AKADEMIA GÓRNICZO-HUTNICZA
im. St. Staszica w Krakowie
WEAIiE, Katedra Automatyki
Laboratorium Biocybernetyki

Przedmiot: Interfejsy Multimodalne

MM133 KAMERA-MYSZKA

Temat projektu: Program poruszający myszką gestami rejestrowanymi przez kamerę internetową

Spis treści:

Wykonali: Jan Bieroń, Mateusz Kotlarz, Sabina Rydzek
3 rok *Informatyki*
konsultant: *Mateusz Zuń, dr inż. Mirosław Jabłoński*

Kraków, maj 2014.

1. Abstrakt

Celem projektu jest stworzenie aplikacji, która pozwoli użytkownikowi na ruch myszą i korzystanie z paru skrótów klawiaturowych za pomocą gestów. Gesty rozpoznawane są na podstawie wyliczanych przez program współczynników, a następnie interpretowane.

Do tego celu wykorzystywana jest kamera internetowa Logitech Pro 9000 oraz biblioteka EmguCV służąca do przechwytywania i analizy obrazu. Aplikacja napisana jest w języku C#. Wykrywanie położenia obiektu i wyliczanie współczynników zostało wstępnie przetestowane w programie Matlab, dając zadowalające rezultaty. Dokładne wykrywanie obiektu na obrazie możliwe jest dzięki zamianie modelu RGB na model YCbCr. Wybrane kombinacje współczynników pozwalają na identyfikację wykonywanego gestu i późniejsze jego przełożenie na ruch myszy lub skrót klawiaturowy.

Twórcom projektu udało się osiągnać podstawowe funkcjonalności projektu. W planach jest stworzenie modułu „uczącego", który pozwala na definiowanie gestów przez użytkownika.

Słowa kluczowe: rozpoznawanie gestów, kamera internetowa, współczynniki kształtu, binaryzacja, operacje kontekstowe, EmguCV, C#, ruchy myszy, skróty klawiszowe,

2. Wstęp i badania literaturowe

Cel i założenia projektu

Aplikacja umożliwia użytkownikowi sterowanie myszą oraz używanie prostych skrótów klawiaturowych przy użyciu gestów. Obraz zbierany jest przy pomocy kamery internetowej, a następnie przetwarzany i analizowany z wykorzystaniem biblioteki EmguCV w stworzonej aplikacji. Gesty mogą być rozpoznane z dowolnego miejsca na obrazie. Było to możliwe do osiągnięcia poprzez wykorzystanie binaryzacji, wykrywania obiektów na obrazie oraz wyliczania i porównywania współczynników kształtu. Projekt ma wykrywać gesty dwóch przedmiotów lub dłoni równocześnie.

W literaturze znaleźć można już podobne rozwiązania. VisualMouse [2] pozwala na sterowanie myszą za pomocą ruchów głowy i czterech zdefiniowanych przez użytkownika gestów. Inne prace, jak na przykład Camera Mouse [1] są wyspecjalizowane i skierowane do osób niepełnosprawnych, które mogą komunikować się z komputerem tylko za pomocą ruchów swojego ciała. Używane są w nich nie tylko gesty rąk i głowy ale również czubka nosa, bądź palca. Jedne z najbardziej skomplikowanych narzędzi pozwalają na rozpoznawanie ruchów gałki ocznej lub śledzą położenie lasera na skórze. [3][4] Ten projekt ma definiować 10 gestów użytkownika, rozpoznając obie dłonie na ekranie, jest więc sporym rozszerzeniem programów podobnych Visual Mouse. Chociaż możliwe, że nie będzie miał takiej dokładności, jak wyspecjalizowane programy dla niepełnosprawnych, możliwość dostosowania progów binaryzacji do wybranego przedmiotu i tym samym dobór przedmiotu, za pomocą którego chce się sterować myszą, sprawiają, że program staje się użytecznym narzędziem.

Słownik pojęć

- Model przestrzeni barw – matematyczny model widma fal elektromagnetycznych przedstawiony w przestrzeni. W projekcie wyróżnia się model RGB (osobne kanały dla czerwieni, zieleni i koloru niebieskiego) oraz model YCbCr – kanał luminancji (Y) oraz dwa kanały chrominancji (Cb, Cr).
- Współczynniki kształtu – współczynniki charakteryzujące kształt cząstki. Główną zaletą jest niezmienniczość ze względu na transformacje, np. zmianę skali, obrót.
- Binaryzacja – operacja punktowa, która w przypisuje pikselowi jedną z dwóch wartości (0 lub 1) porównując ją do jednego lub dwóch progów.
- Dylatacja – operacja kontekstowa na obrazie binarnym, która przypisuje przetwarzanemu pikselowi wartość 1, jeżeli którykolwiek z sąsiadów ma wartość 1.

