Artur FOGIEL, Jacek TKACZ
UNIWERSYTET ZIELONOGÓRSKI, WYDZIAŁ ELEKTROTECHNIKI, INFORMATYKI I TELEKOMUNIKACJI, INSTYTUT INFORMATYKI I ELEKTRONIKI,
ul. Podgórna 50, 65-246 Zielona Góra

Rozpoznawanie symboli języka migowego z wykorzystaniem algorytmów cyfrowego przetwarzania obrazów

Mgr inż. Artur FOGIEL

Absolwent Uniwersytetu Zielonogórskiego. Interesuje się zagadnieniami cyfrowego przetwarzania obrazu i algorytmami uczenia maszynowego. Pasjonat nowych technologii.

e-mail: fogiel.artur@gmail.com

Dr inż. Jacek TKACZ

Absolwent Uniwersytetu Zielonogórskiego. Od 2009 roku pracownik zatrudniony na stanowisku adiunkta w Zakładzie Inżynierii Komputerowej. Prowadzi badania nad symboliczną metodą dowodzenia twierdzeń i jej praktycznymi zastosowaniami w informatyce oraz elektronice. Interesuje się nowoczesnymi technologiami projektowania i wytwarzania aplikacji, w tym aplikacji mobilnych. W latach 1997-2005 zajmował się projektowaniem i rozwojem oprogramowania dla polskich bibliotek (PROLIB).

e-mail: J.Tkacz@iie.uz.zgora.pl

Streszczenie

W artykule został opisany proces projektowania i realizacji aplikacji, przeznaczonej do analizy, w czasie rzeczywistym, obrazów cyfrowych zawierających symbole języka migowego. Stworzone oprogramowanie zostało napisane w języku C++. Wykorzystuje ono bibliotekę OpenCV w połączeniu z sensorem Microsoft Kinect. Oprogramowanie przetwarza około 15 klatek obrazu w ciągu sekundy i rozpoznaje gesty ze skutecznością 71% . Opracowane oprogramowanie jest odporne w znacznym stopniu na wpływ warunków oświetleniowych.

Słowa kluczowe: Microsoft Kinect, OpenCV, OpenNI, język migowy.

Realtime sign language recognizing using digital video processing algorithms

Abstract

The paper describes the implementation process of application that recognizes sign language symbols which are presented by a user in real time. The first part of this paper presents software and libraries used by this program. It also contains description of a Microsoft Kinect sensor with explanation of its operation. The second part of this paper is dedicated to KinectSL application. It presents the structure of the application, its algorithm and a study that led to the described method of recognizing sign language gestures. Section 5 deals with main problems that appeared while working on the project such as detecting writs and fingers. Subsection 5.4 contains the detailed explanation of parameters that are passed to a KNN classifier and formulas necessary to compute them. The presented application is multi-platform and can be run on Windows, Linux and MacOS operating systems. This software is able to recognize gestures for two hand at once that allows it to support advanced gestures. The tests carried out showed that KinectSL processed 15 frames per second and its performance of gestures recognizing was above 71 percent. The described system is a good basis to create a program that will be able to fluent translate sign language. It can also be used in industry and entertainment to control processes with user gestures.

Keywords: Microsoft Kinect, OpenCV, OpenNI, sign language.

1. Wprowadzenie

Dialog w języku migowym polega całkowicie na wykorzystaniu narządu wzroku. Rozmówcy obserwują ruch dłoni, ich położenie, czy mimikę twarzy, a następnie interpretują ich znaczenie. Zadaniem prezentowanego rozwiązania jest odtworzenie tego procesu.

Problem rozpoznawania gestów języka migowego za pomocą komputera jest jednak wciąż nierozwiązany. Naukowcy, studenci i pasjonaci próbują opracować rozwiązanie uniwersalne, niezależne od warunków zewnętrznych, takich jak nasłonecznienie pomieszczenia, kolor ubrania, czy tła osoby obserwowanej za pomocą kamery [1, 2].

