ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2008
Seria: AUTOMATYKA z. 150 Nr kol. 1796

Aleksander NAWRAT, Marcin JASIŃSKI
Politechnika Śląska

SZYBKIE PRZETWARZANIE OBRAZU W ROBOTYCE

Streszczenie. Komputerowa analiza obrazów jest zadaniem bardzo trudnym i czasochłonnym. Ma to szczególne znaczenie w przypadku analizy obrazu w czasie rzeczywistym, gdy dane należy przetwarzać w tempie kilkunastu bądź nawet kilkudziesięciu klatek na sekundę. Wynik musi być w takim przypadku dostępny niemal natychmiast. Niniejszy referat przedstawia wyniki prac prowadzonych nad algorytmami pozwalającymi na wyizolowanie oraz śledzenie obiektu określonego koloru w obrazie. Otrzymane rezultaty pozwalają zastosować opracowane algorytmy do przetwarzania obrazu w czasie rzeczywistym.

FAST IMAGE PROCESSING IN ROBOTICS

Summary. Computer image processing is considered to be very hart and time consuming task. It is particularly meaningful while analyzing images in real time speed – when new data comes in a dozen or so frames per second. This paper show result of work about algorithms that allows to detect and watch shape of designed color in source image. Result that were gain are good enough to use this algorithm in the real time image processing.

1. Wstęp

W literaturze [1, 2, 3] odnaleźć można wiele przykładów opisujących algorytmy filtracji obrazów statycznych. Dają one bardzo dobre rezultaty, które zostają niestety okupione długim czasem potrzebnym na analizę oraz wykonanie skomplikowanych operacji matematycznych. Podejście takie nie zdaje egzaminu w przypadku przetwarzania obrazu w czasie rzeczywistym, gdzie ważniejszy od bezbłędnego przeprowadzenia filtracji jest czas, w którym sama operacja zostaje przeprowadzona. Prace prowadzone w tej dziedzinie nie są tak liczne jak w przypadku analizy obrazów statycznych, ich wyniki są jednak bardzo obiecujące [5, 6].

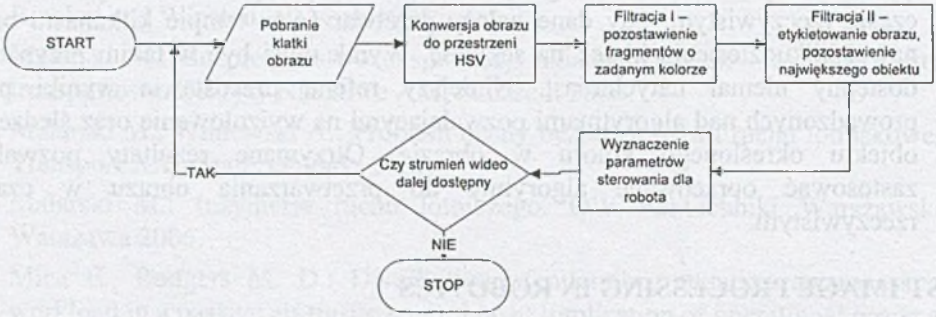
2. Założenia opracowanego algorytmu

Opracowana metoda filtracji obrazów pozwala na wyizolowanie w obrazie barwnym kształtu o określonym kolorze, odpowiadającym kolorowi przyjętego wzorca. Nadaje się ona doskonale do śledzenia położenia oraz rozmiaru kształtu

o konkretnym kolorze, wykorzystana może być zatem jako źródło danych dla algorytmów sterowania obiektami autonomicznymi. Jako że śledzony obiekt powinien zawsze stanowić jeden spójny kształt w obrazie, opracowana metoda pozwala na pozostawienie jedynie największego spośród wykrytych kształtów. Wszystkie inne, jako nieistotne dla celów sterowania, zostają zignorowane.

Algorytm (*Rys. 1.*) opracowany w wyniku przeprowadzonych badań składa się z dwóch głównych części:

1. Filtracja obrazu barwnego w celu pozostawienia jedynie tych jego obszarów, których kolor odpowiada kolorowi przyjętego wzorca;
2. Etykietowanie obiektów obecnych w obrazie w celu nadania im unikalnych identyfikatorów, sprawdzenie ich rozmiaru oraz pozostawienie w obrazie wynikowym jedynie największego spośród nich.