- Erozja – operacja kontekstwo na obrazie binarnym, która przypisuje przetwarzanemu pikselowi wartość 0, jeżeli którykolwiek z sąsiadów ma wartość 0.
- Filtr medianowy – operacja kontekstowa uśredniająca, która wylicza średnia wartość piksela na podstawie pikseli otaczających.
- Odległość Pitagorejska – liczona odległość pomiędzy punktami w przestrzeni (może być wielowymiarowa), jest to pierwiastek sumy kwadratów odległości kolejnych współrzędnych punktów od siebie).

Zarys ogólny proponowanego rozwiązania

Obraz z kamery przetwarzany jest za pomocą dostępnych bibliotek EmguCV. Następnie jest on binaryzowany według piksela wskazanego przez użytkownika. Program kalibrowany jest przez autorów, następnie progi binaryzacji dobierane są już automatycznie. Z obrazu brany pod uwagę jest największy zbinaryzowany obiekt i na podstawie wyznaczonego z niego bounding box, wyliczane są poszczególne współczynniki kształtu. Wybór współczynników branych pod uwagę omówiony jest w trzecim rozdziale sprawozdania. Na ich podstawie z kolei, możliwe jest rozpoznanie gestu.

Największą trudnością w aplikacji jest stworzenie algorytmu, który w wystarczająco dokładny sposób rozpoznawałby gesty użytkownika. Dobranie odpowiednich progów binaryzacji umożliwia „wyłuskanie" z przetwarzanej ramki obiektu, którym jest wykonywany gest (może to być ręka, rękawiczka, kartka itd.), nie jest jednak wystarczające do określenia typu gestu. Z powodu zakłóceń występujących na obrazie, współczynników kształtu nie można wyznaczyć na podstawie jednej ramki, takie podejście uniemożliwiałoby również rozpoznawanie gestów ciągłych.

W literaturze pojawiają się rozwiązania, które bazują na porównywaniu ze sobą kolejnych ramek przetwarzanego obrazu i na tej podstawie określaniu ciągłego gestu [1]. Tam też ruch myszy jest określany na podstawie współrzędnych przedmiotu znajdującego się na ekranie i jego przesunięcia. W tym projekcie podejście jest podobne – mysz przesuwa się zgodnie ze środkiem ciężkości lewej ręki. Rozpoznawanie gestów w niniejszym projekcie jest rozwiązane za pomocą wyliczania 5-wymiarowych odległości pitagorejskich. Pozwala to na dość dokładne ich rozpoznawanie. Zastosowanie szablonów i porównywania obrazów nie spełnia założenia projektu, że obiekt, za pomocą którego wykonywane są gesty może znajdować się w każdym miejscu obrazu. W przeciwieństwie do rozwiązania z [1], obraz nie jest centrowany na obiekcie, dlatego metoda *template* jest w opisywanym tutaj projekcie nieskuteczna.

W rozdziale 3 przedstawiona jest koncepcja rozwiązania i poszczególne etapy algorytmu. Omówione są również najważniejsze klasy. Ostatni rozdział zawiera podsumowanie projektu i propozycje jego dalszego rozwoju.

3. Koncepcja proponowanego rozwiązania

Przetwarzanie obrazu

Do przetwarzania obrazu wybrana została biblioteka *EmguCV*. Jest to opakowanie na bibliotekę *OpenCV*, które domyślnie nie wspiera języka programowania C#. Pozwala na zebranie obrazu z kamery i wyświetlenie jej w aplikacji, zdefiniowanie różnych modeli kolorów oraz inne operacje na obrazie.

Filtrację obrazu, znalezienie największego obiektu na obrazie i wyznaczanie współczynników możliwe jest dzięki bibliotekom *AForge*. W programie zostały użyte biblioteki *AForge.Imaging* do przetwarzania obrazu oraz *AForge.Math* do pomocy przy wyliczaniu współczynników.

Użytkownik ma możliwość doboru parametrów obrazu tak, by binaryzacja była jak najbardziej skuteczna. Branych jest pod uwagę aż 9 różnych współczynników ustawień kamery, co pozwala na bardzo elastyczne dopasowanie obrazu do panujących warunków i wstępne wyeliminowanie zakłóceń.

Dokumentacja dla obu bibliotek dostępna jest w [5] i [6].

