POLITECHNIKA WARSZAWSKA
Wydział Elektroniki i Technik Informacyjnych
Instytut Informatyki

Rok akademicki
2009/2010

Praca Dyplomowa Inżynierska

Damian Turczyński

Interfejsy użytkownika bazujące na rozpoznawaniu ruchu i gestów dłoni

Opiekun pracy:
mgr inż. Krzysztof Gracki

Ocena: .....

.....
Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Damian Turczyński

Specjalność: Inżynieria Systemów Informatycznych
Data urodzenia: 09.10.1987 r.
Data rozpoczęcia studiów: 01.10.2006 r.

Życiorys

Urodziłem się 09.10.1987 r. w Mielcu. Po ukończeniu szkoły podstawowej i gimnazjum naukę kontynuowałem w IX Liceum Ogólnokształcącym im. Bohaterów Monte Cassino w Szczecinie. W szkole średniej uczęszczałem do klasy o profilu informatyczno-matematycznym. W październiku 2006 r. rozpocząłem studia na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej.

.....

Damian Turczyński

EGZAMIN DYPLOMOWY

Złożył egzamin dyplomowy w dniu .....2010 r

z wynikiem .....

Ogólny wynik studiów: .....

Dodatkowe wnioski i uwagi Komisji: .....

.....

.....

Interfejsy użytkownika bazujące na rozpoznawaniu ruchu i gestów dłoni

Streszczenie

Praca traktuje o interfejsach użytkownika służących do komunikacji człowieka z komputerem. Zawarty jest w niej obszerny opis systemów rozpoznawania obrazów oraz współczesnych interfejsów użytkownika. Omówienie jest napisane pod kątem detekcji ruchu i gestów dłoni za pomocą kamery internetowej.

Praktyczną część pracy stanowi implementacja systemu pozwalającego na komunikowanie się z komputerem w czasie rzeczywistym. Przedstawiono wymagania systemu, założenia projektowe oraz koncepcję rozwiązania wraz z przeprowadzonymi testami. Stworzono w pełni funkcjonalną aplikację pozwalającą na symulację myszy komputerowej za pomocą gestów i ruchu dłoni.

Dodatek stanowi instrukcja obsługi stworzonego programu wraz ze zwięzłym przedstawieniem biblioteki OpenCV.

Słowa kluczowe: *dłoń*, *interfejs*, *rozpoznawanie*, *obraz*, *kamera*

User interfaces based on recognition of hand motion and gestures

Abstract

This thesis refers to the user interfaces which allows people to communicate with PC. It includes a comprehensive description of the pattern recognition and modern user interfaces. Discussion is focused on motion and hand gestures recognition using a webcam.

The practical part of this thesis concerns the implementation of a system that allows communication with the computer in real time. The paper contains system requirements, project goals and the concept of a solution with carried out tests.

The appendix contains manual to created program and concise presentation of the OpenCV library.

Keywords: *hand*, *interface*, *recognition*, *picture*, *webcam*

Spis treści

Rozdział 1

Wstęp

Obecnie zastosowanie komputerów w życiu codziennym jest coraz powszechniejsze. Komputery wykonują za nas coraz więcej prac i służą już nie tylko jako zaawansowane maszyny liczące, ale towarzyszą nam podczas wykonywania codziennych zajęć i rozrywki. Rozwój techniki spowodował, że w celu interakcji z maszyną nie wystarczy już klawiatura i myszka. Coraz większą popularność zyskują panele dotykowe, zaawansowane kontrolery i czujniki. Chcielibyśmy by komputer rozumiał nasze intencje, mowę, gesty, a nawet zachowania. Jednocześnie staramy się by porozumiewanie się z maszyną było jak najbardziej intuicyjne i proste. Tworząc coraz bardziej zaawansowane metody percepcji komputerowej, przybliżamy się do stworzenia „sztucznego człowieka", będącego jak najlepszą imitacją nas samych.

Najbardziej intuicyjną formą przekazywania informacji pomiędzy ludźmi jest mowa oraz gestykulacja. Badania psychologiczne mówią, że gesty człowieka to aż 65% informacji przekazywanej podczas rozmowy [41]. Komputer potrafiący zrozumieć gesty wykonywane przez człowieka jest więc w stanie poznać jego intencje i zareagować w odpowiedni sposób.

Wiele książek i filmów, jak np.: książka Isaak'a Assimova pt. „Ja Robot" dotykało problemu uczłowieczenia komputerów. Obecnie inżynierowie pracują nad tym, aby maszyna była w stanie jak najlepiej rozpoznawać intencje, gesty oraz mowę człowieka. Niniejsza praca jest tego potwierdzeniem, jak również przekonuje o istotności zapotrzebowania na tego typu rozwiązania, tym bardziej, że interfejs oparty na wizji jest dopiero we wczesnym stanie rozwoju i wydaje się być niszowym.

W pracy skupiono się na zagadnieniu rozpoznawania przez komputer gestów wykonywanych dłonią oraz ruchów dłoni w celu stworzenia interfejsu HID (*Human Interface Device*), dzięki któremu przy pomocy kamery internetowej użytkownik może sterować komputerem [40]. Program taki może również posłużyć do sterowania innymi urzą-

dzeniami, robotami czy zabawkami technicznymi wyposażonymi w kamerę internetową. Zastąpienie myszy komputerowej rozpoznawaniem obrazu z kamery jest obecnie często brane pod uwagę jako zagadnienie przyszłościowe i warte uwagi [48] [43].

W typowych zastosowaniach w komunikacji użytkownika z komputerem klawiatura i myszka wydaje się być wystarczająca. Jednak w przypadku, gdy mamy do czynienia np. z wirtualną rzeczywistością czy bardziej interaktywnymi grami, rozpoznawanie gestów dłoni staje się dużo bardziej atrakcyjne [51].

Za pomocą systemu, który został stworzony podczas pisania pracy, możliwa jest obsługa kursora myszy komputerowej i wykonywanie kilku operacji za pomocą dłoni. Obecnie, gdy ekrany komputerów przeważnie są dwuwymiarowe, słuszność tego rozwiązania jest wątpliwa, jednak w przekonaniu autora, rozwój techniki zaowocuje trójwymiarowymi wyświetlaczami, oraz wyświetlaniem obrazu w przestrzeni. Niemożliwość adaptacji myszki do poruszania kursora w trzech wymiarach wyeliminuje ją w przyszłych rozwiązaniach na rzecz bardziej zaawansowanych urządzeń lub rozpoznawaniem samej dłoni i gestów nią wykonywanych na podstawie obrazu z kamery, bądź kilku kamer.

Zaimplementowane rozwiązanie nadaje się również do zastosowań w dziedzinie robotyki, podczas gdy człowiek chce wydawać polecenia robotowi. W celu wydania polecenia najłatwiej jest wydać komendę głosowo, lub po prostu pokazać robotowi ręką co ma zrobić. Rozwiązania oparte na rozpoznawaniu mowy są jednak wrażliwe na głośne dźwięki w tle, jak również na akcent różnych autorów oraz język, jakim się oni posługują. Proste gesty są naturalne i intuicyjne niezależnie od narodowości, przez co komendy typu stop, jedź, zawróć byłyby oczywiste jak te wydawane psu podczas tresury.

1.1 Cel pracy

Wykorzystanie kamery internetowej w celu stworzenia interfejsu użytkownika do komunikacji z komputerem jest głównym zadaniem, które postawiono pisząc tę pracę. Tworzony dla osiągnięcia tego celu program powinien być możliwie prosty w obsłudze i intuicyjny dla odbiorcy.

Dodatkowo w pracy zostały przebadane algorytmy segmentacji koloru skóry, oraz rozpoznawania gestów dłoni.

Zaprojektowanie systemu wymagało rozwiązań następujących zagadnień:

- propozycja i przebadanie metody segmentacji dłoni z obrazu, na podstawie modelu barwy skóry ludzkiej;
- przebadanie wpływu czynników świetlnych na proces segmentacji;

- opracowanie metody analizy gestów wykonywanych dłonią na podstawie ekstrakcji niezmienników momentowych z obrazów dłoni uzyskanych podczas segmentacji;
- opracowanie metody wykrywania i śledzenia ruchu dłoni w obrazie z kamery;
- zaprojektowanie metody wysyłania komunikatów do systemu w celu przekazania informacji uzyskanej od użytkownika programu.

1.2 Przegląd pracy

W pierwszej części pracy zostały omówione i przedstawione podstawowe składniki systemów rozpoznawania obrazów. Wszystkie etapy rozpoznania obrazu od akwizycji poprzez przetwarzanie, segmentację cech, aż do analizy cech zostały omówione w rozdziale 2.

Rozdział 3 traktuje o współczesnych interfejsach człowiek - komputer. Znajduje się w nim podział interfejsów i przykłady istniejących rozwiązań.

W rozdziale 4 omówiono budowę zaimplementowanego systemy. W podrozdziale 4.2 założenia poczynione przy tworzeniu pracy i wymagania, przed którymi musiał stanąć autor.

Implementacja systemu niosła za sobą wiele interesujących rozwiązań. W podrozdziale 4.4.2 omówiono sposób, w jaki następuje wykrywanie dłoni za pomocą systemu zaimplementowanego wraz z pracą. Następnie podrozdział 4.5 zawiera informacja o rozpoznawaniu gestów na przykładzie aplikacji.

Dokładne i płynne rozpoznawanie ruchu było priorytetem podczas tworzenia aplikacji. Autorski sposób śledzenia dłoni oraz przeniesienie rozpoznanych wyników na ruch myszką zostały opisane w podrozdziale 4.7.

Podsumowanie wyników rozpoznawania gestów dłoni i śledzenia trajektorii ruchu oraz uwagi wynikające z implementacji systemu zamieszczono w rozdziale 5.

Uzupełnienie pracy stanowią dodatki, w których można znaleźć instrukcję obsługi programu (Dodatek A) i omówienie biblioteki OpenCv (Dodatek B) oraz dysk CD, na którym można znaleźć zaimplementowany system wraz z pracą w formie elektronicznej.

Rozdział 2

Systemy rozpoznawania obrazów

Człowiek potrafi rozpoznawać i reagować na bodźce z otoczenia. W trakcie dorastania uczy się on znaczenia poszczególnych sygnałów pochodzących ze świata. Identyfikacja obrazów staje się wtedy naturalna i oczywista dla każdego z nas. Jednym z najważniejszych narządów percepcji człowieka jest wzrok. To on dostarcza do mózgu dane o otoczeniu w postaci obrazów, które tak wiele mówią nam o środowisku, w którym się znajdujemy. W celu przystosowania maszyny do odbierania i przetwarzania informacji zawartej w wizji należy wykonać wiele czynności, począwszy od akwizycji obrazu, aż po podjęcie decyzji na podstawie informacji w nim zawartej.

Systemy wizyjne stają się coraz popularniejsze w przemyśle i rozrywce dzięki wielu możliwościom zastosowania ich dla ułatwienia życia. Przetwarzanie obrazu satelitarnego, analiza zdjęć medycznych, rozpoznawanie twarzy w systemach zabezpieczeń to tylko nieliczne z zastosowań, o których słyszy się coraz więcej. Niestety takie systemy są niesłychanie trudne w implementacji, co często jest zaskakujące dla człowieka, gdyż nasz mózg wykonuje wszystkie operacje związane z rozpoznawaniem obrazów natychmiastowo i bez większego wysiłku. Jeżeli jednak dochodzi do zaimplementowania wyżej wymienionego systemu okazuje się, że istnieje szereg bardzo złożonych problemów, które należy rozwiązać.