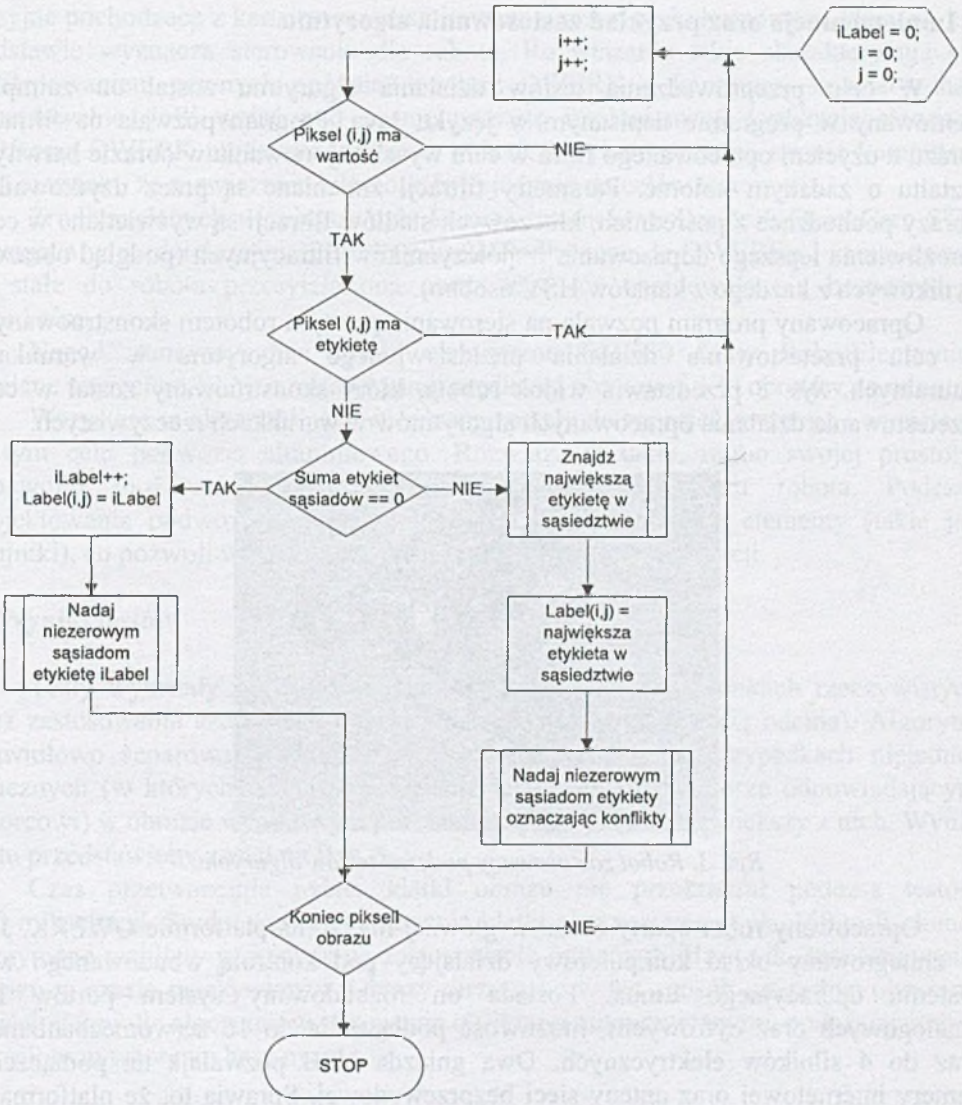


Rys. 1. Opracowany algorytm jako źródło danych decyzyjnych dla algorytmu sterowania platformą jezdną

W celu przeprowadzenia pierwszego etapu filtracji obraz poddany zostaje obróbce przez opracowany w ramach tego projektu filtr pasmowo-przepustowy. Barwa reprezentowana jest w tym przypadku w przestrzeni HSV. Obraz pochodzący z każdego z kanałów analizowany jest oddzielnie. Piksel jest pozostawiany w obrazie, jeśli jego wartość mieści się w określonym przedziale (wartości zbyt małe oraz zbyt duże są usuwane). Powstałe w ten sposób obrazy są progowane w celu uzyskania ich reprezentacji binarnej, a obraz wynikowy powstaje przez wymnożenie odpowiadających sobie pikseli z kanału H przez piksele kanału S.