Binaryzacja i wyznaczenie obiektu z obrazu

Obraz binaryzowany jest przy użyciu omówionych wcześniej bibliotek. Użytkownik ma możliwość zarówno dobrania progów ręcznie w wybranym przez siebie modelu koloru, bądź zbinaryzowanie obrazu według piksela wybranego z obrazu kamery. Progi wyliczane są według kanałów i odchylenia binaryzacji. Im większe odchylenie binaryzacji, tym większy jest zakres, w którym piksele po binaryzacji będą miały wartość 1. Dostępne operacje kontekstowe (dylatacja, erozja, filtr medianowy) pozwalają dodatkowo na wyeliminowanie zakłóceń powstałych po binaryzacji obrazu, tak, aby funkcja wykrywająca położenie obiektów na obrazie mogła bezbłędnie wskazać na przedmiot (przedmioty) za pomocą których wykonywane będą gesty. Projekt korzysta tutaj z *AForge.Imaging.BlobCounter*, która odpowiada za policzenie obiektów znajdujących się na obrazie i szeregowania ich według wielkości.

Wybrane współczynniki

Wskaźniki testowane były początkowo na pomarańczowej rękawiczce na czarnym tle bądź tle „rzeczywistym" - obecne meble, ściana, inne przedmioty, które miały sprawdzić zachowanie się algorytmu przy obecnych zakłóceniach. Kolor okazał się jednak problematyczny, ponieważ często jego odcień podobny był do skóry. Do kolejnych testów i kalibracji używana była już rękawiczka niebieska.

Testowane były współczynniki: Compactness, Rmin/Rmax, Blair-Bliss, Haralick, M7, Malinowskiej, Zmodyfikowany Malinowskiej, W8, Fereta, Orientation.

Korzystanie ze współczynnika Compactness pozwala stwierdzić, czy ręka ustawiona jest tak, że wszystkie palce przylegają do siebie – współczynnik wtedy rośnie.

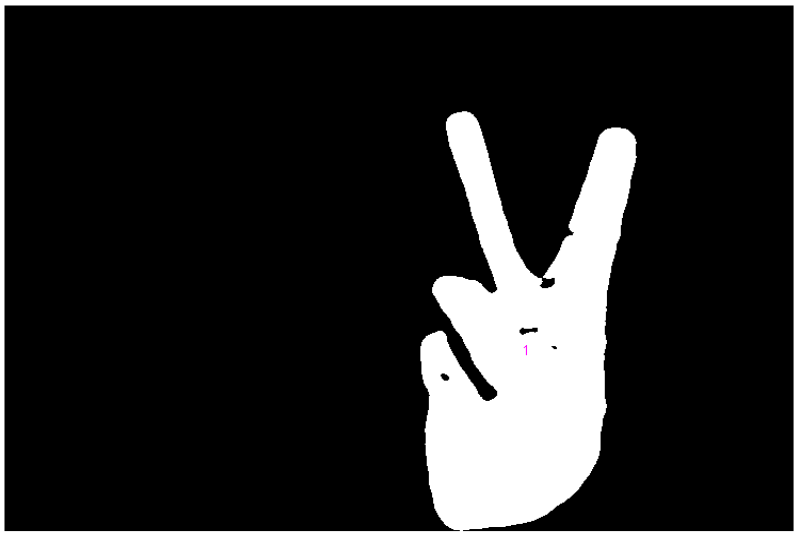
Współczynnik Blaira-Blissa pozwala określić cyrkularność obiektów, jest bowiem wyższy dla obiektów zbliżonych do koła lub elipsy. Rozpoznać można tym samym rękę zwiniętą w pieść oraz rękę wyprostowaną ze złożonymi palcami. Współczynniki Malinowskiej i zmodyfikowany Malinowskiej również dobrze oddają zbliżenie kształtu figury do koła.

Współczynnik W8 oddaje wydłużenie obiektu, jego wartości są łatwo zakłócane przez nieregularność badanych obiektów. Można go wykorzystać w połączeniu z innymi współczynnikami.

Współczynnik Fereta bardzo dobrze oddaje położenie obiektu w pionie i poziomie, dlatego w połączeniu z Orientation (odchylenie ręki od poziomej osi x), daje bardzo dobre wyniki.

Omówione powyżej współczynniki okazały się na tyle dokładne i znaczące, że można było zastosować je w projekcie. Pozostałe współczynniki ze względu na zniekształcone wyniki (Rmin/Rmax, M7) oraz mały zakres zmienności (Haralick) zostały odrzucone.

Przykładowe wyliczenia współczynników w programie MATLAB zgromadzone zostały w tabeli w Załączniku [1]. Matematyczne wzory współczynników można znaleźć w Załączniku [2].



