokładne wyniki wynikajace z operacji erozji oraz rozmycia.

Przedstawione algorytmy można użyć do operacji rozpoznawania elementów znajdujących się w ruchu.

BIBLIOGRAFIA

1. Zbigniew Wawerek, „Machine vision, widzenie maszynowe albo...", *PAR*, nr 4, ss. 18–19, 2008.
2. E. R. Davies, *Computer and Machine Vision: Theory, Algorithms, Practicalities*. Academic Press, 2012.
3. R. Tadeusiewicz, *Systemy wizyjne robotów przemysłowych*. Wydawnictwa Naukowo-Techniczne, 1992.
4. J. R. Parker, *Algorithms for Image Processing and Computer Vision*, 2 edition. New York: Wiley, 2010.
5. A. Kaehler i G. Bradski, *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*, Second Edition edition. Beijing ;Köln: O'Reilly Media, 2015.
6. D. L. Baggio *i in.*, *Mastering OpenCV with Practical Computer Vision Projects*. Birmington, UK: Packt Publishing, 2012.
7. Paweł Melnarowicz, „System wizyjny do analizy ruchu i położenia obiektów". Politechnika Wrocławcka, 2008.
8. „HSV (grafika) – Wikipedia, wolna encyklopedia". [Online]. Dostępne na: https://pl.wikipedia.org/wiki/HSV_(grafika). [Udostępniono: 30-cze-2015].

9. H. R. Myler, *The Pocket Handbook of Image Processing Algorithms in C*. PTR Prentice Hall, 1993.
10. T. P. Zieliński, *Cyfrowe przetwarzanie sygnałów: od teorii do zastosowań*. Wydawnictwa Komunikacji i Łączności, 2007.
11. „OpenCV dokumentacja, time". [Online]. Dostępne na: https://docs.python.org/2/library/time.html. [Udostępniono: 29-cze-2015].
12. „Strona internetowa projektu OpenCV". .
13. „Poradnik OpenCV Progowanie obrazu", *GitHub*. [Online]. Dostępne na: https://github.com/abidrahmank/OpenCV2-Python-Tutorials. [Udostępniono: 29-cze-2015].
14. J. E. Solem, *Programming Computer Vision with Python: Tools and algorithms for analyzing images*, 1 edition. Sebastopol, CA: O'Reilly Media, 2012.

Algorithms of vision system of research position of pick and place application

This article presents algorithms based on the OpenCV image processing library, to recognize and determine the location and orientation of objects. Algorithms were used in the image test. The purpose of the algorithm was to determine the location of the discs on the tray and the color classification. The author was looking for a solution about the shortest time of duration

Autor:

mgr inż. Krystian Łygas – Politechnika Lubelska, Wydział Mechaniczny, Katedra Automatyzacji, k.lygas@pollub.pl