W celu zaetykietowania obrazu zastosowany został zmodyfikowany algorytm „blob detection". W przypadku wykrycia piksela o zadanej wartości (w tym przypadku operacje odbywają się na obrazie binarnym – poszukiwaną wartością jest więc „1") należy sprawdzić, czy piksel ma już przyporządkowaną etykietę. Jeśli nie ma, algorytm sprawdza, czy któryś z sąsiadów aktualnie analizowanego piksela ma etykietę. Jeśli tak, piksel otrzymuje etykietę identyczną, jeśli natomiast nie, oznacza to wykrycie nowego kształtu – nadana zostaje mu wówczas zupełnie nowa etykieta, nieobecna jeszcze w obrazie.

Główne operacje wykonywane podczas filtracji obrazu przedstawione zostały na schemacie zilustrowanym na *Rys. 2.*



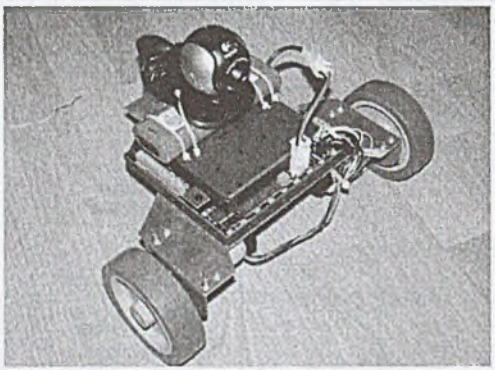
Rys. 2. Główna część opracowanego algorytmu pozwalającego na wyselecjonowanie w obiekcie kształtu o określonej barwie

Filtracja przeprowadzana jest w przestrzeni HSV. Zastosowanie takiego właśnie sposobu reprezentacji kolorów w obrazie pozwala na pewną niezależność od warunków oświetleniowych. Dzieje się tak, ponieważ informacja o jasności (luminancji) danego piksela zapisana jest w osobnym kanale i nie jest ona powiązana z kolorem (w przypadku reprezentacji obrazu przestrzeni RGB zarówno kolor, jak i luminancja zapisane są w tym samym kanale). Reprezentacja obrazu w przestrzeni HSV pozwala ponadto łatwo separować obraz ze względu na kolor (który zapisany jest w kanale H), jak i jego nasycenie (zapisane w kanale S). Operacje takie przeprowadzone w przestrzeni RGB są bardziej czasochłonne, wymagają bowiem analizy wszystkich 3 kanałów, w których zapisane są poszczególne składowe kolorów.

3. Implementacja oraz przykład zastosowania algorytmu

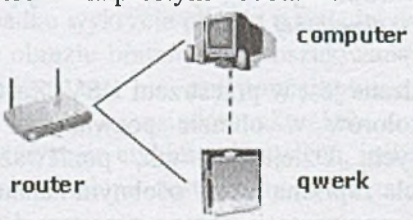
W celu przeprowadzenia testów działania algorytmu został on zaimplementowany w programie napisanym w języku Java. Program pozwala na filtrację obrazu z użyciem opracowanego filtra w celu wyselekcjonowania w obrazie barwnym kształtu o zadanym kolorze. Parametry filtracji zmieniane są przez użytkownika. Obrazy pochodzące z pośrednich, kluczowych stadiów filtracji są wyświetlane w celu umożliwienia lepszego dopasowania współczynników filtracyjnych (podgląd obrazów wynikowych z każdego z kanałów HSV osobno).

Opracowany program pozwala na sterowanie prostym robotem skonstruowanym w celu przetestowania działania przedstawionego algorytmu w warunkach naturalnych. *Rys. 3* przedstawia widok robota, który skonstruowany został w celu przetestowania działania opracowanych algorytmów w warunkach rzeczywistych.



Rys. 3. Robot zastosowany podczas testów algorytmu

Opracowany robot oparty został w głównej mierze na platformie QWERK. Jest to zintegrowany układ komputerowy działający pod kontrolą wbudowanego weń systemu operacyjnego Linux. Posiada on rozbudowany system portów I/O (analogowych oraz cyfrowych), możliwość podłączenia do 16 serwomechanizmów oraz do 4 silników elektrycznych. Dwa gniazda USB pozwalają na podłączenie kamery internetowej oraz anteny sieci bezprzewodowej. Sprawia to, że platforma ta nadaje się doskonale do sterowania prostymi robotami.