2. Cyfrowe przetwarzanie obrazów

Rosnąca jakość urządzeń wizyjnych, rozwój algorytmów oraz wzrastająca wydajność procesorów, sprawiły, że komputerowe przetwarzanie obrazów wykorzystywane jest coraz częściej w bardzo wielu dziedzinach nauki oraz przemysłu.

Prezentowany projekt korzysta z popularnych bibliotek i oprogramowania służącego do pracy z obrazami cyfrowymi, takich jak:

OpenCV (*Open Source Computer Vision Library*) jest zbiorem zawierającym ponad 500 funkcji przeznaczonych do przetwarzania obrazu w czasie rzeczywistym przygotowanym przez firmę Intel. Biblioteka ta jest darmowa i została udostępniona na licencji BSD (co pozwala na jej wykorzystywanie również w celach komercyjnych). Szczegółowy opis możliwości biblioteki wraz z przykładami dostarczają pozycje [3, 4, 5].

OpenNI jest biblioteką dostarczającą zbiór interfejsów programistycznych udostępnianych na licencji open source. Celem twórców biblioteki jest aby udostępniane interfejsy stały się standardem dla aplikacji wykorzystujących urządzenia opierające się na interakcji z człowiekiem.

OpenNI oferuje wsparcie dla:

- Rozpoznawania komend głosowych;
- Obsługi gestów dłoni;
- Śledzenia części ciała użytkownika;

Realizacja tych zadań przebiega za pomocą middleware, np. NITE, z którym komunikuje się OpenNI, udostępniając programistom jednolity interfejs[6].

2.1. Microsoft Kinect

Kinect jest sensorem ruchu wyprodukowanym przez firmę Microsoft i pełni on rolę kontrolera dla konsoli Xbox 360. Pozwala on na interakcję użytkownika z konsolą za pomocą gestów oraz komend głosowych [6].

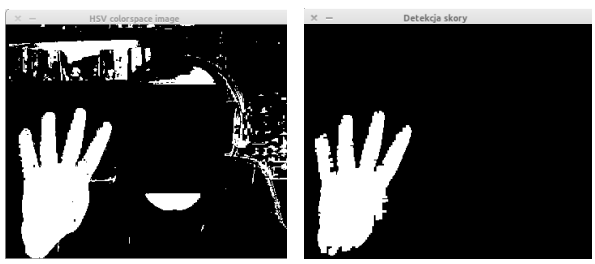
Kinect rejestruje obraz za pomocą dwóch rodzajów kamer. Standardowej kamery RGB o rozdzielczości 640x480, która wykorzystywana jest między innymi do wykrywania twarzy, czy analizy kolorów obserwowanego otoczenia. Druga, wyposażona w filtr podczerwony, pozwala uzyskać informacje o głębi obserwowanego otoczenia. Sensor rejestruje obraz na dystansie od 40 cm do 6,5 metra. Niestety jak widać na rysunku 7 obraz z tej kamery jest niestabilny i pomimo rejestrowania tego samego obiektu kolejne próbki różnią się pomiędzy sobą. Kinect może być wykorzystywany tylko wewnątrz pomieszczeń, gdyż ze względu na wykorzystanie metody światła strukturalnego jest on wrażliwy na duże nasłonecznienie.

Rys. 1. Chmura punktów generowana przez urządzenie Kinect pozwalająca określić położenie obiektu w przestrzeni
Fig. 1. Points cloud generated by the Kinect device used to define object position in space

3. Badanie algorytmów i technologii

Pierwszy etap prac nad projektem polegał na przetestowaniu i porównaniu, metod wykrywania dłoni za pomocą zwykłej kamery internetowej oraz sensora Microsoft Kinect.

Przeprowadzenie testów wymagało przygotowania dwóch rodzajów implementacji, pierwszej bazującej na wykorzystaniu kamery RGB (Logitech C310) oraz drugiej korzystającej z urządzenia Kinect.