Dla umożliwienia stworzenia systemu opartego na wizji należy w pierwszej kolejności zamienić obraz, który nas otacza na postać cyfrową. Etap ten nazywa się akwizycją, bądź też digitalizacją. Następnie przetworzony na postać cyfrową obraz poddaje się przetwarzaniu i poprawianiu jakości. W szczególności w tym etapie nastąpi filtracja, redukcja zakłóceń czy wyostrzanie obrazu. Poprawiony już obraz przechodzi do etapu segmentacji by wydzielić obszary, które są interesujące w danym rozwiązaniu. Ze znalezionych obszarów wydziela się cechy, które następnie służą do analizy obrazu. Przeanalizowany obraz może służyć jako wynik bezpośredni (np. w medycynie wynikiem będzie czy występuje

Rysunek 2.1: Schemat systemu rozpoznawania robrazów

dane schorzenie), bądź też jako dane do sterowania urządzeń.

Schemat przedstawiony na rysunku 2.1 pokazuje etapy rozpoznawania obrazów na przykładzie programu rozpoznającego dłoń, jej ruch i gesty w celu sterowania kursorem myszki zaimplementowany wraz z tą pracą.

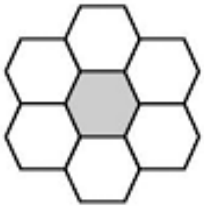
Poniżej został omówiony szczegółowo każdy z etapów rozpoznawania obrazów.

2.1 Akwizycja obrazu

Akwizycja obrazu inaczej digitalizacja jest to przetworzenie rzeczywistego obrazu w postać możliwą do zapisania i przetwarzania w komputerze. Urządzenia służące do akwizycji są tworzone by jak najlepiej naśladować ludzki wzrok i zapisać informację w sposób pozwalający na późniejszą jej zamianę na postać widzialną.

Dzisiejszy poziom techniki nie pozwala jednak zapisać obrazu w sposób tak doskonały i z taką prędkością jak jest w stanie zrobić to człowiek za pomocą wzroku. Nasz mózg przetwarza obrazy z otoczenia z szacunkową prędkością co najmniej 100 Mb/s [45], co nie zawsze jest możliwe dla popularnych obecnie komputerów. Fakt ten powoduje, że zapis obrazu jest obecnie zawsze kompromisem pomiędzy prędkością, a zasobochłonnością i jakością zapisywanego obrazu.

Ograniczenia, które towarzyszą przy akwizycji można podzielić na wiele grup. Będą to ograniczenia związane z ostrością, czyli możliwością rozpoznawania szczegółów, z ilością kolorów, jak również ilością wymiarów, w których dany obraz zostaje zapisany. W dzisiejszych czasach coraz bardziej popularne stają się filmy trójwymiarowe, jednak w celu



Rysunek 2.2: Siatka heksagonalna

Rysunek 2.3: Siatka kwadratów

stworzenia obrazu przestrzennego wymagane są niewiarygodne ilości pamięci cyfrowej i sprzętu. Dodatkowo analiza trójwymiarowego obrazu jest dużo trudniejsza i w słabszym stopniu rozwinięta technicznie aniżeli analiza obrazu w dwóch wymiarach.

W kolejnych podrozdziałach omówiono typowe właściwości dzielące cyfrowe obrazy.

2.1.1 Typ rastra

W celu przetworzenia obrazu widzialnego na obraz komputerowy dokonuje się pikselizacji przestrzeni widzialnej, czyli zapisanie obrazu w postaci szeregu małych jednokolorowych punktów, zwanych w informatyce pikselami (od angielskich słów *picture-element*, czyli element obrazka). Piksele są najczęściej kwadratowe i tworzą siatkę składającą się na obraz.

Znane są również inne reprezentacje bliższe percepcji człowieka jak np.: siatka heksagonalna (Rysunek 2.2). Tworzą ją sześciokątne płaszczyzny o strukturze plastra miodu. Obraz w ten sposób zapisany i wyświetlony będzie dokładniej i w bardziej naturalny sposób przedstawiał rzeczywistość w odbiorze człowieka. Metoda ta jest jednak dużo trudniejsza w implementacji i analizie komputerowej. Dlatego też najczęściej spotykaną reprezentacją jest postać siatki kwadratów (Rysunek 2.3), zwanych rastrami kwadratowymi. Zapisywana jest ona w pamięci komputera przeważnie liniami (w postaci surowej, tzn. bez kompresji).

Przedstawienie obrazu w postaci siatki prostokątnej pozwala na proste pobieranie kolejnych pikseli. Jest to użyteczne w algorytmach przetwarzania i analizy, dzięki czemu sposób ten rozpowszechnił się w dzisiejszej informatyce i jest używany w zdecydowanej większości urządzeń związanych z wizją. W pracy została użyta również ta reprezentacja obrazu, więc została ona omówiona dalej dokładniej niż mało popularna siatka heksagonalna.

2.1.2 Częstotliwość próbkowania

W celu zapisania sceny należy przetworzyć obraz rzeczywisty w skwantowany obraz binarny. Rzeczywistość, która nas otacza jest ciągła. Chcąc zapisać obraz w pamięci

komputera musimy skorzystać z uproszczenia. Matematycznie obraz rzeczywisty można traktować jako funkcję ciągłą `f(x,y)`, która mówi o kolorze dla dowolnych parametrów x i y będących położeniem w przestrzeni dwuwymiarowej. Obraz binarny uzyskuje się poprzez zamianę funkcji ciągłej w funkcję dyskretną. Dokonuje się tego poprzez próbkowanie obrazu widzialnego otrzymując macierz pikseli, czyli fizyczną reprezentację obrazka w pamięci komputera.

Dla uchwycenia obrazu, który będzie dobrze naśladował rzeczywistość należy zapisać dostatecznie dużo małych pikseli na jednostkę powierzchni, a co za tym idzie posiadać urządzenie wykorzystujące odpowiednią liczbę sensorów. Wymiary siatki określają należytą częstotliwość próbkowania. Szczegóły będące mniejsze niż okres próbkowania (będący w tym przypadku odległością) nie zostaną zarejestrowane w obrazie binarnym. Zgodnie z twierdzeniem Nyquist'a każde urządzenie cyfrowe może zarejestrować szczegół, który jest nie większy niż dwa okresy próbkowania urządzenia. Stąd też im dokładniejszy obraz chce się uzyskać tym należy próbkować z większą częstotliwością. Przedmioty będące większe niż połowa odległości próbkowania będą ulegały aliasingowi przestrzennemu, przez co staną się nierozróżnialne.

Aliasing ma miejsce, gdy liczba sensorów jest zbyt mała w stosunku do szczegółów zapisywanego obrazu o dużych częstotliwościach przestrzennych. Efekt aliasingu obserwuje się w miejscach obrazu o skokowych zmianach jasności, zwłaszcza w przypadku powtarzalnych wzorów.

Wybór częstotliwości próbkowania jest ściśle związana z rozdzielczością obrazu. Rozdzielczość jest to liczba pikseli na jednostkę powierzchni. Wybór odpowiedniej rozdzielczości jest sprawa kluczową w akwizycji obrazu. Im większa rozdzielczość tym więcej szczegółów jesteśmy w stanie zarejestrować, jednak wiąże się to z większymi wymaganiami pamięciowymi, a w przypadku późniejszej analizy również z obliczeniowymi. Stosunek ten jest kwadratowy. W przypadku złożonych algorytmów przetwarzania i analizy obrazów wzrost ten może być jeszcze większy.

W pracy przebadano użycie różnych rozdzielczości. Większe rozdzielczości powodowały, że program działał znacznie wolniej (dużo więcej obliczeń), zobacz tabelę 2.1. Długość obliczeń nie przekładała się na jakość rozpoznawania, dlatego też ostatecznie zdecydowano się na użycie mniejszej rozdzielczości.

Ostatecznie zdecydowano się na użycie rozdzielczości 320 na 240 pikseli. Dodatkowym atutem użycia mniejszej rozdzielczości jest fakt, że program można używać na większej ilości sprzętu. Ponadto kamery posiadające możliwość rejestracji z większą rozdzielczością posiadają zdolność dokładniejszej reprezentacji obrazu o mniejszej niż maksymalna rozdzielczości poprzez zbieranie informacji z kilku sensorów na jeden piksel wynikowy.

Rozdzielczość	Liczba klatek na sekundę
320 na 240	40-60
480 na 320	20-40
640 na 480	16-20
1024 na 768	4-6

Tabela 2.1: Wyniki badania szybkości działania programu dla różnych rozdzielczości. Test wykonany na komputerze z procesorem Core 2 Duo 2GHz i kamerze internetowej 1,3 mega pikseli.

2.1.3 Liczba kolorów

Każdy z elementów graficznej reprezentacji obrazu, czyli pojedynczych pikseli może przyjmować tylko jedną spośród ustalonej liczby wartości. Liczba ta mówi o głębi koloru obrazka i jest przyjęte podawanie tej wartości jako liczba bitów potrzebnych do zakodowania jednego elementu obrazka. Oznacza się ją jako bit na piksel (*ang. bit per pixel*). Im większa liczba poziomów, tym więcej informacji. Najpowszechniej stosowane są 3 głębie kolorów [45].

- Binarny obraz czarno - biały (1 bit na piksel)
- Obraz w skali szarości (8 bitów na piksel lub czasami 16 bitów na piksel) - informacje jedynie o jasności pikselów.
- Obraz kolorowy (24 bity na piksel, 32 bity na piksel czasem więcej) zapisane są informacje o kolorach oraz jasności. Najczęściej stosowany format zapisu to RGB, czyli reprezentacja kolejnych trzech składowych barw czerwonej, zielonej i niebieskiej jako ósemki kolejnych bitów – 24 bity na piksel. 24 bity na jeden piksel w zupełności wystarczają do typowych zastosowań

Przy rozpoznawaniu obrazów najczęściej stosuje się zasadę, że obraz wejściowy jest pełno kolorowy, a następnie odrzuca się kolory nieinteresujące by zmniejszyć ilość pamięci potrzebnej do dalszej analizy. W programie rozpoznawania gestów stworzonym z tą pracą następuje wykrycie skóry z informacji zawartej w pełnym kolorze, a następnie obraz jest progowany do postaci binarnej.

Obecnie technika pozwala na analizowanie w czasie rzeczywistym jedynie obrazów statycznych. Informacje zawarte w kolejnych klatkach służą jednak często jako dane pomocnicze do analizy kolejnych obrazów. Tak też czynione jest w tej pracy podczas analizy ruchu dłoni. Zapewne w przyszłości komputery będą analizować obraz zarówno dynamiczny, jak i w trój wymiarze, w celu uzyskania jak najlepszych wyników.

2.1.4 Akwizycja, a warunki oświetlenia

Podczas akwizycji obrazów kluczowe są warunki, w jakich następuje rejestracja obrazu. Jeżeli obraz będzie niedoświetlony, bądź też prześwietlony, większość pikseli będzie odpowiednio zbyt ciemna lub za jasna. Informacja zawarta w takim obrazie będzie znikoma, przez co późniejsze przetwarzanie i analiza będzie utrudniona, a w niektórych przypadkach wręcz niemożliwa.