Rys.4. Przykładowa architektura sieci łączącej komputer sterujący z układem QWERK

Cechą charakterystyczną zastosowania opisanej platformy jest konieczność przeprowadzenia wszystkich obliczeń matematycznych związanych z przetwarzaniem obrazu na komputerze PC. Jak pokazano na Rys.4, układ QWERK działa pod kontrolą komputera PC, z którym połączenie nawiązywane jest przez sieć lokalną. Dane

wizyjne pochodzące z kamery są transmitowane przez sieć do komputera, który na ich podstawie wyznacza sterowanie dla robota. Rozwiązanie takie charakteryzuje się występowaniem pewnych opóźnień na linii QWERK – komputer, są one jednak akceptowalne, jeśli wziąć pod uwagę prostotę implementacji (aplikacja sterująca platformą QWERK może być tworzona w Javie i wykonywana po stronie komputera PC) oraz fakt, że rozwiązanie tworzone było jedynie do celów testowych.

Źródłem danych wizyjnych była kamera internetowa Logitech QuickCam STX pracująca w rozdzielczości 320x240 pikseli. Podłączona do QWERKa i zamocowana na stałe do robota przesyłała ona obraz tego, co znajdowało się bezpośrednio przed nim.

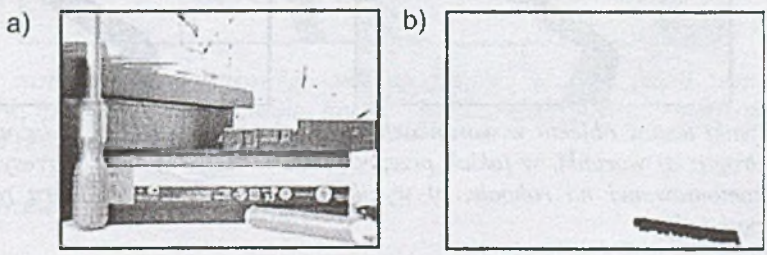
Napęd stanowiły dwa silniki elektryczne EMG30 firmy RobotElectronics zasilane napięciem 12 V, o maksymalnej prędkości obrotowej 170 obrotów na minutę.

Wszystkie te elementy przytwierdzone zostały do zaprojektowanego i wyciętego w tym celu podwozia aluminiowego. Rozwiązanie takie, mimo swojej prostoty, zapewniło możliwość szybkiego montażu oraz demontażu robota. Podczas projektowania podwozia przewidziano miejsce na dodatkowe elementy (takie jak czujniki), co pozwoli w przyszłości na łatwą rozbudowę konstrukcji.

4. Wyniki testów

Testy wykazały prawidłowe działanie algorytmu w warunkach rzeczywistych (bez zastosowania sztucznego tła, na którym obiekt wyraźnie się odcina). Algorytm prawidłowo separował kształty o określonym kolorze. W przypadkach niejednoznacznych (w których wykryto więcej niż jeden kształt o kolorze odpowiadającym wzorcowi) w obrazie wynikowym pozostawiany był jedynie największy z nich. Wynik testu przedstawiony został na Rys. 5.

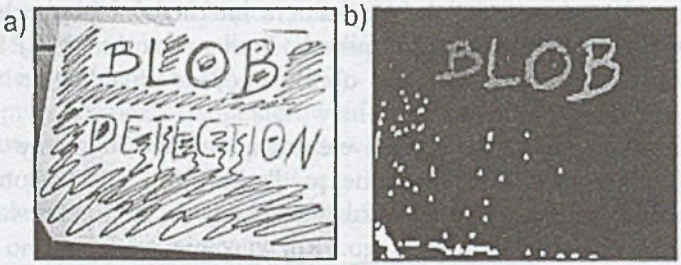
Czas przetwarzania jednej klatki obrazu nie przekraczał podczas testów 500 milisekund. Średni czas przetwarzania klatki obrazu wynosił ok. 150 milisekund. Otrzymane rezultaty pozwalają na zastosowanie opisanych filtrów do rozpoznawania obrazu w czasie rzeczywistym. Obrazy otrzymane w ten sposób mogą być obrazami wejściowymi dla algorytmów sterowania obiektami autonomicznymi, podążającymi za określonym wzorcem barwnym.



