Sterowanie robotem na bazie rozpoznanej barwy obiektu

Adam Makarewicz

Wydział Elektryczny, Politechnika Białostocka

Streszczenie: Celem pracy jest praktyczna analiza tematyki z zakresu sterowania robotem na podstawie przygotowanego algorytmu przetwarzania obrazu fotograficznego oraz jego realizacja w postaci aplikacji w języku C++ umożliwiającej sterowanie urządzeniem zewnętrznym podłączonym do komputera stacjonarnego.

Słowa kluczowe: sterowanie robotem, rozpoznanie koloru, przetwarzanie obrazu, matematyczny zapis obrazu

DOI: 10.14313/PAR_203/119

1. Wstęp

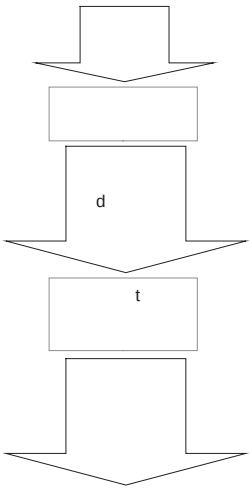
Działanie układów na bazie rozpoznanej barwy obrazu, umożliwia zaproponowanie algorytmów stanowiących łańcuch decyzji w sterowaniu układami elektronicznymi robota. Współczesne języki programowania komputerów umożliwiają zapis danej przestrzeni barw do postaci zrozumiałej dla określonego układu elektronicznego mogącego sterować danym urządzeniem. Choć zakres znanych człowiekowi odcieni kolorów jest ograniczony zakresem widm fal elektromagnetycznych, to ilość możliwych przeznaczeń odcieni w zapisie numerycznym jest wciąż nieograniczona.

2. Wprowadzenie i główne wyniki

Obraz świata zewnętrznego wprowadzany jest do komputera za pomocą urządzeń optycznych, mogących przetwarzać zewnętrzny widok z określoną rozdzielczością.

W zależności od sposobu zapisu analizowanego obrazu w kodzie maszynowym komputera rozróżnia się grafikę rastrową jako zbiór pojedynczych pikseli oraz wektorową, w której obraz powstaje w wyniku kombinacji geometrycznych równań matematycznych. Przetwarzanie obrazu sprowadza się do określenia algorytmu, na wejściu którego podawany jest obraz widziany z poziomu urządzenia wizyjnego, natomiast na jego wyjściu otrzymuje się decyzję jako rozwiązanie narzuconego zadania (rys. 1).

Cyfrowe urządzenia optyczne mają skończoną rozdzielczość a obraz pozyskiwany jest jako tzw. *grafika rastrowa* (ang. *raster graphics*), czyli w postaci siatki złożonej z jednokolorowych pikseli, tworzących jednolity wzór (rys. 2). Proporcję obrazu k zapisuje się zależnością:



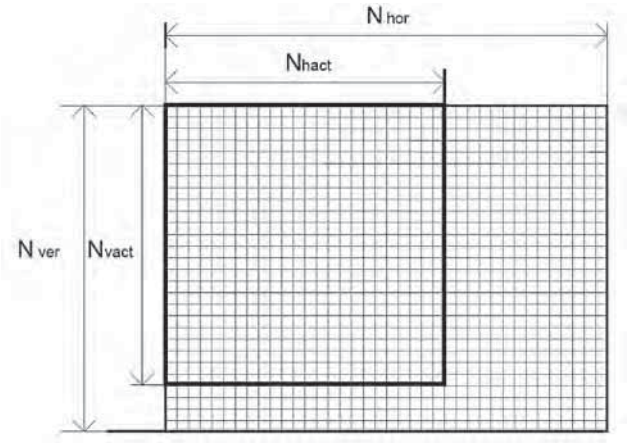
Rys. 1. Ogólna interpretacja przetwarzania obrazu w komputerze [2]

Fig. 1. The overall interpretation of image processing on the computer [2]

$$k = \frac{N_{hor}}{N_{ver}}, \tag{1}$$

gdzie N_{hor}, N_{ver} oznaczają liczbę widocznych na ekranie linii poziomych i pionowych.