Dla jak najlepszych wyników osiąganych w systemach rozpoznawania obrazów warunki akwizycji (jasność, barwa oświetlenia, długość otwarcia migawki aparatu) powinny być jednakowe dla obrazów służących do kalibracji systemu i obrazów badanych. Często jednak dla rzeczywistych zastosowań nie jest możliwe zapewnienie identycznego oświetlenia w każdym przypadku. W systemach wizji rozróżnia się czasami oświetlenie słoneczne od sztucznego i implementuje dwie różne metody przetwarzania w zależności od oświetlenia. Inną metodą radzenia sobie z tym problemem jest staranne ustawianie warunków akwizycji w urządzeniu, a przede wszystkim balansu bieli, czyli barwy odniesienia dla koloru białego.

Często, aby uniezależnić się od jasności przetwarzanych i analizowanych obrazów, konwertuje się przestrzenie barw obrazka ze standardowej przestrzeni RGB na mniej zależne od jasności przestrzenie np.: HSV czy YCbCr (przestrzenie te zostały opisane w dalszej części pracy). Ma to na celu wydzielenie informacji o jasności i usunięcie jej ze zdigitalizowanego obrazka.

2.1.5 Przestrzenie kolorów

Informacja o kolorze na zdjęciu może być zapisywana na wiele sposobów. Typowo korzysta się z modeli matematycznych reprezentujących widma fal elektromagnetycznych w zakresie od 300 do 830 nm., czyli światło widzialne [35]. Klasyczne modele są trójwymiarowymi przestrzeniami barw odzwierciedlającymi część wszystkich możliwych kolorów. Pełną informację o kolorze zapisuję się w trzech składowych, interpretowanych różnie w zależności od przestrzeni. Warto dodać, że żadna z przestrzeni nie jest w stanie przedstawić wszystkich kolorów występujących w naturze.

Wyróżnia się dwa typy przestrzeni barw, są to przestrzenie sprzętowe i percepcyjne. Pierwsze odzwierciedlają fizyczną reprezentację kolorów i zaliczamy do nich przestrzenie RGB, CMYK i pochodne. Swoje pochodzenie zawdzięczają doborze urządzeń w celu akwizycji koloru. Standardowo używa się trzech receptorów wrażliwych na różne częstotliwości światła i ta informacja koduje kolor. Kolejnym typem przestrzeni są modele percepcyjne. Przechowują one informację o kolorze w sposób bardziej naturalny dla człowieka. Poszczególne składowe wydzielają jasność koloru i informację o barwie. Do

tych przestrzeni zaliczamy np. HSV, YCbCr, YIQ [9] i wiele innych. W pracy zostały omówione jedynie używane przestrzenie barw.

Przestrzeń RGB

Przestrzeń kolorów RGB jest najbardziej rozpowszechnioną w grafice komputerowej. Większość obrazów w formie binarnej jest zapisywana w tej właśnie przestrzeni. Kolor jest kodowany w niej jako addytywna kombinacja trzech składowych kolorów: czerwony - R (*ang. red*), zielony – G (*ang. green*) i niebieski – B (*ang. blue*). Główną zaletą tego systemu jest jego prostota. Jednakże nie jest to przestrzeń zunifikowana dla percepcji człowieka, tzn. liniowa zmiana koloru w tej przestrzeni nie przenosi się na liniowe odczucie zmiany barwy przez obserwatora – człowieka. Co więcej w RGB składowa luminancji jest zawarta w każdej z trzech składowych R, G i B, przez co składowe są silnie skorelowane. Fakt ten powoduje, że w rozpoznawaniu skóry ludzkiej (ale również i innych barw) nie można brać pod uwagę pojedynczych składowych, lecz należy uwzględnić zależności pomiędzy kilkoma z nich [30].

Dodatkowym problemem mającym duży wpływ na klasyfikację w przestrzeni RGB jest oświetlenie, w którym nastąpiła akwizycja obrazu. Różne oświetlenie będzie skutkować różnym przedstawieniem tego samego koloru w przestrzeni barw. Kolejny problem pojawia się, gdy chcemy rozpoznawać kolor skóry różnych ras ludzkich. W przestrzeni RGB, dla różnych ras, otrzymywać będziemy zupełnie różne wyniki.

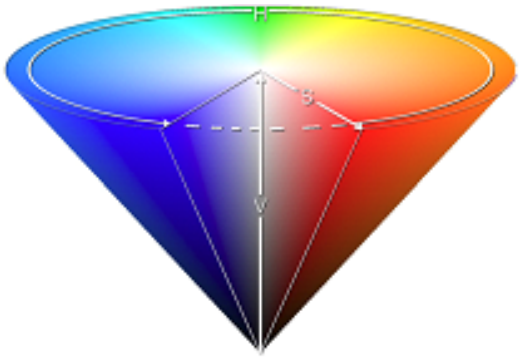
Przestrzeń HSV

Przestrzeń HSV różni się od modelu RGB informacją jaka jest przechowywana w poszczególnych składowych koloru. Jest ona bardziej intuicyjna dla percepcji człowieka i pozwala osiągać lepsze rezultaty przy rozpoznawaniu skóry ludzkiej [7] [30] [39] [44]. Swoją przewagę nad modelem RGB zawdzięcza faktowi, iż informacja o jasności jest wydzielona do pojedynczej składowej. Dzięki temu wraz ze zmianą jasności zmienia się tylko jedna składowa, a nie wszystkie trzy jak w przypadku przestrzeni RGB.

Przestrzeń kolorów HSV reprezentuje kolory w postaci trzech składowych barwa – głębia koloru (*ang. Hue*), nasycenie – czystość koloru (*ang. Saturation*) oraz jasność (*ang. Value*). Inne nazwy tej samej przestrzeni to HSI (*Hue, Saturation, Intensity* - barwa, nasycenie, intensywność), HSB (*Hue, Saturation, Brightness* - barwa, nasycenie, jasność).

Zakresy poszczególnych składowych to:

- dla składowej Hue 0-360 (w niektórych aplikacjach znormalizowane do 0-100%). Odpowiada ona barwie. Kolory ustawione są kolejno od barw czerwonych przez



Rysunek 2.4: Reprezentacja graficzna HSV

zielone do niebieskich, pomiędzy którymi występują barwy pośrednie. W przestrzeni składowa ta jest reprezentowana jako okrąg stożka, na krawędzi którego znajdują się kolory czyste – o maksymalnym nasyceniu;

- dla składowej Saturation – 0-100% - reprezentuje ona nasycenie koloru, innymi słowy czystość koloru. Im mniejsze nasycenie tym kolor staje się bardziej szary. W reprezentacji graficznej wartość ta jest odległością koloru od środka stożka. Nasycenie jest główną składową różnicującą kolor skóry od innych kolorów. Prawidłowy dobór wartości tej składowej w znacznym stopniu zapewnia poprawność określania koloru skóry [50].

- dla składowej Value – 0 -100%. Reprezentuje ona jasność koloru. Im wyższa wartość tym kolor jest jaśniejszy. W reprezentacji przestrzennej wartość ta zapisywana jest jako wysokość na stożku kolorów.

Transformacja pomiędzy przestrzenią RGB, a HSV nie jest liniowa. Wzory na przekształcenie przedstawiają się następująco:

$$\begin{aligned} V &= \tfrac{1}{3}R + \tfrac{1}{3}G + \tfrac{1}{3}B \\ X &= \tfrac{1}{\sqrt{6}}R - \tfrac{1}{\sqrt{6}}G + \tfrac{2}{\sqrt{6}}B \\ Y &= \tfrac{1}{\sqrt{6}}R - \tfrac{1}{\sqrt{6}}G \\ H &= \arctan(\tfrac{Y}{X}) \\ S &= (X^2 + Y^2)^{\frac{1}{2}} \end{aligned} \tag{2.1}$$

Lub gdy wykonujemy przekształcenie na komputerze to znacznie bardziej efektywny

będzie algorytm:

$$
\begin{aligned}
H &= \begin{cases}
\text{nieokreślone} & \text{dla MAX} = \text{MIN} \\
60 \times \frac{G-B}{MAX-MIN} & \text{dla MAX} = \text{R i G} \geq \text{B} \\
60 \times \frac{G-B}{MAX-MIN} + 360 & \text{dla MAX} = \text{R i G} < \text{B} \\
60 \times \frac{G-B}{MAX-MIN} + 120 & \text{dla MAX} = \text{G} \\
60 \times \frac{G-B}{MAX-MIN} + 240 & \text{dla MAX} = \text{B}
\end{cases} \\
S &= \begin{cases}
0 & \text{dla MAX} = 0 \\
1 - \frac{MIN}{MAX} & \text{w przeciwnym przypadku}
\end{cases} \\
V &= MAX \\
Gdzie: \quad & H \in [0, 360] \\
& S, V, R, G, B \in [0, 1]
\end{aligned}
\tag{2.2}
$$

Przestrzeń YCbCr

Podobnie jak w przestrzeni HSV, w przestrzeni YCbCr wydzielana jest luminancja koloru Y i zapisywana w jednej ze składowych. Pozostałe dwie składowe Cb i Cr mówią o chromacji koloru. Odpowiednio Cb niesie informację o składowej niebieskiej, a Cr o składowej czerwonej.

Do przekształcenia przestrzeni RGB na przestrzeń YCbCr używa się poniższego wzoru [39]:

$$
\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.2568 & 0.5041 & 0.0980 \\ -0.1482 & -0.2910 & 0.4392 \\ 0.4392 & -0.3678 & -0.0714 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \tag{2.3}
$$

Gdy składowe RGB przyjmują wartości 0-255, wartość Y po przekształceniu na przestrzeń YCbCr będzie z zakresu 0-255, a składowe Cb, Cr 16-240.

Przestrzeń YCbCr jest powszechnie używane w wielu formatach wideo np.: H.26L. H.264, MPEG-4 [39]. Używa się jej również z dużym powodzeniem w systemach rozpoznawania skóry [33].

2.2 Przetwarzanie

Przetwarzanie obrazu to operacja wykonywana na obrazie, której wynikiem jest inny obraz. Często pojęcie to jest mylone z analizą obrazu, która z kolei polega na zamienieniu obrazu na jego opis w postaci cech charakterystycznych.

W rozpoznawaniu obrazów przetwarzanie ma na celu poprawienie jakości, podkreśleniu istotnych dla danego zagadnienia cech obrazka, wyostrzenie krawędzi, usunięcie szumu i wiele innych operacji, które mają uczynić obraz łatwiejszym w segmentacji.

Krok ten wymaga z reguły największej ilości zasobów obliczeniowych, przez co jest najbardziej pracochłonny dla komputera. Mimo to, nie jest on wymagający algorytmicznie i ideologicznie, ponieważ w przeważającej ilości systemów przetwarzanie polega na tym samym i jest implementowane w bardzo podobny sposób.

Przekształcenia można podzielić na następujące grupy:

- przekształcenia geometryczne – czyli wszelkiego typu obroty, skalowania, odbicia, pochylenia;
- operacje punktowe – lokalne operacje zmieniające wartość piksela badając informację zawartą jedynie w danym pikselu;
- przekształcenia kontekstowe – operacje globalne zależne od całego obrazka lub jego części;
- przekształcenia widmowe – operacje wykorzystujące transformatę Fouriera i operujące na widmie obrazka;
- przekształcenia morfologiczne – przekształcenia warunkowe, czyli wykonywane, gdy spełniany jest warunek logiczny. Mogą być wykonywane iteracyjnie, aż do spełnienia zadanego warunku.

W kolejnych podrozdziałach zostały omówione kolejno powyższe przekształcenia. Podział ten jest umowny i nie jest on jedynym poprawnym podziałem metod przetwarzania obrazu.