Grafika 1: Przykładowe rozpoznanie gestu

Wyliczanie współczynników

W wyliczaniu współczynników bardzo pomocna była biblioteki *AForge*. Wbudowana funkcja *Fullness* działająca na znalezionym największym obiekcie jest odpowiednikiem współczynnika Compactness. Obwód obiektu do współczynnika Malinowskiej może być liczony na kilka sposobów, m.in. jako obwód prostokąta otaczającego lub ilość pikseli krawędziowych. W aplikacji zastosowana drugie podejście ze względu na nieregularność wykrywanych obiektów. Wyliczenie współczynnika Blaira i Fereta również było możliwe przy użyciu wbudowanych funkcji znajdujących środek ciężkości obiektu.

Ruch myszy i rozpoznawanie gestów

Mysz śledzi położenie środka ciężkości lewej ręki. Z jej ruchem związane są dwa parametry: *Smoothnes* i *Sensitivity*. Pierwszy z nich przesuwa mysz o odległość między punktami podzieloną przez ustawioną wartość (im większa, tym ruch myszy jest „gładszy", ale też wolniejszy). Drugi parametr zmniejsza lub zwiększa okno zbierania środka ciężkości. Gesty wykonywane mogą być obiema rękami. Odległość pitagorejska w 5 wymiarach (5 współczynników) między obserwowanymi gestami, a gestami wzorcowymi pozwala na rozpoznanie najbardziej prawdopodobnego gestu. Rozpoznawanie gestów osobno dla lewej i prawej ręki rozszerza możliwe kombinacje i tym samym listę obsługiwanych akcji i skrótów.

Kalibracja

Usprawnienie rozpoznawania gestów wymaga wielu testów i uśrednienia uzyskanych wyników, by móc lepiej dopasować gesty wzorcowe. Sposobem na „przeskakujące" czasem gesty jest zastosowanie licznika, który sprawdza, czy gest nie zmienił się w czasie 30 klatek obrazu i dopiero wtedy uruchamia akcję.

slayer	0,4935	0,8017	0,3969	0,7159	0,5280
fist	0,6877	0,9278	0,0867	0,9202	0,7363
victory	0,6081	0,7955	0,3705	0,7296	0,4295
vopen	0,6826	0,8775	0,2246	0,8166	0,5078
hopen	0,7478	0,8615	0,2237	0,8172	2,1469
fingers	0,4928	0,8738	0,3325	0,7504	0,6084
scissors	0,5700	0,7641	0,3956	0,7166	1,9796
shaka	0,4623	0,8730	0,3519	0,7397	0,8680
thumbup	0,5133	0,8878	0,2926	0,7737	0,8680
thumbleft	0,5095	0,9006	0,3259	0,7542	0,9293

Tabela 1: Współczynniki dla 10 rozpoznawanych gestów

Wprowadzona została również tolerancja, która zapobiega dużym błędom rozpoznawania gestów, a jednocześnie pozwala na pewne odchylenie współczynników.

Skróty klawiszowe

W programie zaimplementowane są dodatkowe, globalne skróty klawiszowe. Jest to Ctrl+Alt+Shift+Z, który włącza/wyłącza sterowanie myszy gestami oraz Ctrl+Alt+Shift+X, który odpowiada za minimalizację i pojawianie się okna.

Uczenie gestów (opcjonalnie)

W aplikacji użytkownik posiada możliwość zdefiniowania własnych gestów, przypisania ich do konkretnych przycisków myszy lub skrótów klawiaturowych i zapisania ich do pliku. Gesty są zapamiętywane i nie usuwają się przy restarcie programu. Zapisywane są jako nazwa, która podaje użytkownik i wyliczone przez program współczynniki, które pozwalają na rozpoznanie gestu. Wadą jest możliwość zdefiniowania przez użytkownika gestów, które nakładają się na siebie i tym samym zakłócają pracę programu. W takim wypadku można wrócić do ustawień początkowych lub wczytać inny zestaw gestów z pliku.

4. Rezultaty i wnioski

Zastosowane w projekcie rozwiązania pozwoliły na rozpoznawanie gestów, co było największą trudnością. Bardzo duże znaczenie odgrywa dokładność binaryzacji, a tym samym dokładność rozpoznawania samego kształtu. Ważny jest również dobór wykorzystywanych współczynników. Niestety, duża czułość współczynników kształtu i bardzo duże wykorzystanie procesora przez aplikację, uniemożliwiło dokładne i bezbłędne rozpoznanie wszystkich gestów. W trakcie prac nad projektem trzeba było zrezygnować z części wymyślonych usprawnień – np. modułu uczącego się i własnych konfiguracji użytkownika.

Korzystanie z kamery, która nie dostosowuje swoich parametrów przy zmianie kąta padania światła albo zmianie odległości, daje zauważalnie lepsze rezultaty.