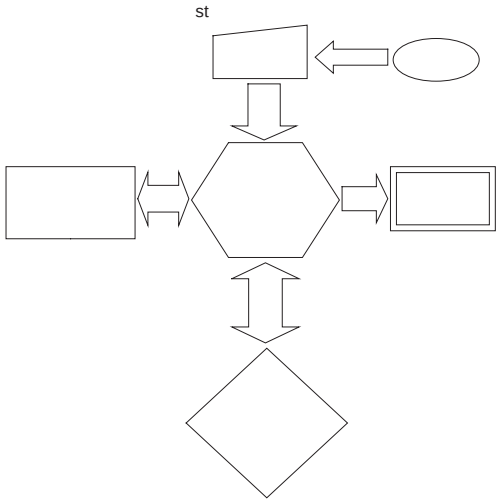
Celem sztucznego przetwarzania (rys. 3) lub analizy obrazu jest takie automatyczne przetworzenie i przeanalizowanie obrazu wybranych obiektów lub całego otoczenia systemu zautomatyzowanego, aby uzyskać użyteczną



Rys. 2. Struktura siatki (rastra) typowego obrazu cyfrowego [2]

Fig. 2. The structure of the grid (raster) of a typical digital image [2]

informację na temat interesujących obiektów (na przykład będących przedmiotem manipulacji ze strony robota przemysłowego) lub na temat otoczenia, które może wpływać na sterowany automatycznie proces.



Rys. 3. System cyfrowego przetwarzania obrazu [2]
Fig. 3. The system digital image processing [2]

Rys. 4. Sterowanie pojazdem samochodowym (robotem mobilnym) na bazie rozpoznanej barwy obrazu [1]
Fig. 4. Control of motor vehicle (mobile robot) based on the identified color image [1]

Element obrazu cyfrowego, tzw. *piksel*, w układzie dwuwymiarowym ma postać skwantowaną, daną zależnością:

$$f = f(x, y);\ x = 0, 1, 2, \ldots, N\text{–}1;\ y = 0, 1, 2, \ldots, M\text{–}1$$

gdzie: N, M – szerokość i wysokość obrazu.

Można stwierdzić, że na proces sztucznego widzenia składa się szereg operacji (rys. 3), takich jak:
- recepcja (akwizycja) obrazu;
- przetwarzanie obrazu (filtracja, eliminacja zakłóceń itp.);
- rozpoznanie obrazu i jego semantyczna interpretacja.

Przechwycenie obrazu za pomocą kamery optycznej w czasie rzeczywistym umożliwia wyodrębnienie z otoczenia określonych fragmentów przestrzeni, które następnie można poddać analizie przez oprogramowanie z poziomu komputera.

Na obrazie w postaci płaskiej tablicy zmiennych, której każdy obiekt stanowi niepodzielny piksel o własnym kolorze można przeprowadzić bardzo wiele operacji matematycznych, z których każda stanowi dodatkowy podprogram w obrębie programu głównego.

Zaproponowana aplikacja rozpoznawania kolorów na podglądzie pozyskiwanym z kamery internetowej bazuje na zapisie tablicowym pikseli jako bitmapy do przechowania rastrowej postaci obrazów. Program został napisany w środowisku *Microsoft Visual C++*

Zadanie polegało na sterowaniu pojazdem po zadanej, jednobarwnej trasie (rys. 4). Obiekt sterowania przedstawiono na rys. 5.

Obiektem sterowania jest robot mobilny, będący modelem samochodu marki BMW z zainstalowaną na masce standardową kamerą internetową, podłączaną do komputera za pomocą złącza USB.