2.2.1 Przekształcenia geometryczne

Pierwsza grupa przekształceń to te, które mają za zadanie poprawić, bądź zmienić geometrię obrazu. Są to wszelkiego typu skalowania, obroty, rozciągnięcia, pochylenia itp. Służą przede wszystkim do ujednolicenia geometrii różnych obrazów, oraz zniwelowania błędów powstałych podczas akwizycji. Dla przykładu obraz, w którym linia horyzontu nie jest idealnie pozioma, należy obrócić o pewien kąt, czyli zmienić położenie początkowe pikseli obrazka zmieniając ich współrzędne według funkcji obrotu.

Zniekształcenia geometryczne powstałe w wyniku użytkowania niskiej jakości optyki w aparatach, bądź kamerach podczas akwizycji są zależne od użytego sprzętu. Jeżeli używa się stale tego samego urządzenia do akwizycji zniekształcenia takie można w łatwy sposób, z dobrym wynikiem zniwelować używając przekształceń geometrycznych. Zniekształcenia te występują również podczas używania kamer szerokokątnych. Będą to najczęściej zniekształcenia poduszkowate i beczkowate (rysunek 2.5).

Rysunek 2.5: Zniekształcenie poduszkowate

Przekształcenia geometryczne są stosowane zarówno do redukcji zniekształceń wynikłych podczas akwizycji, jak również jako pomocnicze funkcje przy analizie i wydzielaniu cech.

W pracy używano standardowej kamery internetowej niewprowadzającej znaczących zniekształceń geometrycznych, przez co użycie przekształceń geometrycznych nie było potrzebne. W przypadku użycia kamer innego typu np.: kamer szerokokątnych konieczne byłoby zaimplementowanie jako pierwszej operacji przekształcenia geometrycznego poprawiającego geometrię obrazu.

2.2.2 Operacje punktowe

Operacje, które przekształcają piksele obrazka niezależnie od innych pikseli, a biorące pod uwagę jedynie wartość przekształcanego piksela, nazywane są operacjami punktowymi. Są to przekształcenia, które w znaczący sposób mogą zmienić reprezentację obrazka, zarówno poprawiając jego jakość i uwidaczniając interesujące cechy, jak i całkowicie tracąc informację w nim zawartą.

Operacje punktowe, z racji swojego charakteru, pozwalają na bardzo szybkie ich wykonanie. Złożoność algorytmów realizujących przekształcenia punktowe jest liniowa (O(n)). Dodatkowo możliwe jest wykonywanie operacji punktowych równolegle dla wielu pikseli, co przy wielordzeniowości dzisiejszych procesorów graficznych przynosi znaczący zysk na czasie wykonania.

Przykładowe operacje punktowe to rozjaśnianie, utworzenie negatywu czy przyciemnianie. Inne bardziej zaawansowane to np.: selektywna zmiana kolorów, rozjaśnianie poszczególnych kanałów, zaciemnianie kolorów części obrazka itp.

Przekształcenia punktowe zostały użyte w systemie do wyrównania jasności obrazka tła. Więcej na ten temat można znaleźć w podrozdziale 4.4.3.

Look up table (LUT)

Większość operacji punktowych jest wykonywana za pomocą tablicy LUT (od ang. Lookup table). Tablica ta służy jako mapa pikseli wejściowych na piksele docelowe i przygotowuje się ją przed przetwarzaniem obrazu, obliczając często bardzo złożone funkcje. Jednokrotnie przygotowana tablica może posłużyć do tego samego typu przekształcenia wielokrotnie, bez potrzeby ponownego obliczania funkcji przekształcenia.

2.2.3 Przekształcenia kontekstowe

Operacje, dla których danymi wejściowymi są nie tylko pojedyncze piksele, ale pewne zbiory pikseli, nazywane są przekształceniami kontekstowymi, lub przekształceniami przy pomocy filtrów. Końcową wartość piksela oblicza się na podstawie jego poprzedniej wartości oraz informacji o sąsiednich pikselach.

Filtry z racji wymogu przeliczania dużej ilości informacji dla każdego pojedynczego piksela, są złożone obliczeniowo. Wymagają przez to znacznie więcej zasobów niż operacje punktowe omówione w poprzednim podrozdziale. Przy badaniu dużego obszaru piksela wejściowego spowolnienie może być nawet kilkudziesięciokrotne. Jednak i te operacje, podobnie jak punktowe, mogą być wykonywane niezależnie od wyników dla kolejnych pikseli, co umożliwia przyspieszenie ich wykonywania na procesorach wielordzeniowych.

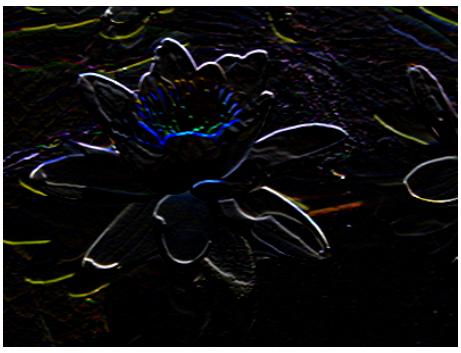
Przekształcenia kontekstowe w znaczący sposób wpływają na przetwarzany obraz, przekształcając go również geometrycznie. Dzięki swojej naturze mogą usuwać szum z obrazka, jak również podkreślać krawędzie, lub inne słabiej widoczne informacje zawarte w obrazie. W literaturze operacje kontekstowe są omówione bardzo szeroko ze względu na ich elastyczność i możliwość dopasowania do wielu zastosowań.

Większość operacji kontekstowych opiera się na użyciu maski filtru. Mogą to być zarówno maski na siatkach w kształcie heksagonalnym, lub częściej spotykanych siatkach kwadratowych.

Maska jest najczęściej kwadratem zawierającym wagi pikseli obszaru, który będzie rozpatrywany dla obliczenia wartości końcowej punktu obrazu. Typowy rozmiar maski to 3 na 3 (rysunek 2.2) z punktem relacyjnym na środku. Dla masek występuje jednak problem z pikselami znajdującymi się na krawędzi obrazka. Dla tych pikseli nie jest możliwe obliczenie wyjściowej wartości z racji niepełnej informacji wejściowej. Nie umniejsza to jednak popularności przekształceń kontekstowych, które są najchętniej i najczęściej używane w systemach rozpoznawania obrazów.

Używając masek można zamodelować operacje takie jak wyostrzanie, rozmazywanie (usuwanie szumu), detekcja krawędzi i wiele innych. Na rysunkach 2.3, 2.5, 2.4, 2.6 przedstawiono przykładowe maski dla kilku znanych operacji.

Rysunek 2.6: Obraz oryginalny



Rysunek 2.7: Obraz po zastosowaniu maski Prewitta

Rysunek 2.8: Obraz po rozmyciu

Rysunek 2.9: Obraz po zastosowaniu maski Robertsa

x	x	x
x	x	x
x	x	x

Tabela 2.2: Maska o rozmiarze 3 na 3. W miejsce x wstawia się wagę poszczególnych pikseli

1	1	1
1	1	1
1	1	1

Tabela 2.3: Maska rozmycia

-1	-1	-1
0	0	0
1	1	1

Tabela 2.4: Maska Prewitta (detekcja krawędzi)

1	2	1
2	4	2
1	2	1

Tabela 2.5: Maska rozmycia - Gaussowska (lepszy efekt)

0	0	0
0	0	-1
0	0	1

Tabela 2.6: Maska Robertsa (detekcja krawędzi)

Dla masek pokazanych na rysunkach 2.3 - 2.6 należy po wymnożeniu pikseli przez wagi znormalizować ich wartość by uzyskać obraz wyjściowy o podobnej jasności co wejściowy. Czyni się to poprzez podzielenie wartości końcowej poprzez sumę wag maski (w przypadku gdy jest ona różna od 0). Przykładowe zastosowanie masek można zaobserwować na obrazach 2.6 - 2.9.

Początkowo w implementowanym systemie używano masek filtracji dolnoprzepustowej w celu usunięcia szumów wysokoczęstotliwościowych. W trakcie rozwoju programu zdecydowano się jednak na użycie jedynie operacji morfologicznych ze względu na wydajność działania aplikacji.

2.2.4 Przekształcenia widmowe

Innym typem przekształceń obrazu są przekształcenia operujące na jego widmie otrzymanym po użyciu Transformaty Fouriera. Bada się w ten sposób informację zawartą w całym obrazie, a nie jak poprzednio w wydzielonych jego częściach.

Przekształcenia widmowe niosą jednak ze sobą duży problem, jakim jest obliczenie widma obrazka. Obecnie kalkulacja transformaty Fouriera staje się jednak coraz łatwiejsza, jako że jest ona często implementowana sprzętowo, szczególnie w procesorach

graficznych. Stosuje się również uproszczone transformaty, jednak obarczone są one dużą stratą informacji. Same przekształcenia, po otrzymaniu współrzędnych widma wykonywane są szybko i nie wymagają dużej ilości zasobów komputera. Po przekształceniu widma następuje powrót do postaci obrazka używając odwrotnej transformaty Fouriera.

Mimo zaawansowanej matematycznie metody przetwarzania obrazu, przekształcenia widmowe są używane rzadziej niż kontekstowe. Przyczyną nie jest sama złożoność algorytmiczna, jednak fakt, iż wyniki otrzymywane tą metodą nie są współmiernie lepsze do kosztów, jakie należy ponieść w celu ich uzyskania. Często zdarza się, że metody filtrowe dają nawet lepsze rezultaty niż przekształcenia widmowe. Z drugiej jednak strony, niektórych wyników nie da się osiągnąć inaczej niż używając metod widmowych.

W trakcie implementacji systemu nie było potrzeby używania przekształceń widmowych.

2.2.5 Przekształcenia morfologiczne

Kolejny typ przekształceń to przekształcenia morfologiczne. Są one podobne do przekształceń za pomocą filtrów, tzn. też operują na obszarze obrazka, a nie na pojedynczych pikselach, jednak różni je warunkowość wykonania operacji. Każde przekształcenie morfologiczne wykonywane jest na pikselach, których otoczenie spełniają zadany warunek.

Ten typ przekształceń należy do najbardziej skutecznych metod przekształcania obrazów. Nie jest to jednak osiągane przez pojedyncze przekształcenia morfologiczne. Dopiero gdy są one zgrupowane i wykonane kolejno, skutkują wyodrębnieniem oraz podkreśleniem najistotniejszych, z punktu widzenia rozpoznawania obrazów, informacji.

Największą wadą przekształceń morfologicznych jest ich duża złożoność obliczeniowa. Podczas wykonywania programu w związku z tą pracą operacje morfologiczne okazały się najbardziej zasobochłonne ze wszystkich części algorytmu. Operacja zamknięcia, złożona z operacji dylatacji i erozji szacunkowo zajmuje kilkukrotnie więcej czasu od wszystkich pozostałych części algorytmu rozpoznawania gestów ręki. Pomimo powyższej wady, operacja zamknięcia pozwala na uzyskanie znacznie lepszych wyników rozpoznawania i jest często implementowana w podobnego typu systemach.

Operacje morfologiczne operują na maskach podobnych jak te przedstawione w trakcie omawiania przekształceń kontekstowych. Są one wykorzystywane jednak w trochę inny sposób. Podstawową operacją przekształceń morfologicznych jest sprawdzanie czy piksele z sąsiedztwa spełniają warunek zapisany w masce. Jeżeli warunek ten jest spełniony wykonywane jest przekształcenie aktualnie badanego piksela. Przekształcenie to może być proste tj. rozjaśnienie bądź ściemnienie, jak również inne, bardziej złożone. Przekształcenia takie mogą być przetwarzane iteracyjnie, aż do spełnienia jakiegoś wa-

Rysunek 2.10: Obraz przed erozją

Rysunek 2.11: Obraz po erozji

runku, np. maksymalnej ilości iteracji.