Aplikacja na pewno jest wygodniejsza w użyciu niż Visual Mouse i posiada więcej możliwości. Dodanie modułu „uczenia", co można osiągnąć w przyszłości, poszerzy jeszcze jego zastosowania. Projekt można rozwijać dalej, implementując kolejne sposoby wyliczania prawdopodobieństwa gestów, zwiększając dokładność obliczeń lub dodając nowe współczynniki kształtu.

Praca nad projektem pozwoliła nam powtórzyć informacje dotyczące algorytmów przetwarzania obrazu, w szczególności binaryzacji i segmentacji. Rozpoznawanie gestów za pomoca kamery i prostej aplikacji okazało się zadaniem trudnym, ale wykonalnym. Wydaje się, że wprowadzanie kolejnych algorytmów poprawiających skuteczność wykrywania odpowiednich gestów może bardzo poprawiać działanie aplikacji. Innym sposobem mogłoby być rozpoznawanie obiektu w dziedzinie częstotliwości, co wydaje się być prostsze i pochłaniające mniej zasobów procesora.

5. Podsumowanie

Zostały spełnione założenia projektowe z wyjątkiem modułu uczącego się gestów pokazywanych przez użytkownika. Jego implementacja jest prosta, wymaga jednak lepszego dopracowania części głównej aplikacji.

6. Literatura

[1] Margrit Betke, James Gips, Peter Fleming "The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access for People With Severe Disabilities", IEEE Transactions Of Neural Systems And Rehabilitation Engineering, Vol. 10, No. 1, March 2002

[2] http://www.5star-shareware.com/Windows/Desktop/EnhancementUtilities/visual_mouse_ii.html
[3] L. Young and D. Sheena, “Survey of eye movement recording methods,” Behav. Res. Meth. Instrum., 1975
[4] R. B. Reilly and M. J. O’Malley, “Adaptive noncontact gesture-based system for augmentative communication,” IEEE Trans. Rehab. Eng., vol. 7, no. 2, 1999
[5] http://www.emgu.com/wiki/files/2.4.2/document/Index.html
[6] http://www.aforgenet.com/framework/docs/

7. Załączniki

[1] Tabela wyników wyliczania współczynników w programie MATLAB
[2] Wzory na współczynniki kształtu
[3] Opis klas i funkcji projektu

7. DODATEK A: Opis opracowanych narzędzi i metody postępowania

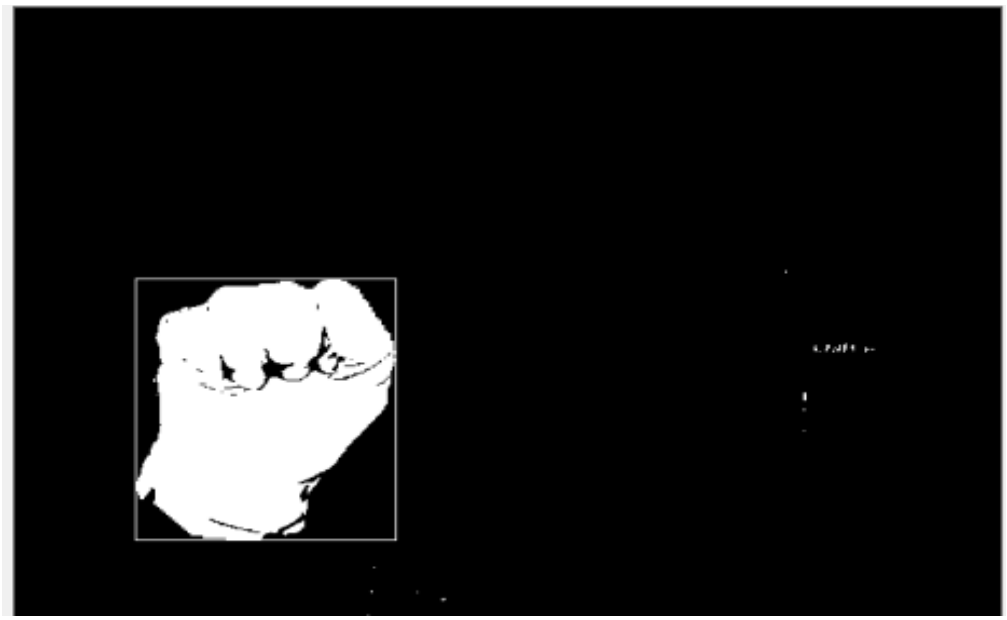
Środowisko programistyczne: Visual Studio
Język : C#
Użyte biblioteki: EmguCV, Aforge

Uruchomienie aplikacji od razu pozwala na korzystanie z gestów. Wystarczy nacisnąć piksel na lewym obrazie, aby zbinaryzować obraz względem niego. Użytkownik może zmieniać progi binaryzacji, użyty model kolorów lub ustawienia kamery. Po prawej stronie panelu wyświetlane są rozpoznane współczynniki oraz gest.