Rys. 5. Obiekt sterowania wykorzystany do demonstracji programu rozpoznającego kolor (model plastikowy auta marki BMW model X5 w skali redukcyjnej 1:10) [1]
Fig. 5. The control object used for the demonstration program to identify the color (plastic model the car BMW X5 model in 1:10 scale reduction) [1]

Sterowanie robota mobilnego odbywa się za pomocą urządzenia elektronicznego własnej konstrukcji, zbudowanego na bazie mikrokontrolera PIC 16F627A, zadaniem którego jest odebranie sygnału sterującego, wysyłanego z komputera za pomocą interfejsu łącza szeregowego RS-232, a następnie przetworzenie go na ciąg impulsów zrozumiałych dla silnika prądu stałego stanowiącego napęd robota mobilnego. Dodatkowo, wspomniane urządzenie współpracuje z układem zdalnego sterowania stanowiącego wyposażenie zakupionego samochodu zabawki i pośredniczy tym samym w głównym sterowaniu, jakim jest sterowanie robota mobilnego na podstawie rozpoznanej barwy obrazu. Sterowania odbywa się na podstawie

rozpoznanej barwy obrazu, jaki przekazuje do komputera kamera optyczna zamontowana na masce modelu. Analiza obrazu przebiega w sposób pokazany na rys. 6.

Zadaniem podprogramu sterowania pojazdem jest włączenie każdego ze sterów kierunku jazdy, jak jazda prosto, skręt w lewo czy prawo tylko w określonym z trzech obszarów, którego rozmiar określono dzieląc poziomo rozmiar okna na trzy jednakowe obszary.

Rys. 6. Podział obrazu (320 × 240 pikseli) na trzy strefy aktywności sterów kierunku jazdy [1]

Fig. 6. Image partitioning (320 × 240 pixels) on the three zones of activity of the rudders drive [1]

Wielkość każdego obszaru zawiera się w następujących przedziałach – strefach, których wymiary określono w pikselach: „I" (0,107), „II" (107,214), „III" (214,320).

Jak widać, tablica zmiennych została podzielona na trzy podtablice, w których każdemu z podobszarów przypisano oddzielne kierunki jazdy pojazdu.

Jeżeli wynik rozpoznania koloru pojawi się w lewej części tablicy, układ sterowania załączy stery jazdy `OnLewo()` oraz `OnProsto()`, natomiast w pozostałych częściach tablicy proces sterowania przebiega analogicznie.

Podprogram sterowania pojazdem – algorytm, tworzy następujący kod źródłowy:

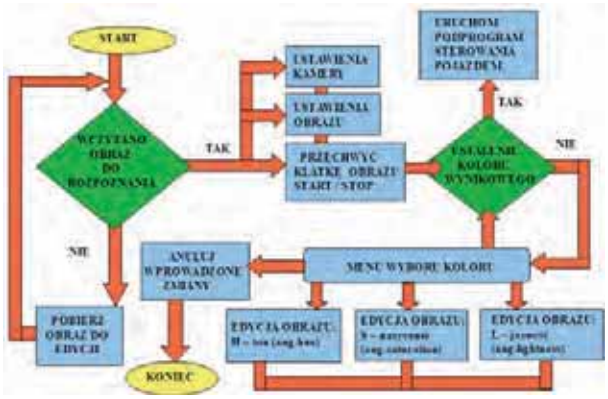
```
if(avgX >0 && avgX <107) //skręć w LEWO
{
   m_decyzja = „lewo”;
   wynik = ‚i’;
}

if(avgX >107 && avgX <214) //jedź prosto
{
   m_decyzja = „prosto”;
   wynik = ‘o’;
}
if(avgX >215 && avgX <=320) //skręć w PRAWO
{
   m_decyzja = „prawo”;
   wynik = ‘p’;
}
```

Algorytm stanowiący łańcuch decyzyjny poruszania się pojazdu po jednobarwnym torze ilustruje rys. 7.

Widok aplikacji, w której zaimplementowano algorytm analizy obrazu i sterowania pojazdem (rys. 7), zrealizowanej za pomocą oprogramowania *Microsoft Visual C++*, zaprezentowano na rys. 8.