Rys. 2. Dłoń wyodrębniona za pomocą filtrów nałożonych na obraz z kamery internetowej
Fig. 2. Hand extracted from the webcam image with help of various image filters

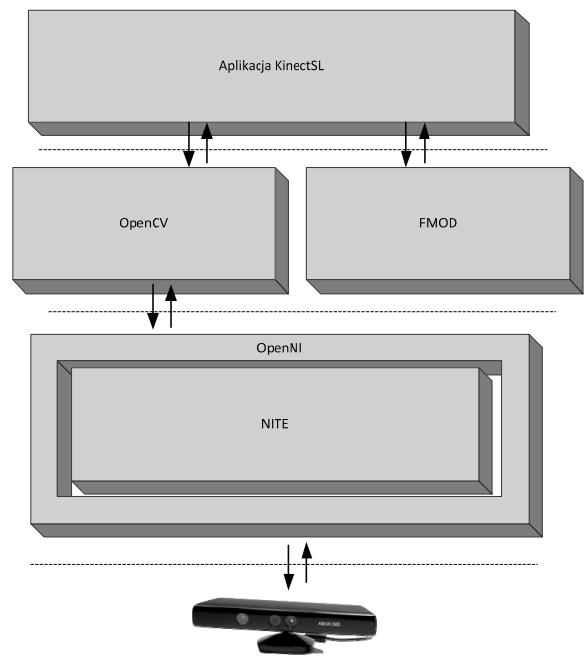
Badania przy użyciu kamery internetowej pokazały, że proces wyodrębniania dłoni na podstawie tonów skórnych [7, 8] pozwala dobrze odwzorować jej kontur, jednak jest on bardzo wrażliwy na czynniki zewnętrzne i musi odbywać się w niezmiennych warunkach. Zmiana oświetlenia, czy obecność przedmiotów o barwie zbliżonej do skóry, powodowało błędy w pracy programu. Wyodrębniany kształt w momencie pojawienia się czynnika powodującego zakłócenia przybierał najróżniejsze formy. Proces rozpoznawania kształtów nie był możliwy np. w momencie pojawienia się innej osoby w obszarze obserwowanym przez kamerę, lub rozświetlenia pomarańczowej ściany promieniami słońca. Poziom skuteczności w momencie wystąpienia zakłóceń wahał się dramatycznie, a testy pokazały, że nie przekraczał wtedy 10%. Drugie podejście do problemu, wykorzystujące urządzenie Kinect, pozwoliło wyeliminować większość ograniczeń.

4. Aplikacja KinectSL

Zaprojektowana aplikacja umożliwia rozpoznawanie, w czasie rzeczywistym, gestów języka migowego, prezentowanych przez osobę znajdującą się przed urządzeniem Microsoft Kinect. Program pozwala na pracę w trudnych warunkach oświetleniowych, np. wieczorem lub nocą. Prezentowany system obsługuje 22 gesty reprezentujące litery oraz 9 odpowiadających cyfrom [9]. Ważną cechą programu jest to, że pozwala on na wykrywanie i analizę dwóch dłoni jednocześnie, dzięki czemu obsługuje skomplikowane symbole, a także nie wymaga od użytkownika trzymania dłoni w ściśle określonym obszarze. Pozwala to na swobodne wykonywanie gestów i umożliwia aplikacji dostosowanie się do obsługującej ją osoby. Zrealizowany projekt jest wieloplatformowy i może zostać uruchomiony na wielu systemach operacyjnych, na przykład: Windows, Linuks, MacOS.