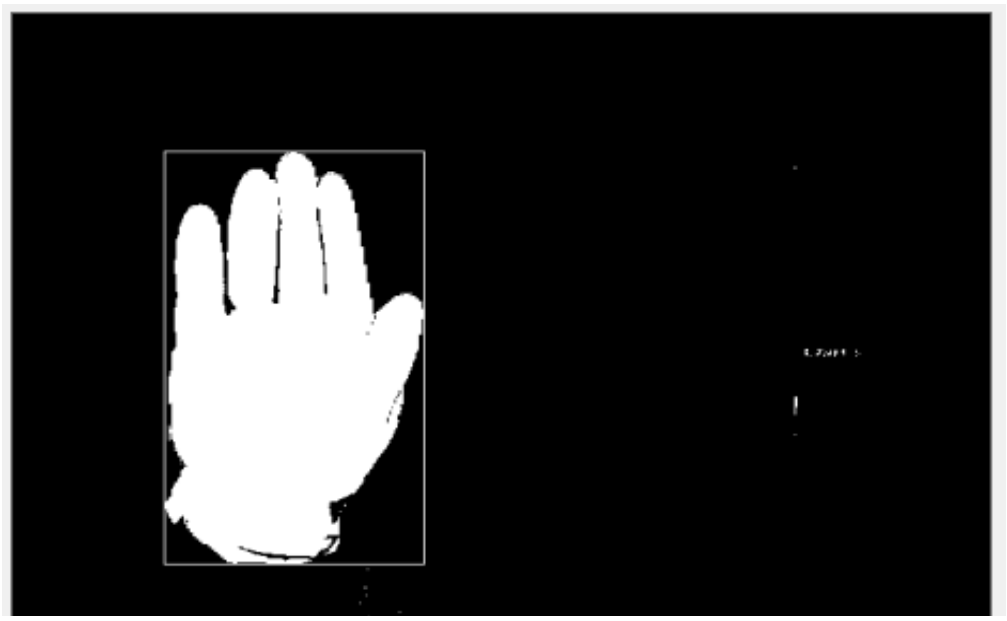
Globalne skróty klawiszowe: Ctrl+Alt+Shift+Z, który włącza/wyłącza sterowanie myszy gestami oraz Ctrl+Alt+Shift+X, który odpowiada za minimalizację i pojawianie się okna.

8. DODATEK B: Opis wykrywanych gestów

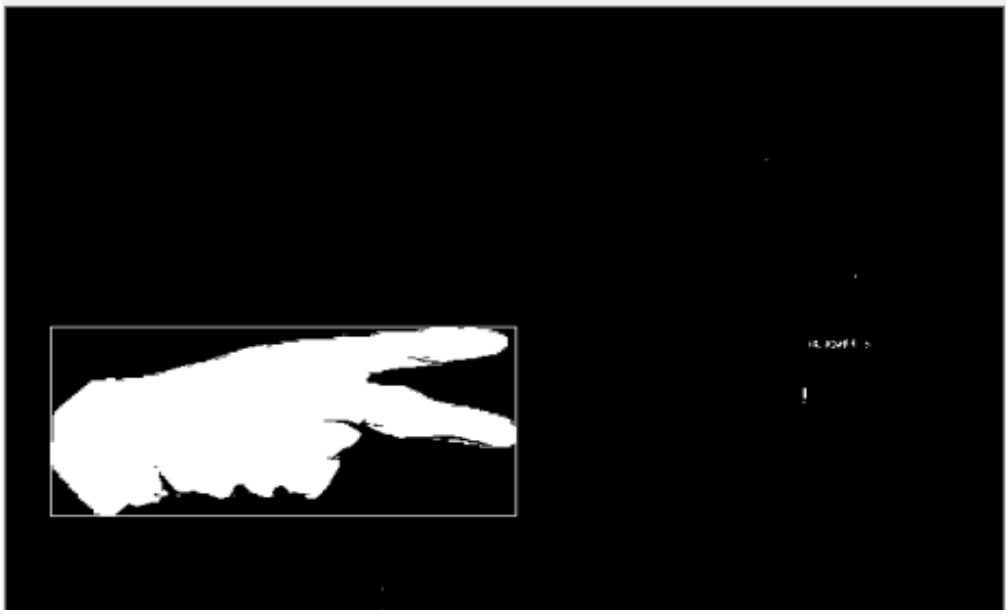
- Kliknięcie LPM – prawa ręka „fist”
- Wciśnięcie LPM – lewa ręka „fist”
- Zwolnienie LPM – lewa ręka „vopen”
- Scroll UP – prawa ręka „slayer”
- Scroll DOWN – prawa ręka „victory”
- Kliknięcie PPM – prawa ręka „fingers”
- Ctrl+C – prawa ręka „thumbup”
- Ctrl+V – prawa ręka „thumbleft”
- Ctrl+X – prawa ręka „scissors”
- Kliknięcie bocznego przycisku myszy – prawa ręka „shaka”