Prezentowany interfejs działa w oparciu o obraz w postaci mapy bitowej o rozmiarze w pikselach (320 × 240) jako wynik chwilowej rejestracji kamery optycznej. Pierwsze okno obsługi programu stanowi oryginalny obraz pobrany z kamery internetowej, natomiast w drugim oknie zawarty jest wynik analizy obrazu pod



Rys. 7. Algorytm działania programu analizy obrazu i sterowania pojazdem [1]

Fig. 7. The algorithm of the program of image analysis and control of the vehicle [1]

Rys. 8. Interfejs analizy obrazu i sterowania ruchem pojazdu [1]

Fig. 8. Interface of image analysis and motion control of the vehicle [1]

kątem rozpoznanego koloru obiektu. Interfejs obsługi pojazdu przystosowany jest do ciągłej reakcji na zmianę obrazu pobieranego za pośrednictwem kamery optycznej.

Przykład sterowania pojazdem za pomocą opracowanego interfejsu został przedstawiony na rys. 9.

W zależności od struktury powierzchni danego przedmiotu można zaobserwować wiele kolorów będących pochodną koloru szukanego. Na rys. 9 przedstawiony jest przedmiot o powierzchni odbijającej światło, która w zależności od natężenia oświetlenia przyjmuje wiele kolorów pochodnych szukanemu. Działanie funkcji rozpoznającej kolor polega na dokonaniu selekcji kolorów w obszarze zarejestrowanym w czasie rzeczywistym przez kamerę optyczną. Kolor rozpoznany zostaje zabarwiony na czerwono w całym obszarze, z wyjątkiem fragmentu pola o wysokości 40 pikseli zliczanych od dołu tablicy, które podświetlane są na niebiesko.

Liczebność rozpoznanych pikseli regulowana jest potencjometrem widocznym na interfejsie, o nazwie *„ILOSC PIXELI W GRUP.”* w zakresie od 0 do 100.

Obszarowi o zadanej we wspomniany sposób liczebności przypisywany jest kursor, będący dodatkową bitmapą o kształcie szarego, przekreślonego na krzyż okręgu, którego zadaniem jest śledzenie grupy rozpoznanych pikseli.

Rys. 9. Rozpatrywany obiekt analizy koloru o pofalowanym kształcie [1]

Fig. 9. Examined object color analysis of undulating shape [1]

Pobranie próbki szukanego koloru z otoczenia odbywa się przez wciśnięcie lewego klawisza myszy na podglądzie z kamery. Wynik pobranego koloru uwidoczni się w postaci zabarwionego pierwszego paska obsługi o nazwie „SZUKANY KOLOR” (rys. 10).

W interfejsie wykorzystano przestrzeń barw o nazwie *HSL*, gdzie *H* (ang. *Hue*) oznacza częstotliwość światła, wyrażoną kątem na kole barw, która przyjmuje wartości od 0° do 360°. Model jest rozpatrywany jako stożek, którego podstawą jest koło barw (rys. 11).

Wymiary stożka opisuje składowa *S* – nasycenie koloru (ang. *Saturation*) jako promień podstawy. Nasycenie mierzone jest w skali 0–100 %. Wartość 0 % tego atrybutu ustawia odcień szarości dla koloru, natomiast wartość 100 % oznacza maksymalne nasycenie barwą.

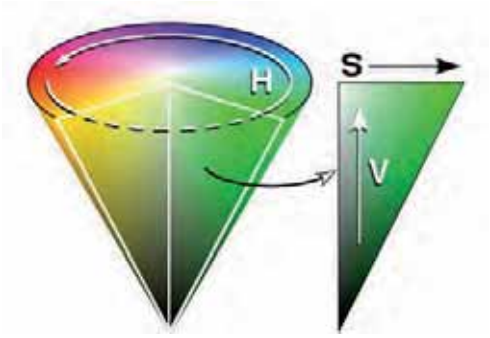
Rys. 10. Rozkład koloru na paskach edycji suwaków liniowych HSL (fragment obsługi interfejsu głównego) [1]

Fig. 10. The distribution of the bands linear editing HSL sliders (section handling the main interface) [1]

Jasność *L* – oznacza moc światła białego (ang. *Lightness*) jako wysokość stożka. Jasność mierzona jest w skali 0–100 %. Wartość 0 % tego atrybutu ustawia kolor czarny, a wartość 100 % ustawia kolor biały. Składowa *V* – (ang. *Value)* oznacza wartość koloru w danym punkcie palety koloru, która zmienia się w skali 0–100%, podobnie jak moc światła L.