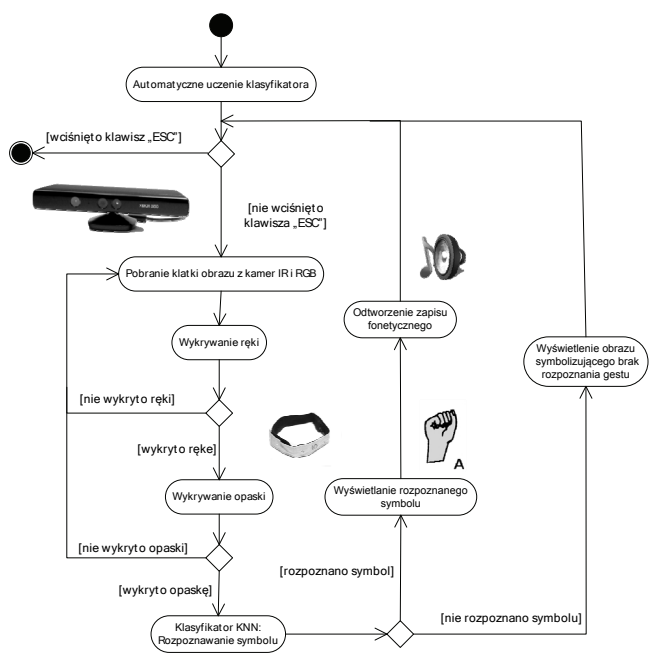
4.1. Struktura projektu

Na poniższym rysunku 3 widoczna jest struktura aplikacji oraz sposób komunikacji poszczególnych elementów programu pomiędzy sobą. Można zauważyć, że biblioteka OpenCV współpracuje z biblioteką OpenNI, która odpowiada głównie za komunikację z urządzeniem Kinect. System wykorzystuje dodatkowo bibliotekę FMOD do odtwarzania zapisu fonetycznego symboli języka migowego.



Rys. 3. Struktura aplikacji
Fig. 3. Application structure

4.2. Algorytm działania



Rys. 4. Diagram czynności aplikacji KinectSL
Fig. 4. Activity diagram for the KinectSL application

W momencie uruchomienia, pierwszą czynnością wykonywaną przez program jest uczenie klasyfikatora KNN, na podstawie plików zbioru uczącego, znajdujących się w katalogu programu. Po zakończeniu tego procesu, uruchamiany jest sensor Kinect. Jeżeli wszystko zostało wykonane poprawnie, następuje odtworzenie komunikatu głosowego o uruchomieniu aplikacji.

Kolejnym krokiem jest pobranie obrazu z kamer RGB oraz IR. Następnie obraz jest przetwarzany i poszukiwana jest opaska LED umieszczona na nadgarstku użytkownika. W momencie znalezienia opaski następuje rozpoznawanie gestów użytkownika. Poprawnie rozpoznany gest jest wyświetlany w postaci graficznej oraz odtwarzany jest jego zapis fonetyczny.

5. Implementacja

Proces opracowywania programu składał się z kilku kluczowych etapów, których zakończenie było niezbędne do umożliwienia realizacji skomplikowanego procesu rozpoznawania gestów. Warto zaznaczyć, że wszystkie opisane procesy zostały zaimplementowane samodzielnie przez autorów aplikacji, przy użyciu funkcji udostępnianych przez bibliotekę OpenCV, oraz technik zaproponowanych w literaturze [3, 4, 10, 11]. Aplikacja nie wykorzystuje algorytmów dostarczanych przez bibliotekę OpenNI ponieważ ich działanie wymaga, aby osoba znajdowała się w dużej odległości od urządzenia, co nie jest możliwe w przypadku prezentowanej aplikacji. Ograniczenia te wynikają z ze specyfiki zastosowanych sensorów w urządzeniu Kinect, które z założenia miały znacząco inne planowane zastosowanie.

5.1. Wyodrębnianie obszaru ręki

Kamera IR zwraca obraz w skali odcieni szarości, w którym piksele obiektu znajdującego się blisko urządzenia mają ciemniejszy odcień. Obiekty znajdujące się poza zasięgiem kamery mają odcień 255 (kolor biały). Tym samym kolorem oznaczane są zakłócenia.