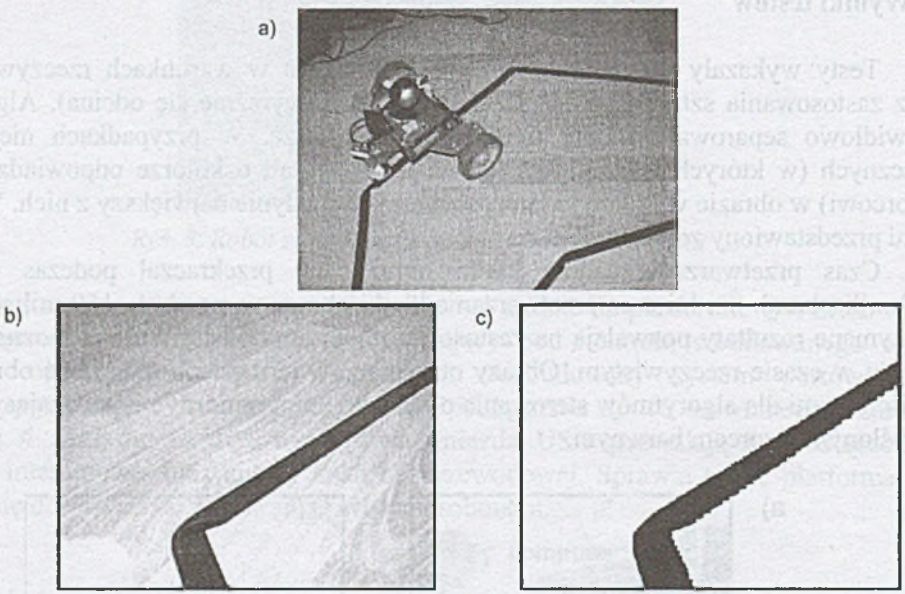
Rys. 5. Wyniki testów separowania kształtu o określonym kolorze w obrazach statycznych: a) widok z kamery, b) obraz wynikowy filtracji (pozostawiony został jedynie największy spośród wykrytych zielonych obiektów).

Jako że opracowany algorytm pozwala na zaetykietowanie wszystkich wykrytych w obrazie elementów, możliwe jest ich późniejsze rozróżnienie. Umożliwia

to zastosowanie algorytmu do takich zadań, jak komputerowe rozpoznawanie pisma. Jak pokazano na Rys. 6 czerwony napis został prawidłowo wykryty. Jako że każda litera oznaczona jest własną etykietą, możliwe jest zastosowanie algorytmów porównujących kształt wykrytych obiektów (liter) z wzorcem danej litery w celu odczytania całego wykrytego napisu.



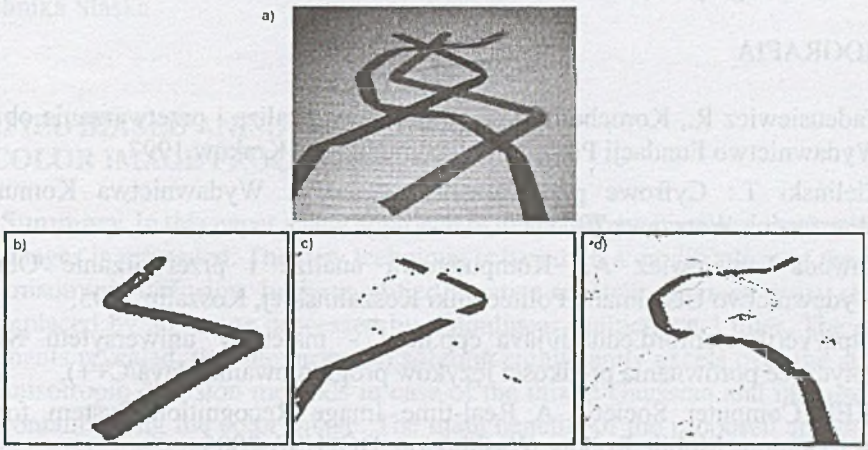
Rys. 6. Przykład zastosowania algorytmu - wykrywanie kolorowego tekstu: a) obraz oryginalny, b) obraz przetworzony (każdy obiekt w przefiltrowanym obrazie oznaczony jest osobną etykietą)



Rys. 7. Wyniki testów obiektu w warunkach rzeczywistych – przykład podążania za wykrytą drogą: a) warunki, w jakich przeprowadzano testy, b) obraz przesyłany z kamery zamontowanej na robocie, c) wynik filtracji – droga, za którą robot powinien podążać.