W grafice rastrowej (ang. *raster graphics*) obraz jest przedstawiany jako dwuwymiarowa tablica danych. Każdy jego mały fragment jest opisany przez określoną ilość informacji, czyli bitów.

Kontekst urządzenia (ang. *device context*) jest strukturą przechowującą informacje na temat urządzenia graficznego w danym systemie operacyjnym, w którym pracuje jednostka centralna komputera. Na rzecz kontekstu można



Rys. 11. Reprezentacja przestrzeni barw *HSV (HSL)* (źródło: Wolna Encyklopedia)

Fig. 11. Representation of the HSV color space (HSL) (Source: The Free Encyclopedia)

wywoływać funkcje, tworząc w ten sposób obrazy wyświetlane na tym urządzeniu. Chcąc wykonać określona operację w oknie, musimy pobrać kontekst odnoszący się do niego. Uchwyt kontekstu okna możemy pobrać posługując się funkcją `GetDC()`:

```
HDC hdcObszarKlienta = GetDC(hWnd);
```

Po zakończeniu pracy z kontekstem należy go zwolnić. W systemie Windows tworzy się go tymczasowo i dlatego należy go zwolnić, kiedy już go nie potrzebujemy. Robimy to przez wywołanie funkcji `ReleaseDC()`:

```
ReleaseDC (hWnd, hdcObszarKlienta);
```

Funkcja `GetDC()` pozwala nam zajmować się jedynie obszarem klienta okna. Pozostała jego część, czyli obszar pozakliencki jest wtedy poza naszym zasięgiem.

Jeśli jednak chcemy zająć się także i tym rejonem, potrzebujemy kontekstu dla całego okna. Pozyskujemy go funkcją `GetWindowDC()`:

```
HDC hdcOkno = GetWindowDC(hWnd);
```

Otrzymany tą drogą kontekst pokrywa obszar nie tylko wnętrza okna `hWnd`, ale też jego paska tytułu. Posługiwanie się nim należy więc do sytuacji specjalnych, gdyż te części okna są ważne dla systemu Windows.

Po zakończeniu pracy z kontekstem należy go zwolnić za pomocą funkcji `ReleaseDC()`:

```
ReleaseDC (hWnd, hdcOkno);
```

Do stworzenia obiektu bitmapy w implementacji języka Visual C++ można wykorzystać funkcje:

– `CreateBitmap()`, tworzącą bitmapę o podanych wymiarach, głębi kolorów i zawartości,

– `CreateCompatibleBitmap()`, tworzącą bitmapę kompatybilną z danym kontekstem urządzenia i mającą podane wymiary,
– `LoadImage()`, wczytać bitmapę z pliku dyskowego.

Za wyświetlanie bitmap odpowiadają funkcje:
– `BitBlt()`, która dokonuje przekopiowania pikseli z jednego kontekstu urządzenia do drugiego,
– `TransparentBlt()`, dokonująca kopiowania z możliwością wybrania koloru,

Funkcją, która umożliwia wczytywanie obrazków z plików jest `LoadImage()`, wywoływana jako:

```
HANDLE LoadImage(HINSTANCE hInstance,
LPCTSTR lpszName, UINT uType, int cx-
Desired, int cyDesired,UINT fuLoad);
```

Przy jej pomocy odczytujemy obrazek rastrowy zapisany w pliku, a następnie używamy go w operacjach graficznych. Funkcja `LoadImage()` obsługuje formaty bitmapowe, tzw. BMP.