Na podstawie obrazu pobranego z kamery IR, generowany jest histogram odległości pikseli od urządzenia Kinect. Wygenerowany histogram poddawany jest analizie, w trakcie której poszukiwana jest wartość >= 30, dla której liczba pikseli jest większa od 1. Wartość spełniająca te kryteria symbolizuje obiekt znajdujący się najbliżej kamery (będący w założeniu, dłonią użytkownika). Następnie, wykryty obszar zostaje wyodrębniony z całości obrazu i umieszczony w osobnym obiekcie przechowującym dane obrazu. Jest to możliwe dzięki użyciu operacji progowania binarnego na podstawie wartości uzyskanej z histogramu.

5.2. Wykrywanie nadgarstka

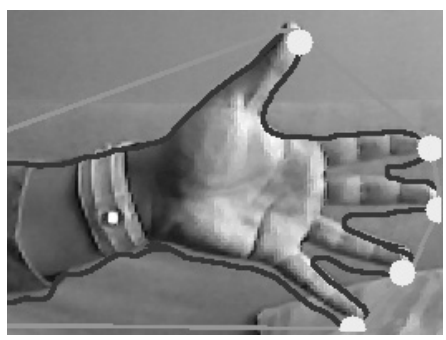
Wyodrębnienie obszaru ręki, pozwoliło rozpocząć pracę na sposobem wyznaczania obszaru należącego do dłoni. Określenie granic obszaru dłoni wymaga wcześniejszego poznania pozycji nadgarstka. Początkowe rozwiązanie polegało na odnalezieniu krzywizny odpowiadającej nadgarstkowi, jednak ze względu na zakłócenia generowane przez urządzenie, nie udało się uzyskać zadowalających efektów. Punkt reprezentujący nadgarstek przesuwał się, nawet kilka centymetrów od punktu właściwego, co powodowało błędy podczas pobierania próbek do zbioru uczącego klasyfikator oraz obniżało znacznie możliwości rozpoznawania gestów.

Problem udało się rozwiązać wykorzystując odblaskowe opaski na nadgarstki, wyposażone w diody LED, to rozwiązanie okazało się stabilne i działające (dzięki diodom), nawet przy bardzo słabym oświetleniu. Konieczne okazało się przygotowanie dodatkowych funkcji analizujących obraz za pomocą kamery RGB w celu odnalezienia jaskrawej opaski.

Rys. 5. Opaska wyodrębniona na podstawie koloru
Fig. 5. Strap extracted using color value

5.3. Wykrywanie palców

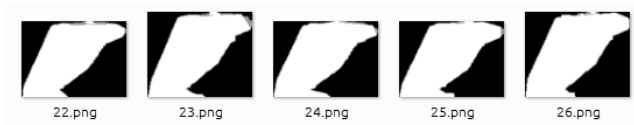
Poprawne wykrycie opaski umożliwiło przejście do kolejnego etapu jakim jest wyodrębnienie obszaru dłoni z konturu reprezentującego rękę użytkownika. Proces rozpoczyna się od wykrycia za pomocą algorytmu *k-curvature* krzywizn w konturze ręki, których kąty mieszczą się w określonym zakresie odpowiadającym czubkom palców. Następnie, aby usunąć błędnie rozpoznane krzywizny dla badanego konturu oblicza się punkty tzw. Hull points, które wyznaczają jego zewnętrzny obrys. Miejsce przecięcia obrysu ze środkiem krzywizny jest poprawnym czubkiem palca (rys. 6).



Rys. 6. Poprawnie wykryte czubki palców
Fig. 6. Correctly detected fingertips

5.4. Rozpoznawanie gestów

Ostatnim etapem prac nad programem było opracowanie kodu odpowiedzialnego za rozpoznawanie gestów. Obszar dłoni wyodrębniony w poprzednich krokach poddawany jest analizie, w wyniku której zwracany jest wektor zawierający jego właściwości [12, 13, 14, 15]. Generowanie wektora jest procesem dwuetapowym, najpierw obliczane są właściwości dla kształtu reprezentującego całą dłoń użytkownika.



Rys. 7. Kształt dłoni pokazującej literę „E”
Fig. 7. Hand shape that represents letter „E”

Następnie obliczane są parametry wypukłych obszarów dłoni.