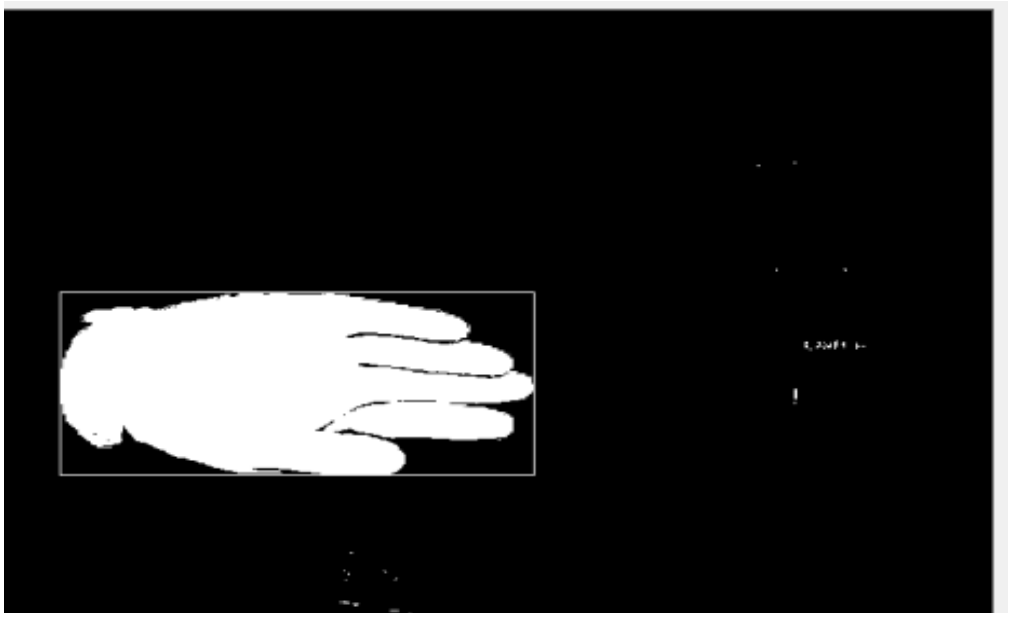
Grafika 2: Gest "fist"



Grafika 3: Gest "vopen"

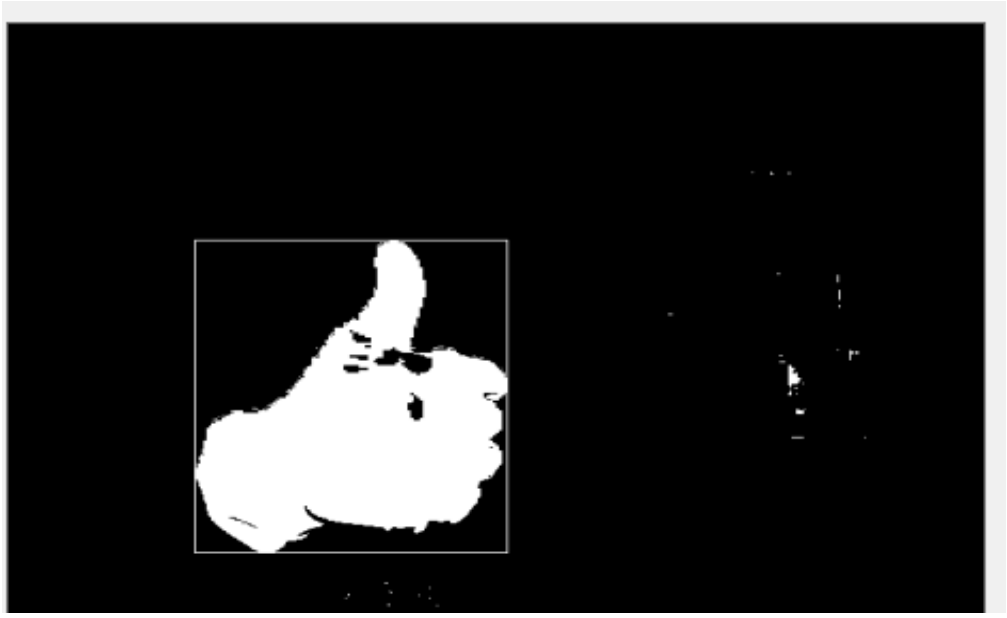


Grafika 4: Gest "scissors"