Przygotowanie bitmapy, programowo wygląda następująco:

1. Wczytanie bitmapy
   ```
   HBITMAP hbmpBitmapa = (HBITMAP)
   LoadImage(NULL,"bitmapa.bmp", 0, 0,LR_
   LOADFROMFILE);
   ```
2. Stworzenie pamięciowego kontekstu urządzenia
   ```
   HDC hdcPamiec = CreateCompableDC(NULL);
   ```
3. Wybranie wczytanej bitmapy w kontekście pamięciowym
   ```
   HBITMAP hbmpStara = (HBITMAP)
   SetObject(hdcPamiec, hbmpBitmapa);
   ```
4. Dokonanie transferu pikseli, czyli wyświetlenie bitmapy w punkcie (nX, nY) kontekstu hdcKontekst
   ```
   BitBlt (hdcKontekst, nX, nY, Bimapa.
   bmWidth,Bitmapa.bmHeight, hdcPamiec, 0,
   0, SRCCOPY);
   ```
5. Zwolnienie obiektu bitmapy
   ```
   DeleteObject (hbmpBitmapa);
   ```

3. Uwagi końcowe

Sterowanie robotem na bazie rozpoznanej barwy obiektu stanowi jedną z wielu możliwości obsługi urządzeń technicznych. Zaprezentowana aplikacja rozpoznawania koloru, stanowi konstrukcję uogólnioną, której rozwiązania można przenieść na wybrane urządzenie techniczne.

Uogólnienie, jakim jest samochód-zabawka sterowany zdalnie, stanowi jedynie przykład sterowania robotem na bazie koloru trajektorii. Stosując ten sam algorytm analizy koloru można zaproponować sterowanie wielu innych urządzeń, które pod względem konstrukcyjnym mogą wykonać znacznie więcej złożonych czynności.

Przykładem takiego urządzenia może być spawarka bezprzewodowa jako urządzenie poruszające się samodzielnie po wykrytym ściegu, tworzonym przez dwa arkusze blach objętych procesem spawania.

Na efektywność pracy programu rozpoznającego kolory wpływają ograniczenia urządzeń wizyjnych, które dostarczają obraz w postaci bitmapy do dalszej obróbki.

Możliwych zastosowań układów sterowanych na podstawie rozpoznanego koloru obiektu jest bardzo wiele, z których każde pozwala interpretować dane obliczeniowe w wizualnej szacie graficznej, dzięki czemu można stwierdzić, że cel pracy został osiągnięty.

Bibliografia

1. Makarewicz A., *Sterowanie robotem na bazie rozpoznanej barwy obiektu*, praca magisterska Politechnika Białostocka 2008.
2. Tadeusiewicz R., Korohoda P., *Komputerowa analiza i przetwarzanie obrazów*, wydawnictwo Fundacji Postępu Telekomunikacji, Kraków 1997
3. Davis Chapman, *Visual C++ dla każdego*, wyd. Helion, Gliwice 1999
4. Borkowski P., *Tworzenie aplikacji dla Windows. Od prostych programów do gier kompurowych*, Helion, Gliwice 2008
5. Strony internetowe:
   Windows GDI:
   http://xion.org.pl/productions/texts/coding/megatutorial
   Microsoft – oprogramowanie sprzętowe:
   http://msdn.microsoft.com.

Object Colour Based Robot Control

Abstract: The goal of the work is the analysis of the robot control using the developed algorithm for processing the still photo image, implemented as a C++ application, making control of the external device connected to the desktop computer possible.

Keywords: robot control, colour recognition, image processing, mathematical image recording

Artykuł recenzowany, nadesłany 21.11.2013 r., przyjęty do druku 20.12.2013 r.

mgr inż. Adam Makarewicz

Otrzymał tytuł magistra Automatyki i Robotyki na Wydziale Mechanicznym (2008) oraz tytuł magistra Elektroniki i Telekomunikacji na Wydziale Elektrycznym (2011) Politechniki Białostockiej. Obecnie jest studentem drugiego roku Studiów Doktoranckich na Wydziale Elektrotechniki Politechniki Białostockiej.

e-mail: adammakarewicz@interia.pl