Rys. 8. Wypukłe obszary dłoni pokazującej literę „E”
Fig. 8. Convex area of hand showing letter „E”

Zwrócony wektor przekazywany jest na wejście klasyfikatora KNN. Kompletny zestaw parametrów przekazywanych do klasyfikatora przedstawiony jest w tabeli 1.

Tab. 1. Parametry przekazywane do klasyfikatora
Tab. 1. Parameters passed to the classifier

Analiza konturu dłoni
Liczba wykrytych palców
6 pierwszych Hu momentów
Mimośród (ang. Eccentricity)
J1 / J2 (półoś wielka i półoś mała)
Nierówność izoperymetryczna (ang. Compactness)
Długości 12 odcinków, nachylonych pod różnymi kątami, od krawędzi do środka konturu
Analiza wypukłego obszaru
4 pierwsze Hu momenty
J1 / J2 (półoś wielka i półoś mała)
Mimośród (ang. Eccentricity)
Długości 12 odcinków, nachylonych pod różnymi kątami, od krawędzi do środka konturu

W przypadku projektowanego systemu, wykorzystanie tylko Hu momentów do rozpoznawania konturów gestów okazało się niewystarczające. Na postawie momentów, można jednak obliczyć dodatkowe parametry opisujące figurę, między innymi półoś wielką (1), półoś małą (2), czy mimośród (3). Szczegółowe informacje na temat wzorów oraz ich poszczególnych parametrów można znaleźć w opracowaniu [12].

$$J_1 = \frac{m_{00}}{2} \times \left(m20 + m02 + \sqrt{\left(m_{20} - m_{02}\right)^2 + 4m_{11}^2} \right), \quad (1)$$

$$J_2 = \frac{m_{00}}{2} \times \left(m20 + m02 - \sqrt{\left(m_{20} - m_{02}\right)^2 + 4m_{11}^2} \right), \quad (2)$$

$$\varepsilon = \frac{\left(\mu_{2,0} - \mu_{0,2}\right)^2 - 4\mu_{1,1}^2}{\left(\mu_{2,0} + \mu_{0,2}\right)^2}, \quad (3)$$

6. Wydajność i skuteczność aplikacji

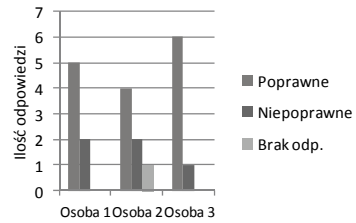
Głównym założeniem programu, było uzyskanie jak najwyższej skuteczności rozpoznawania gestów oraz jak największe skrócenie czasu przetwarzania obrazu.

Spełnienie warunku dotyczącego czasu jaki aplikacja potrzebuje na przetworzenie obrazu i analizę gestu udało się zweryfikować poprzez umieszczenie odpowiednich instrukcji w kodzie programu. Na podstawie danych zwróconych przez aplikacje można zauważyć, że program, przetwarza około 15 klatek obrazu w ciągu sekundy. Do testu wykorzystano laptop Acer Aspire 6930g z procesorem Intel Core 2 Duo T5800 2 GHz oraz wyposażonym w 3 GB pamięci RAM.

Weryfikacja skuteczności rozpoznawania gestów, wymagała przeprowadzenia badania, które wykonano na grupie trzech osób. Zadaniem grupy było pokazywanie wybranych gestów, następnie dokonano zapisu ilości gestów rozpoznanych poprawnie, niepoprawnie lub nierozpoznanych w ogóle.

Badanie zostało powtórzone trzykrotnie, na tej samej grupie osób w celu uzyskania dokładnych danych o pracy programu. Uśrednione wyniki przedstawiono na rysunku 9.

Na podstawie przeprowadzonego badania można stwierdzić, że skuteczność rozpoznawania gestów przez zaprojektowany system przekracza 70 %. W trakcie wykonywania testu zauważono, że na końcowy wynik największy wpływ mają dwa czynniki. Pierwszym są zakłócenia sensora Kinect wynikające z metody jego działania. Drugim, niedokładne odwzorowanie gestów przez użytkownika.