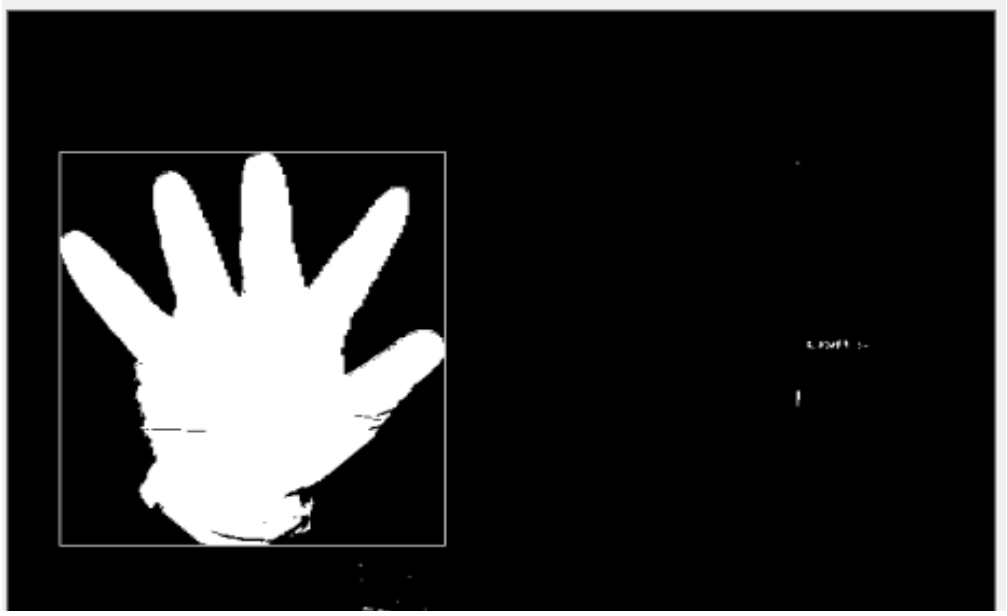


Grafika 5: Gest "hopen"

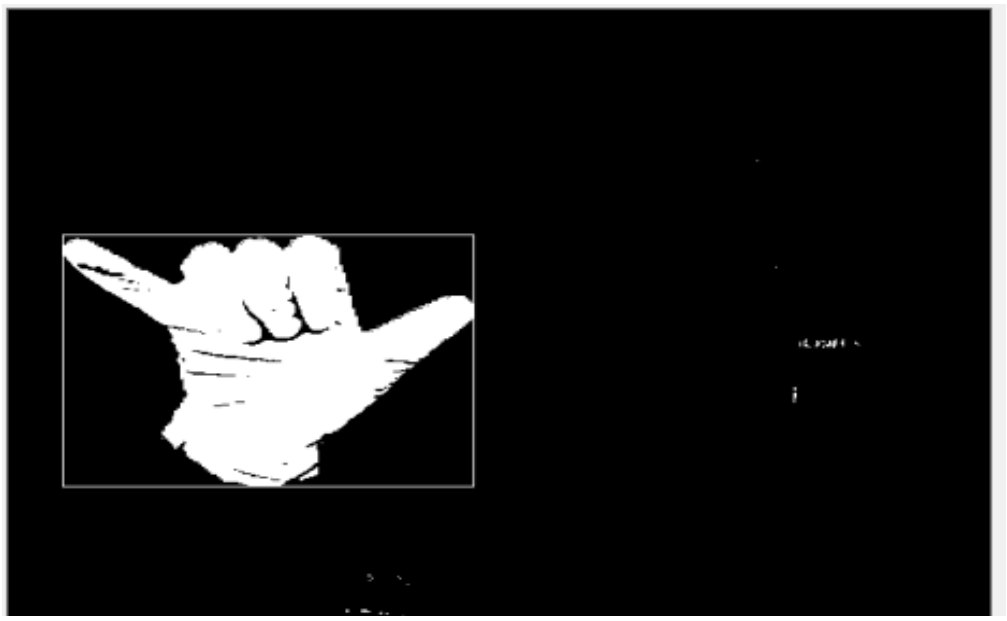
Grafika 6: Gest "thumbleft"



Grafika 7: Gest "thumbup"



Grafika 8: Gest "fingers"



Grafika 9: Gest "shaka"

Grafika 10: Gest "slayer"

Grafika 11: Gest "victoria"