Operacje morfologiczne są używane w zaimplementowanym systemie i mają duże znaczenie w poprawie jakości działania programu. W pierwszej kolejności w celu usunięcia wysokoczęstotliwościowych zakłóceń wnoszonych podczas akwizycji obrazu. W tym celu używana jest operacja otwarcia. Następnie już po wykryciu obszaru dłoni następuje korekcja obszarów zaklasyfikowanych jako skóra. Dzieje się to poprzez wykonanie kolejno erozji (usunięcie pojedynczych źle zaklasyfikowanych pikseli), dylatacji (połączenie mniejszych obszarów skóry oraz wygładzenie krawędzi obszarów) i erozji (zmniejszenie obszarów do pierwotnych rozmiarów). Kolejne operacje są wykonywane z różną ilością iteracji, odpowiednio 1, 3, 2 iteracje.

Operacje morfologiczne dla większych rozdzielczości muszą być wykonywane z większą ilością iteracji w celu osiągnięcia podobnych rezultatów końcowych. Dla przykładu w wyższej rozdzielczości działania implementowanego programu (640 na 480 pikseli) ustawieniem optymalnym do powyższej operacji korekcji obszarów skóry było odpowiednio 2, 7, 5 iteracji.

Poniżej omówiono podstawowe operacje morfologiczne, używane często w systemach rozpoznawania obrazów, w tym również w programie stworzonym wraz z tą pracą.

Erozja

Erozja jest podstawowym przekształceniem morfologicznym. Operacja ta służy do usuwania zniekształceń w postaci rys i zakłóceń mogących wprowadzić błędną informację do dalej analizowanego obrazu. Czasami operacja ta nosi miano minimum, czyli przepisania piksela z maski o minimalnej wartości jasności.

Erozja składa się ze sprawdzenia czy wszystkie piksele objęte maską spełniają zadany warunek, najczęściej są powyżej progowej jasności. W przypadku gdy warunek ten jest spełniony, to element centralny maski jest niezmieniony w obrazie wejściowym, w przeciwnym przypadku piksel ten nie jest przepisywany do obrazu wyjściowego (jest zastępowany kolorem tła).

Erozja jest pokazana na obrazach 2.10,2.11.

Rysunek 2.12: Obraz przed dylatacją

Rysunek 2.13: Obraz po dylatacji

Dylatacja

Dylatacja to również jedno z podstawowych przekształceń morfologicznych. Rozszerza ona obszary, zapełniając wycięcia i ubytki z elementów obszarów. Zastosowana na jednolitym kształcie zwiększa jego rozmiar. Czasami operacja ta nosi miano przekształcenia maksimum, czyli przepisania piksela z maski o maksymalnej wartości jasności.

Dylatacja składa się ze sprawdzenia czy którykolwiek z pikseli objętych maską spełnia warunek (najczęściej czy jest powyżej progowej wartości). W przypadku gdy warunek ten jest spełniony, to element centralny maski jest niezmieniony w obrazie wejściowym, w przeciwnym przypadku piksel ten nie jest przepisywany do obrazu wyjściowego.

Dylatacja jest pokazana na obrazach 2.12, 2.13.

Otwarcie

Operacja otwarcia jest złożeniem podstawowych operacji morfologicznych. W pierwszej kolejności na obrazku jest wykonywana erozja, a następnie dylatacja. Ma to na celu usunięcie błędnych połączeń obiektów nie zmniejszając jednocześnie ich rozmiarów.

Zamknięcie

Zamknięcie, podobnie jak operacja otwarcia, jest złożeniem operacji dylatacji i erozji. Różnica polega na kolejności wykonywania tych przekształceń. W pierwszej fazie wykonywana jest dylatacja, a następnie erozja. Celem zamknięcia jest wypełnienie luk powstałych na obrazku, zamalowaniu dziur wewnątrz obiektu itp., nie zmieniając przy tym rozmiarów obiektów.

2.3 Segmentacja

Analiza obrazu to proces, w którym z przetworzonego wstępnie obrazu wydziela się informacje w nim zawartą. Informacja ta ma postać danych i nie jest koniecznie wizualna. Mogą być to liczby mówiące o kształcie obiektów w obrazie, jak również dane dotyczące

jasności i inne. Podczas analizy obrazu uzyskuje się informację zależną od implementowanego systemu rozpoznawania obrazów. Może być to decyzja dotycząca rozpoznania, jak również zestaw szczegółów rozpoznawanego obiektu.

Proces analizy obrazu rozpoczyna się od segmentacji. Jest to czynność, w której obraz dzieli się na obszary jednolite pod względem pewnych cech, najczęściej zawierające szukane obiekty. Dla przykładu, w trakcie etapu segmentacji w systemie implementowanym wraz z tą pracą, wydzielane są potencjalne obszary dłoni.

Obszary są segmentowane na podstawie cech, które można odczytać z obrazu. Są to kolor pikseli, częstotliwość zmian jasności, krawędzie i inne. Często w trakcie segmentacji łączy się kilka metod dla uzyskania lepszego efektu końcowego i poprawniejszej klasyfikacji.

Metody segmentacji obrazów można podzielić na trzy główne grupy. Są to:

- metody punktowe,
- metody krawędziowe,
- metody obszarowe.

Powyższy podział nie jest jedynym poprawnym, ma jednak na celu uporządkowanie wiedzy dotyczącej segmentacji obrazów. Znane są również metody hybrydowe łączące kilka z powyższych. Poniżej opisane zostały kolejno grupy metod segmentacji.

2.3.1 Metody punktowe

Segmentacja punktowa jest podobna do operacji przekształceń morfologicznych na pojedynczych punktach. Dla potrzeb segmentacji definiuje się warunki, które muszą zostać spełnione by punkt przydzielić do danej grupy (obiektu). Operacje tą wykonuje się niezależnie dla każdego piksela badając informację w nim zawartą, zazwyczaj kolor, ale również inne dodatkowe dane w nich umieszczone przez inne algorytmy np. algorytmy grupowania.

Podstawowym typem segmentacji punktowej jest progowanie, w wyniku którego powstaje obraz binarny. Dzięki progowaniu można oddzielić piksele potencjalnie należące do szukanych obiektów od pikseli tła.

Operacja segmentacji punktowej jest obszernie opisana na przykładzie aplikacji rozpoznającej gesty dłoni w rozdziale 4.4.2.

2.3.2 Metody krawędziowe

Kolejną grupę algorytmów segmentacji tworzą metody krawędziowe. Operują one na informacji o krawędziach wyszukanych innymi pomocniczymi algorytmami. Algorytmy

z tej grupy nie wykrywają bezpośrednio obszarów obiektów, lecz poprzez detekcje krawędzi pomiędzy obiektami (najczęściej pomiędzy obiektem a tłem).

Metody krawędziowe dobrze nadają się do wykrywania obiektów na niejednolitym tle o charakterystyce łagodnych przejść. Mogą posłużyć np. do wykrywania samolotu na tle nieba, łodzi na tle morza. Nie nadają się jednak do obrazów o dużej ilości zakłóceń wysokoczęstotliwościowych oraz obrazów o małym kontraście.

W pracy rozważono możliwość użycia krawędzi do korekcji kształtu wykrytej dłoni. Zdecydowano się jednak na nie korzystanie z metod krawędziowych ze względu na liczne błędy w działaniu przy niejednolitym tle. Metoda krawędziowa w rozpoznawaniu gestów została użyta i opisana w artykule [37].

2.3.3 Metody obszarowe

Ostatnią grupą segmentacji obszarów z obrazu, jednak nie mniej istotną, są metody obszarowe. Wykorzystują one informacje o sąsiednich pikselach i na tej podstawie tworzone są grupy pikseli.

Metody obszarowe nie zostały użyte w programie, a ich opis ma na celu skompletowanie wiedzy o metodach segmentacji. Rozróżniane jest kilka typów metod obszarowych, opisanych poniżej.

Rozrost obszarów

Pierwszym omawianym typem metod obszarowych jest rozrost obszarów. Metodę tą rozpoczyna wybór punktu z obrazka. Może być on wybierany na podstawie informacji z poprzedniej klatki sekwencji (jeżeli taka jest dostępna), jak również na podstawie innych algorytmów, czy nawet losowo. Kolejne punkty są dobierane z najbliższego sąsiedztwa wybranego punktu na podstawie właściwości takich jak jasność czy kolor piksela. Jeżeli kolejne piksele zostaną zaklasyfikowane jako należące do tego samego obiektu co wcześniejszy piksel, to jest on dołączany do grupy i następuje kolejny rozrost obszaru.

Bardziej zaawansowane algorytmy rozrostu obszarów rozpoczynają nie od pojedynczego punktu, lecz od zbioru punktów i próbują dopasować do tego zbioru kolejne piksele. Ponadto szukanych obszarów może być kilka, więc operacje rozrostów powtarza się w takim przypadku dla niesklasyfikowanych jeszcze pikseli w celu uzyskania pożądanego efektu segmentacji.

Podział obszarów

Segmentacja przez podział obszarów polega na stopniowym podziale obrazka na mniejsze części. Każda część obrazka jest dzielona w przypadku, gdy nie jest jednolita pod

względem pewnych cech (kolor, jasność itp.). Podział może być dokonywany na stałą liczbę podobszarów, lub też zmienną w zależności od zawartości dzielonego obszaru.

Metoda podziału i łączenia

W metodzie podziału i łączenia występują kolejne podziały i połączenia w celu uzyskania jak najdokładniejszego dopasowania obszarów. W przypadku, gdy obszar jest niejednolity następuje jego podział, a następnie podzielone obszary łączone są w jednolite według różnicujących cech.

Segmentacja wododziałowa

Metoda wododziałowa opiera się na znajdowaniu lokalnych minimów na podstawie jasności pikseli. Jasność może być interpretowana jako wysokość podłoża, a znajdowanie przyległych obszarów polega na „zalewaniu" obszarów do momentu znalezienia najlepszego dopasowania. W ogólności jasność pikseli może być zamieniona na inny parametr zmienny np. nasycenie poszczególnej barwy, stosunek dwóch składowych koloru, barwa piksela i podobne.

2.4 Wydzielanie cech

Wydzielanie cech to kolejny etap rozpoznawania obrazu. Jest on dokonywany po wstępnym przetwarzaniu obrazu i segmentacji. Etap ten, to proces poszukiwania cech, które są charakterystyczne dla danych klas. Informacje o cechach mogą być wydzielane na kilka różnych sposobów.

W trakcie tego etapu dane zawarte w obrazie ulegają znacznemu zmniejszeniu. Z kilkudziesięciu kilobajtów danych dotyczących koloru, jasności poszczególnych pikseli wydziela się kilka specyficznych cech opisujących obraz. Ma to na celu wydzielenie informacji, która jest istotna z punktu widzenia rozpoznawania obrazów. W związku z powyższym etap ten można traktować jako swoista redukcja wymiarowości.

W dobrze zaprojektowanym systemie rozpoznawania obrazów różne obrazy przedstawiające te same obiekty będą, po etapie wstępnego przetwarzania i wydzielania cech, reprezentowane przez podobne dane. Wniosek z tego, że w celu zapewnienia dobrej jakości rozpoznawania obrazów przez projektowany system należy znaleźć takie cechy, które będą niezmienne, lub zmienne w małym zakresie dla danego rozwiązania.

Poniżej przedstawiono różne metody wydzielania cech.