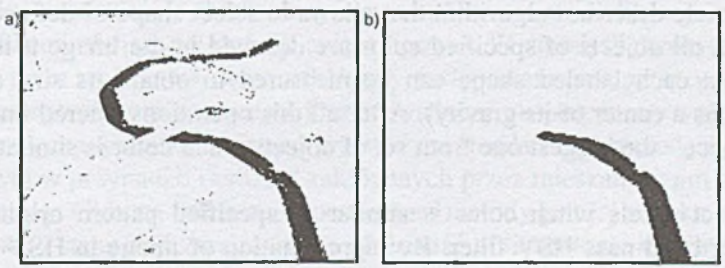
Przeprowadzone testy wykazały, że algorytm separuje kolory na tyle dobrze, że możliwe jest jego zastosowanie w celu sterowania obiektami autonomicznymi. Doświadczenie polegające na zaprogramowaniu robota w ten sposób, aby podążał on za wykrytą drogą, zakończyło się sukcesem. Przykładowe obrazy z przeprowadzonych testów zilustrowane zostały na Rys. 7.

Swobodny dostęp do parametrów filtracyjnych (które mogą być zmieniane podczas działania algorytmu) umożliwia wybór koloru drogi, po jakiej porusza się obiekt. Jak pokazano na Rys. 8, opracowany algorytm doskonale sprawdza się przy wyszukiwaniu drogi na skrzyżowaniach składających się z różnokolorowych linii.



Rys. 8. Sposób odnajdywania przez obiekt drogi na różnokolorowym skrzyżowaniu: a) sytuacja testowa, b) wynik filtracji mającej na celu odnalezienie czerwonej linii, c) wynik poszukiwania linii niebieskiej, d) wynik poszukiwania linii zielonej.

Zastosowanie dodatkowych, wchodzących w skład opracowanego algorytmu filtrów pozwoliłoby na wyeliminowanie z obrazu zbędnych szumów oraz wyszukanie najbliższego odcinka drogi będącego spójnym kształtem, jak pokazano na Rys. 9.



Rys. 9. Efekt zastosowanie dodatkowego zestawu filtrów: a) obraz przed zastosowaniem filtrów, b) obraz po zastosowaniu filtrów – pozostawiono jedynie najbliższy fragment drogi stanowiący spójną całość.

5. Podsumowanie

Opracowany algorytm doskonale nadaje się do przetwarzania obrazu w celu wyizolowania w nim obiektów o określonym kolorze. Przygotowane w ten sposób obrazy stanowią dobry punkt wyjściowy dla algorytmów pozwalających na sterowanie obiektami bezzałogowymi podążającymi za określonym wzorcem barwnym.

Implementacja algorytmu w Javie pociąga za sobą znaczne nakłady obliczeniowe oraz pamięciowe (co wynika bezpośrednio ze specyfiki języka). Mimo to w chwili

obecnej program jest w stanie przerabiać ok. 3 klatki na sekundę. Jak wynika z testów porównawczych prędkości języków 4, zastosowanie języka programowania niskiego poziomu (takiego jak C lub C++) pozwoliłoby przyspieszyć obliczenia pięciokrotnie.

Zastosowanie filtracji w przestrzeni HSV pozwala zachować pewną niewrażliwość algorytmu na zmienne warunki oświetleniowe.

BIBLIOGRAFIA

1. Tadeusiewicz R., Korochoda P.: Komputerowa analiza i przetwarzanie obrazów. Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków 1997.
2. Zieliński T.: Cyfrowe przetwarzanie sygnałów. Wydawnictwa Komunikacji i Łączności, Warszawa 2005.
3. Zawada-Tomkiewicz A.: Komputerowa analiza i przetwarzanie Obrazów. Wydawnictwo Uczelniane Politechniki Koszalińskiej, Koszalin 2005.
4. http://verify.stanford.edu/uli/java_cpp.html - materiały uniwersytetu Stanford dotyczące porównania prędkości języków programowania (Java/C++).
5. IEEE Computer Society: A Real-time Image Recognition System for Tiny Autonomous Mobile Robots. Washington, DC, USA 2005.
6. You J., Bhattaharya P., Hungenahally S.: Real-time object recognition: hierarchical image matching in aparallel virtual machine environment. Pattern Recognition, 1998.

Recenzent: Prof. zw. dr hab. inż. Marek Kurzyński

Abstract

This article describes algorithm that allows to select shape of defined color from image. When all objects of specified color are detected in the image it is possible to label it. Then each labeled shape can be measured to obtain its size and position (represented as a center of its gravity). After all this operations filtered image contains only one object – the biggest one from set of objects witch color is similar to the color of pattern.

To select pixels witch color is similar to specified pattern original image is filtered using band-pass HSV filter. By representation of image in HSV color model this solution is quite independent from light condition because color of pixel is represented in one channel and its value (luminance) in another. It is faster than processing image in RGB color model because there is only need to analyze one channel of image.