Rys. 9. Wykres przedstawiający skuteczność rozpoznawania gestów przez system
Fig. 9. Chart showing the efficiency of sign language recognition by the application

7. Podsumowanie i wnioski

Celem artykułu było zaprezentowanie możliwości biblioteki OpenCV, sensora Kinect oraz algorytmów uczenia maszynowego. W pierwszej części dokumentu zostały opisane biblioteki służące do cyfrowego przetwarzania obrazu, a także budowa i zasada działania urządzenia Microsoft Kinect. W drugiej części, na podstawie przedstawionych narzędzi i metod, została zaprojektowana aplikacja pozwalająca na rozpoznawanie gestów w czasie rzeczywistym. Opracowany system zapewnia akceptowalną szybkość przetwarzania obrazu oraz sprawność rozpoznawania gestów. Stanowi bardo dobrą bazę do opracowania aplikacji będącej kompletnym tłumaczem języka migowego, obsługującym także gesty wykonywane w trzech płaszczyznach. Sensor Kinect dla konsoli Xbox 360 został zaprojektowany z myślą o rozrywce. Na rynku pojawiły się jednak nowe urządzenia, dedykowane obsłudze gestów, takie jak „Kinect for Windows" lub Asus Xtion Pro Live, których zastosowanie prawdopodobnie pozwoliłoby usprawnić pracę opracowanej aplikacji.

8. Literatura

[1] Morteza Zahedi, Philippe Dreuw, David Rybach, Thomas Deselaers, Hermann Ney: Geometric Features for Improving Continuous Appearance-based Sign Language Recognition http://goo.gl/DpPKyh
[2] Zahedi M.: Robust Appearance-based Sign Language Recognition http://goo.gl/xN80nh
[3] Gary Bradski, Adrian Kaehler: Learning OpenCV Computer Vision with the OpenCV Library, O'Reilly 2008.
[4] Robert Laganière: OpenCV 2 Computer Vision Application Programming Cookbook, Packt Publishing, 2011.
[5] Rafajkowicz Ewaryst, Rafajłowicz Wojciech, Rusiecki Andrzej: Algorytmy przetwarzania obrazów i wstęp do pracy z biblioteką OpenCV; Oficyna Wydawnicza Politechniki Wrocławskiej, 2009.
[6] Greg Borenstein, Making Things See: 3D vision with Kinect, Processing, Arduino, and MakerBot, Maker Media, Inc; 2012.
[7] Oliveira V. A., Conci A.: Skin Detection using HSV color space, http://goo.gl/acNcE5
[8] Michael Van den Bergh, Luc Van Gool: Combining RGB and ToF Cameras for Real-time 3D Hand Gesture Interaction http://goo.gl/NNNo6S
[9] Olgierd Kosiba, Piotr Grenda: Leksykon języka migowego, Silentium 2011.
[10] Richard Szeliski: Computer Vision, Springer 2011.
[11] Ryszard Tadeusiewicz, Przemysław Korohoda: Komputerowa analiza i przetwarzanie obrazów; Społeczeństwo Globalnej Informacji, Kraków 1997.
[12] Johannes Kilian: Simple Image Analysis By Moments, http://goo.gl/DUiiH1
[13] Asbjørn Berge: Describing "shape" (feature extraction); http://goo.gl/KqObhq
[14] Marcin Kiełczewski: Reprezentacja i analiza obszarów http://goo.gl/PtT8tO
[15] Chris Kiefer, Nick Collins, Geraldine Fitzpatrick, Phalanger: Controlling Music Software With Hand Movement Using A Computer Vision and Machine Learning Approach http://goo.gl/pUYfVK

otrzymano / received: 21.08.2013
przyjęto do druku / accepted: 01.10.2013 *artykuł recenzowany / revised paper*