2.4.1 Niskopoziomowe

Do niskopoziomowego wydzielania używa się algorytmów operujących na pikselach bezpośrednio. Są to algorytmy wykrywania krawędzi, wierzchołków, obszarów, ale również algorytm taki jak skaloniezmiennicze przekształcenie cech. Rezultatem są informacje jedynie o pewnych charakterystycznych miejscach obrazu, które mogą posłużyć do dalszej analizy. Np. dla różnokolorowych pudełek na taśmie produkcyjnej, jeżeli mamy rozpoznawać jedynie ich kształt, informacje o krawędziach są w zupełności wystarczające.

2.4.2 Badające krzywiznę

Algorytmy badające zmienność na obrazie to algorytmy badające krzywiznę. Mogą one wydzielać informację o gradiencie koloru, intensywności jak również wyszukiwać krawędzie. Do tego typu należy min. algorytm autokorelacji [38].

2.4.3 Oparte na kształcie

Kolejny typ algorytmów wydzielających cechy tworzą procedury opierające się na analizie kształtu wydzielonych obiektów.

Do tej grupy zaliczamy takie procedury jak dopasowanie kształtów, transformatę Hough'a, jak również obliczanie momentów geometrycznych obrazka.

Geometryczne momenty bezwładności

Do wyznaczenia cech z obiektu mogą posłużyć geometryczne momenty bezwładności, nazywane również momentami przestrzennymi, określające środek ciężkości figury oraz jej miary bezwładności. Obliczane są one na podstawie punktów zawartych w obiekcie wg następującego wzoru:

$$\begin{aligned} m_{pq} &= \sum_{i=1}^{M} \sum_{j=1}^{N} i^p j^q f(i,j) \\ gdzie f(i,j) &= \begin{cases} 1 & \text{gdy piksel należy do obiektu} \\ 0 & \text{w przeciwnym przypadku} \end{cases} \end{aligned} \tag{2.4}$$

Obliczone w ten sposób momenty bezwładności mogą powiedzieć nam o umiejscowieniu środka ciężkości obiektu oraz podstawowego jego kształtu. Jednak w rozpoznawaniu obrazów są one bardziej przydane po znormalizowaniu i obliczeniu na ich podstawie momentów centralnych oraz niezmienników momentowych [52].

Momenty centralne

Momenty bezwładności są wrażliwe na wszelkiego typu przekształcenia obiektu (obroty, skalowania), dlatego też, w czystej postaci, dla typowych zastosowań używa się ich tylko do wyznaczania położenia środka ciężkości. Na ich podstawie można obliczyć jednak bardziej niezależne momenty centralne (ozn. η).

Znormalizowane momenty centralne można obliczyć wg następującego wzoru [40]:

$$\begin{aligned} &\eta = \frac{U_{pq}}{U_{00}^r} \\ &gdzie \qquad r = \frac{p+q}{2} + 1 \\ &\qquad\qquad p + q = 2, 3, \ldots \end{aligned} \tag{2.5}$$

Momenty centralne

Momenty centralne są znormalizowane jeżeli chodzi o skalowanie, jednak w przypadku, gdy rozpoznawane obiekty mogą być obracane, warto skorzystać z niezmienników momentowych (ozn. ϕ).

Najpopularniej używanych momentów jest 7 i cechuje je niewrażliwość na przekształcenia obrotu i skalowania. W trakcie rozpoznawania obiektów mogą one znajdować się w różnej odległości od obserwatora, a dodatkowo mogą przyjmować różną orientację. Niezmienniki momentowe określą wtedy cechy tych obiektów identycznie, lub z dużym podobieństwem.

Oblicza się je na podstawie momentów centralnych według następujących wzorów [40]:

$$\phi_1 = \eta_{20} + \eta_{02} \tag{2.6}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{2.7}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{2.8}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{2.9}$$

$$\begin{aligned} \phi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + 3(\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{2.10}$$

$$\begin{aligned} \phi_6 = {} & (\eta_{20} - \eta_{02})[(\eta_{30} - \eta_{12})^2 - (\eta_{21} - \eta_{03})^2] \\ & + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \end{aligned} \tag{2.11}$$

$$\begin{aligned} \phi_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{2.12}$$

Niezmienniki momentowe zostały użyte w pracy do wyznaczania cech dłoni, na podstawie których następuje klasyfikacja gestu. Podczas testów okazało się, że najlepiej rozróżniające gesty są cztery pierwsze niezmienniki momentowe i to na ich podstawie

system klasyfikuje jaki gest jest aktualnie pokazywany. Więcej na ten temat można przeczytać w podrozdziale 4.3.

2.5 Wykrywanie ruchu

Oprócz analizy obrazów statycznych niektóre systemy rozpoznawania obrazów mogą wykrywać również ruch obiektów. W tym celu stworzonych zostało wiele algorytmów, opartych na różnych podejściach. W literaturze można znaleźć rozwiązania od bazujących na starej już metodzie obliczania różnic kolejnych ramek po zupełnie innowacyjne rozwiązania opierających się na bardziej wyrafinowanych algorytmach [7].

W celu rozpoznawania ruchu wymagana jest informacja o kilku następujących po sobie obrazach. Informacje taką można uzyskać w czasie rzeczywistym używając kamery lub też analizując wcześniej nagrany film wideo.

Liczba klatek na sekundę, we wszystkich rozwiązaniach, jest kluczowym elementem warunkującym wydajność algorytmu. Im większa liczba klatek na sekundę, tym algorytm musi działać szybciej by nadążyć z analizą w czasie rzeczywistym. Nie zawsze jednak wydajność jest w tego typu rozwiązaniach kluczowa. Czasami systemy detekcji ruchu nie muszą działać szybko, a dużo istotniejsza jest dokładność uzyskiwanych wyników. Takim przykładem może być badanie nagrań wideo po popełnieniu przestępstwa. Przykładowym zastosowaniem takiego systemu jest wyodrębnienie tylko tych sekwencji obrazów, w których występuje ruch, w celu odfiltrowania nieistotnych dla śledztwa statycznych obrazów.

Poniżej skupiono się na kilku podejściach wykrywania ruchu w sekwencji obrazów.

2.5.1 Analiza różnicowa kolejnych klatek

Analiza różnicowa to najbardziej intuicyjne i jednocześnie najbardziej podstawowe z możliwych implementacji detekcji ruchu. W najprostszej postaci algorytm jest w stanie wykryć ruch, ale bez dodatkowych zabiegów nie uzyska się dzięki niemu informacji o kierunku czy dynamice ruchu.

W podstawowej wersji algorytm opiera się na badaniu różnic poszczególnych pikseli dwóch następujących po sobie klatek sekwencji wideo. Procedura ta jest wykorzystywana przy założeniu, że tło jest stałe, a jedynymi poruszającymi się obiektami są te, które są aktualnie analizowane. Wielką zaletą rozwiązania jest jego prostota, zarówno implementacyjna jak i obliczeniowa, co czyni je odpowiednim w zastosowaniach działających w czasie rzeczywistym [7].

Różnice pomiędzy klatkami mogą być liczone na podstawie informacji o jasności pik-

seli, kolorze, a także intensywności i innych danych dotyczących pojedynczego piksela. Detekcja ruchu następuje dla każdego piksela osobno. Badana jest zmiana poszczególnych pikseli i porównywana z wartością progową. Użycie wartości progowej jest spowodowane zakłóceniami, które występują w zdigitalizowanym obrazie. Uzyskanie idealnie podobnych obrazów w rzeczywistości, nawet dla całkowicie statycznych obrazów, jest w typowych zastosowaniach niemożliwe. Zakłócenia są wnoszone zarówno przez minimalne zmiany w oświetleniu, jak również przez niedoskonałość przetworników analogowo-cyfrowych instalowanych w kamerach.

W przypadku gdy zmiana wartości piksela przekroczy zadany próg, na jego miejscu jest zapisana informacja o ruchu. Ostatecznym wynikiem analizy różnicowej kolejnych klatek będzie więc obraz binarny zawierający informację, które piksele zostały poruszone.

Badając więcej niż jedną klatkę można uzyskać informację o dynamice ruchu, jednak gdy wymagana jest duża dokładność rozwiązania, algorytm ten nie jest wystarczający. Dodatkowymi jego wadami są: niewykrywanie powolnych ruchów lub słabe działanie na obiektach małokontrastowych w stosunku do tła.

Ze względu na niską dokładność algorytmu różnicowego oraz wady wcześniej wspomniane okazał się on niewystarczający do zastosowania w systemie wykrywającym ruch dłoni.

2.5.2 Przepływ optyczny

Kolejną metodą wykrywania ruchu jest przepływ optyczny. Jest to również znana już technika, często używana w systemach rozpoznawania ruchu obiektów i opisywana w artykułach naukowych [7] [2] [46] [49] [4] [43].

Metoda przepływu optycznego bazuje na odnalezieniu nowego położenia dla punktów z poprzedniej klatki. Jej interpretacją jest więc pole wektorowe umożliwiające przekształcenie poprzedniego obrazu sekwencji wideo w obecny obraz. Innymi słowy jest to zbiór translacji przekształcających poprzednią klatkę sekwencji w klatkę obecną.

Idea ta zakłada, że warunki oświetlenia w dwóch kolejnych klatkach są relatywnie niezmienne. Jeżeli obiekt zmieni swoje położenie to w kolejnej klatce piksele mu odpowiadające zostaną przesunięte w pobliże pierwotnego położenia. W przypadku, gdy prędkość poruszania się obiektów nie będzie zbyt duża oraz liczba klatek sekwencji wideo na sekundę będzie wystarczająca, przesunięcie pikseli nie będzie duże i możliwe do wyznaczenia.

Istnieje wiele implementacji przepływu optycznego. Znane są metody gradientowe, czyli te bazujące na analizie pochodnych intensywności obrazów, metody częstotliwościowe oparte na filtrowaniu informacji o obrazie w dziedzinie częstotliwości oraz metody

korelacyjne bazujące na odpowiedniości obszarów obrazów.

W systemie użyto algorytmu rozpoznawania ruchu bazującego na obliczaniu przepływu optycznego metodą Lucasa Kanade [4]. Więcej informacji na temat detekcji ruchu znajduje się w podrozdziale 4.6.

2.6 Analiza cech obrazu

Po procesie wydzielania cech obiektów następuje ostatni etap rozpoznawania obrazów - analiza wyznaczonych cech. Proces ten kończy się wynikiem rozpoznawania, czyli najczęściej wyznaczeniem obszarów szukanych obiektów oraz czasami określeniem dodatkowych właściwości szukanych obiektów. Dla przykładu w systemie rozpoznawania gestów zostanie wyznaczony obszar dłoni oraz dodatkowa informacja o geście jaki jest wykonywany.

Analiza cech może być procesem trywialnym, jak również bardzo zaawansowanym. Liczba metod analizy cech jest bardzo duża, gdyż są do tego celu używane algorytmy z dziedziny uczenia maszynowego i sztucznej inteligencji. Poniżej zostały przedstawione tylko niektóre algorytmy analizy cech obrazu. Wszystkie z tych metod posłużyły w systemie do określenia wykonywanego gestu.

2.6.1 Metoda Bayesa

Jedną z najprostszych implementacji analizy cech obrazu jest naiwny klasyfikator Bayesa opierający się na twierdzeniu Bayesa. Jest on szczególnie odpowiedni jeżeli problem, który jest rozwiązany cechuje się dużą ilością wymiarów. Mimo swojego prostego charakteru dla wielu przypadków rozpoznawania obrazów daje lepsze wyniki aniżeli inne bardziej skomplikowane algorytmy [38].

Koncepcja metody Bayesa polega na obliczaniu prawdopodobieństwa z jakim klasyfikowany obiekt może należeć do danej klasy. Wyróżnia się prawdopodobieństwo *a priori*, czyli znane odgórnie, obliczone ze zbioru uczącego oraz prawdopodobieństwo *a posteriori*, czyli prawdopodobieństwo, że dany obiekt należy do odpowiedniej klasy. Prawdopodobieństwo *a posteriori* jest liczone na podstawie prawdopodobieństwa *a priori* oraz aktualnych danych o badanym obiekcie.

Dla ustalenia uwagi weźmy pod uwagę problem rozpoznawania dwóch gestów dłoni. Prawdopodobieństwo *a priori* w tym przypadku to szansa, z jaką dany gest może być pokazany. W zbiorze uczącym powinno znaleźć się więcej gestów, które mają większą szansę na pojawienie się w analizowanym obrazie.

Dla powyższego przypadku prawdopodobieństwo *a posteriori* będzie obliczane jako iloczyn prawdopodobieństwa *a priori* oraz informacji na temat najbliższych obiektów ze

zbioru uczącego do badanego obiektu. Przeważnie, dla uzyskania tej informacji, obliczany jest stosunek ilości wystąpień poszczególnych klas w zadanym otoczeniu badanego obiektu.

Ostateczny wynik, czyli ustalenie do której klasy należy badany obiekt uzyskuje się poprzez wybór tej klasy, dla której obliczone prawdopodobieństwo *a posteriori* jest największe.

2.6.2 Metoda k - najbliższych sąsiadów

Inną metodą wyznaczania klasy z wydzielonych cech jest metoda k - najbliższych sąsiadów (*ang. K-nearest nighbour*). Polega ona na znalezieniu ze zbioru uczącego najlepiej dopasowanych próbek i określeniu klasy na ich podstawie.

Najczęściej używa się klasyfikatorów o nieparzystym k czyli 1-NN (*od ang. 1-nearest nighbour*), 3-NN, 7-NN. Nieparzysta liczba najbliższych sąsiadów pozwala na wybór klasy przez proste głosowanie większościowe. W przypadku, gdy mamy do czynienia z najprostszą implementacją algorytmu 1-NN szukana jest próbka, która jest najbliżej badanej, a klasa wynikowa jest taka jak znalezionej próbki.

W celu ustalenia odległości próbek ze zbioru uczącego do testowanego obiektu liczona jest najczęściej odległość euklidesowa. Inne odległości jakie mogą być użyte w tym algorytmie, to np. odległość Hamminga, Mahalanobisa.

2.6.3 Maszyna wektorów nośnych

Kolejnym klasyfikatorem jest maszyna wektorów nośnych. Główna jego idea opiera się na wyznaczeniu hiperpłaszczyzny rozdzielającej badane klasy. Podczas, gdy inne koncepcje klasyfikatorów, w tym omówione już klasyfikatory k - najbliższych sąsiadów i klasyfikator Bayes'a, opierają się na minimalizacji błędu klasyfikacji, maszyna wektorów nośnych działa na zasadzie minimalizacji ryzyka poprzez maksymalizację marginesu rozdzielającego dwie klasy [33].

2.6.4 Odległość Mahalanobisa

Czasem, problemem jest ustalenie, czy obiekt należy do klas zadanych obiektów, czy też nie. Tradycyjne algorytmy analizy cech obrazu wymagałyby dla takiego przypadku zbioru treningowego dla szukanych klas oraz próbek ze wszystkich możliwych innych obiektów. W ogólności zapewnienie takiego zbioru jest niemożliwe. Dla takiego przypadku dobrym rozwiązaniem jest policzenie odległości Mahalanobisa pomiędzy kształtem badanego obiektu a obiektu wzorcowego [48] [38].

Odległość Mahalanobisa jest liczona wg wzoru [8]:

$$d(v_1, v_2) = \sqrt{\sum_{i,j} cov(i,j)'(v_1(i) - v_2(i))(v_1(j) - v_2(j))} \tag{2.13}$$

gdzie $cov(i,j)'$ jest odwóconą macieżą kowariancji

Dla obliczonej odległości Mahalanobisa stosuje się następnie progowanie, czyli ustalanie czy odległość obiektu od wzorca jest mniejższa niż progowa wartość i należy do szukanej klasy.

W niniejszej pracy odległość Mahalanobisa została wykorzystana przy ustaleniu czy wyszukany obiekt jest dłonią. Dla uzyskania odpowiedzi liczona jest odległość pomiędzy średnią wartością niezmienników momentowych ze zbioru uczącego a badanym gestem. Odległość Mahalanobisa jest liczona pomiędzy obecnym gestem a średnimi wszystkich gestów klasyfikowanych przez system. Jeżeli najmniejsza z tych odległości jest mniejsza od ustalonego w trakcie badań progu, wydzielony obszar jest klasyfikowany jako dłoń i badany następnie jest gest za pomocą pozostałych algorytmów analizy cech.

Rozdział 3

Współczesne interfejsy użytkownika

Porozumiewanie się z maszyną od początku powstawania komputerów było zagadnieniem wartym uwagi. Początki były trudne i sięgają jeszcze okresu, w którym popularne były karty perforowane. Wynalezienie klawiatury i ekranu było przełomowe. Są one obecne i istnieją z dużym powodzeniem do dziś. Od czasu systemów okienkowych popularna stała się również mysz komputerowa. Jednak od tamtego czasu mało się zmieniło jeżeli chodzi o urządzenia wejściowe. W dalszym ciągu w komputerach domowych używamy klawiatur i myszek. Ostatnio popularnym jednak stał się również ekran dotykowy, w szczególności w urządzeniach mobilnych.

W latach 50 i 60 XX wieku praktycznie nie istniało pojęcie interakcji pomiędzy komputerem a człowiekiem. Skomplikowane algorytmy, wymagające pomocy obliczeniowej komputera, wykonywano w trybie wsadowym lub manualnie poprzez fizyczne przełączanie kontrolek i przycisków. Odbywało się to jednak jedynie przez ludzi ściśle związanych z informatyką. Programowanie odbywało się w dużej mierze „na kartce", a program wynikowy był sekwencją czynności jakie należy wykonać, by wczesne komputery potrafiły rozpoznać intencje człowieka.

Klawiatura została wynaleziona w latach 70 i spowodowała zdecydowane ułatwienie komunikacji z komputerem. Stał się on dzięki temu bardziej przyjazny człowiekowi. Wynalazek ten powoduje, że interakcja odbywa się tekstowo w czasie rzeczywistym. Osiągnięcie to było kamieniem milowy w rozwoju komputerów.

Następne wielkie wydarzenie nastąpiło gdy inżynierowie zastąpili popularny wtedy terminal, graficznym interfejsem użytkownika. Umożliwiło to rozszerzenie funkcjonalności interfejsu pozwalającego jedynie na wykonywanie poleceń tekstowych. Tak w latach 80 narodziła się koncepcja pulpitu opartego na oknach, ikonach, skrótach i menu znana w dzisiejszych popularnych systemach operacyjnych jak np.: w Windows czy środowisku graficznym Linux GNOME. Paradygmat ten został nazwany WIMP od angielskiego *Win-*

dows, Icons, Menus and Pointing devices, co znaczy okna, ikony, menu i urządzenia do wskazywania. Wraz z nim do klawiatury dołączyła mysz komputerowa, która znacząco ułatwiła obsługę graficznego interfejsu.

Kolejne 20 lat informatyki minęło pod wpływem schematu WIMP. Na początku dwudziestego pierwszego wieku człowiek zaczął się jednak starać, by komputer był jak najbardziej przyjazny człowiekowi, a komunikacja z nim odbywała się możliwie podobnie do komunikacji z drugim człowiekiem. Choć przez długi okres czasu rozpoznawanie gestów i interfejsy oparte na wizji były uważane jako możliwe w zastosowaniu tylko w wyspecjalizowanych rozwiązaniach, to uważa się, że niedługo tendencja ta odmieni się diametralnie. Główną przesłanką jest fakt, że coraz więcej deweloperów poświęca uwagę na tego typu rozwiązania. Między innymi zainteresowane rośnie u gigantów na rynku IT takich jak: Microsoft, IBM, Sony i Toshiba [43].

3.1 Podział ze względu na zastosowanie

Zastosowanie programów opartych na wizji można podzielić na trzy podstawowe części [49]:

- służące do nadzoru,
- służące do kontroli,
- służące do analizy.

Pierwszą gałęzią rozwoju rozpoznawania wizji jest nadzór ludzi i zbieranie informacji z otoczenia. Aplikacje tego typu pokrywają większość z klasycznych problemów związanych z automatycznym monitoringiem w obszarach o dużym zagęszczeniu ludzi. Są instalowane np.: na lotniskach, stacjach kolejowych czy w metro. Służą one do badania przepływu tłumu, obliczania rozkładu gęstości prawdopodobieństwa przepełnienia i zatamowania przepływu. Nowsze aplikacje tego typu są używane również w celach poprawienia bezpieczeństwa. Są w stanie rozpoznawać twarze ludzi i dopasowywać je do poszukiwanych przestępców, badać nietypowe zachowania zarówno całego tłumu jak również pojedynczych osób, czy wykrywać sytuacje niebezpieczne lub mogące stanowić niebezpieczeństwo w miejscach publicznych.

Inną gałęzią rozwoju aplikacji opierających się na percepcji komputera jest rozrywka i wszelkiego typu kontrolery. Mogą być to kontrolery gier, w których gracz używa kamery, w zamian za powszechnie znane interfejsy wejściowe, jak klawiatura, czy mysz komputerowa. Kamera może śledzić w takim przypadku ruch zarówno gałek ocznych,

głowy, dłoni czy nawet całego człowieka, bądź kilku ludzi jednocześnie. Często w programach graficznych dwuwymiarowe kontrolery (mysz, klawiatura) są niewystarczające. Również i w takich przypadkach przydaje się rozpoznawanie ruchu człowieka, w szczególności w trzech wymiarach. Rozwiązania te są używane także w sterowaniu robotów, zabawek oraz innych prostych urządzeń.

Ostatnią częścią aplikacji są te wykorzystujące analizę ruchu, na przykład człowieka. Powszechnie wykorzystywane są programy do badania ruchu sportowców, jak również w rehabilitacji. Bardzo znany i używany również na polskich uczelniach jest komercyjny system VICON [25]. Jest on rozwijany od wielu lat i daje bardzo dobre i rzetelne wyniki w medycynie np.: traumatologii, neurologii i reumatologii. Jest on używany również w generowaniu animacji dla potrzeb gier i filmów. W tej grupie aplikacji znajdują się również zastosowania z przemysłu motoryzacyjnego takie jak automatyczna kontrola poduszek powietrznych, wykrywanie snu i inne.

W 2004 roku Turk i Kolsch zaproponowali rodzaj urządzeń wejściowych zwanych Visual Based Interfaces (VBI), czyli urządzeń opierających się na obrazie widzianym przez kamery [27]. Prawdziwa popularność systemów wizyjnych zaczęła się jednak od sukcesu artykułu Moeslund et al., 2006 [49]. Z punktu widzenia interfejsu człowiek-komputer najistotniejszym jest badanie ruchu człowieka i interpretacja go, jako zdarzenia systemowego. W tym sensie podejścia wykorzystywane do rozpoznawania i analizy ruchu człowieka ogólnie można podzielić na trzy główne kategorie:

- analiza ruchu,
- oparte na wyglądzie,
- związane z modelem.

Oparte na analizie ruchu rozpoznają jedynie sam ruch nie badając struktury fizycznej. Nie dopasowują jednak rzeczywistych modelów człowieka a bazują jedynie na informacji uzyskanej z samego obrazu [49].

Oparte na wyglądzie, używają informacji takich jak: obrazy w skali szarości, krawędzie, sylwetki ciała. Na podstawie tych danych tworzona jest baza danych służąca do uczenia algorytmów dopasowania i weryfikacji informacji wydzielonej z obrazu [49].

Oparte na modelu to te, które skupiają się na dokładnym zamodelowaniu ciała człowieka, starając się odtworzyć rzeczywistość. Są to modele najczęściej przestrzenne i zawierają najwięcej informacji z trzech powyższych. Podejście to różni się od pozostałych dużą wiedzą *a priori*. Wiedza ta tworzy model, często bardzo dokładny i do tego modelu dopasowuje się rzeczywistość [27] [6]. Model może zawierać w sobie informacje o kształcie

w trzech wymiarach, dokładnym wyglądzie i kolorze badanych obiektów, strukturę ruchową (np. zakres i kierunek możliwych ruchów, dynamiczność, typ), oraz wiele innych informacji przydatnych podczas modelowania ruchu i rozpoznawania gestów.

3.2 Tendencje w rozwoju

Większość artykułów traktujących o interfejsach człowiek – komputer (*Human Computer Interaciton – HCI*) skupia się na ogólnych pojęciach dotyczących interfejsów (58%). O około połowę mniej (23%) jest na temat ogólnego rozwoju i zagadnień związanych z implementacją oraz wdrożeniem. 13% artykułów kładzie nacisk na sprawy związane z komputerem natomiast tylko 6% z artykułów omawai cechy charakterystyczne człowieka.

Główna gałąź rozwoju to zastosowania w komputerach mobilnych – około połowa (46%). Następnie (około 21%) to rynek sprzedaży, 17% opieka zdrowotna, 8% praca w terenie, oraz po 4% roszczenia ubezpieczeniowe i dziennikarstwo [31].

3.3 Istniejące rozwiązania

Na rynku powstaje coraz więcej projektów związanych z rozpoznawaniem wizji. Duże firmy informatyczne spostrzegły zainteresowanie użytkowników tego typu rozwiązaniami i zainwestowały duże środki finansowe w rozwój tej technologii. Poniżej przedstawiono dwa projekty skierowane na rozrywkę dwóch konkurujących firm Nintendo i Microsoft na rynku konsol do gier.

3.3.1 Wii

Wii jest konsolą produkowaną przez firmę Nintendo. Pojawiła się ona na rynku europejskim pod koniec 2006 roku. Komunikacja z komputerem odbywa się za pomocą specjalnego pilota na podczerwień. Firma udostępnia również rozpoznawanie sylwetki ciała, jednak cała komunikacja komputer-człowiek odbywa się przy pomocy dodatkowych urządzeń.

Przykładowym rozwiązaniem zastosowanym w konsoli Wii jest wykrywanie ruchu palców dłoni. W tym celu potrzebny jest czujnik Wiimote oraz tabliczka z diodami LED emitującymi promieniowanie podczerwone. Promienie te odbijają się od palców i są analizowane przez kontroler Wiimote. Urządzenie to jednak działa słabo bez dodatkowych tablic odblaskowych umieszczanych na końcach palców. Dopiero po wykonaniu tych wszystkich zabiegów otrzymujemy wykrywanie trajektorii ruchu palców z dość dobrą dokładnością.

Więcej informacji o projekcie można znaleźć na stronie Johnny'a Lee [10], autora projektów związanych z kontrolerem Wii lub oficjalnej stronie Nintendo [23].

3.3.2 Natal

Projekt Natal jest konkurencyjnym projektem dla Wii i jest rozwijany przez Firmę Microsoft dla konsoli Xbox 360. W przeciwieństwie do rozwiązania firmy Nintendo w projekcie tym nie są potrzebne żadne dodatkowe kontrolery. Było to główne założenie firmy z Redmond podczas tworzenia rozwiązania.

Czas wydania projektu jest ustalony na jesień 2010 roku, a więc tuż po skończeniu pisania tej pracy, dlatego też wszystkie informacje na temat projektu są spekulacjami. Wiadomo jest, że projekt Natal ma umożliwiać automatyczne rozpoznawanie głosu i gestów, ruchów wielu osób jednocześnie. Przewidywane zastosowanie konsoli to głównie rozrywka, ale projekt ma służyć również do komunikacji międzyludzkiej.

Rozdział 4

Projekt aplikacji

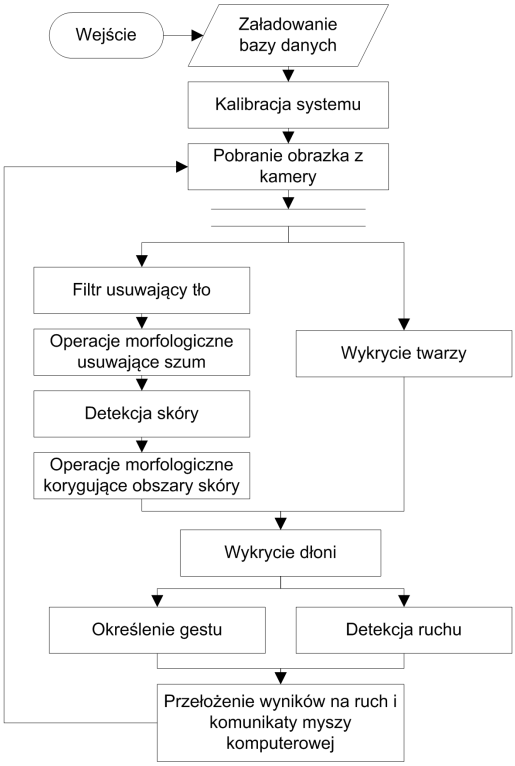
Jednym z celów pracy było stworzenie aplikacji służącej jako interfejs w komunikacji człowiek-komputer. By spełnić to założenie został napisany program w języku C++ z użyciem otwartej biblioteki programistycznej OpenCV (Dodatek B).

Tworzona aplikacja musiała zostać zaprojektowana z myślą o działaniu w czasie rzeczywistym, dlatego też zdecydowano się na użycie języka programowania kompilowanego do kodu binarnego. Użycie języków interpretowanych w trakcie wykonania, takie jak Java czy C# powodowałoby wolniejsze działanie aplikacji, przez co zmniejszyłoby wydajność działania systemu. Program jest napisany modułowo i jego ogólną konstrukcję przedstawiono na schemacie 4.1.

Aplikacja rozpoczyna się od załadowania bazy danych zawierającej informację o badanych gestach i przygotowuje klasyfikatory do późniejszego ich rozpoznawania. W tym etapie inicjowana jest również kamera internetowa zainstalowana w komputerze.

Główne działanie programu odbywa się w pętli. W każdym przebiegu następuje przetworzenie obrazka, analiza i podjęcie decyzji. W niezależnym wątku odbywa się wykrywanie obszaru twarzy służącego do eliminacji źle zaklasyfikowanych obszarów dłoni.

Poszczególne etapy działania programu zostały szczegółowo opisane w kolejnych podrozdziałach po części opisującej gesty rozpoznawane przez system. W podrozdziale 4.4 dotyczącym wykrywania dłoni znalazły się wszystkie etapy od filtru usuwającego tło, aż do wykrycia dłoni. W podrozdziale 4.5 został szczegółowo opisany etap określania gestu. Podrozdział 4.6 traktuje o etapie detekcji ruchu. Przełożenie wyników na ruch i komunikaty myszy komputerowej zostało zawarte w podrozdziale 4.7.



Rysunek 4.1: Schemat systemu rozpoznawania gestów i ruchu dłoni

4.1 Wymagania interfejsu człowiek – komputer

Pierwszym etapem powstawania programu jest określenie wymagań. Sformułowanie wymagań jest kluczowe dla powodzenia kolejnych kroków wytwarzania oprogramowania. Wymagania podzielono na funkcjonalne, oraz niefunkcjonalne i przedstawiono w osobnych podrozdziałach poniżej.

4.1.1 Wymagania funcjonalne

- Program powinien umożliwiać poruszanie kursora po ekranie. Zakres ruchu powinien obejmować cały obszar ekranu. Ruch powinien być płynny i zgodny z ruchem dłoni użytkownika.
- Program powinien poprawnie rozpoznawać co najmniej 4 gesty statyczne wykonywane dłonią. Informacje o wykonywanych gestach program ma pobierać z kamery internetowej.
- W programie powinno znaleźć się przełożenie wykonywanych gestów na komunikat lewego przycisku myszy. Powinna zostać zaimplementowana możliwość wykonania osobno naciśnięcia i puszczenia lewego klawisza myszki. Możliwa powinna być

również obsługa zdarzenia *przeciągnij i upuść*, czyli kombinacja: naciśnięcie lewego przycisku myszy, ruch kursora, a następnie puszczenia lewego kursora myszy.

- Program powinien umożliwiać przełożenie gestów na prawy i środkowy przycisk myszy. Wykonanie odpowiednich gestów ma skutkować wysłaniem do systemu zdarzeń kliknięcia i puszczenia odpowiedniego klawisza myszki.

4.1.2 Wymagania niefunkcjonalne

- Program powinien działać w czasie rzeczywistym.
- Program powinien działać zarówno w nocy jak i w dzień, przy odpowiednim oświetleniu.
- Program nie powinien blokować odbierania przez system komunikatów związanych z myszą komputerową.
- Program powinien być niezależny od typu kamery internetowej.

4.2 Podstawowe założenia

Podczas tworzenia aplikacji poczynione zostały założenia, które należy przestrzegać w celu poprawnej pracy systemu. Pierwszym z nich jest założenie, że program rozpoznaje gesty i ruch maksymalnie jednej dłoni.

Dla poprawnej detekcji ruchu należy zapewnić stałość tła podczas sterowania komputerem. System będzie działać dużo gorzej, jeżeli tło nie będzie statyczne, np. gdy kamera będzie zainstalowana w laptopie trzymanym na kolanach.

Dłoń powinna znajdować się bezpośrednio przed kamerą w odległości naturalnej dla siedzącego człowieka przed komputerem (około 50cm) podczas, gdy kamera powinna znajdować się na wysokości monitora. Gesty wykonywane muszą być tak, by wewnętrzna część dłoni zwrócona była w kierunku kamery. Podobne założenia czynione są w innych systemach rozpoznawania gestów [7].

Dla uzyskania jak najdokładniejszego pomiaru ruchu dłoni prędkość jej poruszania się nie powinna być zbyt duża w stosunku do ilości klatek osiąganych przez system. Ograniczenie to wynika z charakteru użytego algorytmu analizy ruchu poprzez badanie przepływu optycznego [46].

Pomiędzy śledzonym obiektem (dłonią) a kamerą nie powinny znajdować się żadne inne obiekty. Mogą one negatywnie wpłynąć na ciągłość ruchu i poprawność rozpoznawanych gestów.

W pracy, z racji braku wystarczającej ilości próbek oraz możliwości przetestowania poprawności działania programu skupiono się na kolorze skóry ludzi białych.

4.3 Gesty rozpoznawane przez program

W celu stworzenia interfejsu mającego naśladować myszkę komputerową potrzebne są co najmniej 2 gesty tzn. jeden odpowiadający nieprzyciskaniu żadnego przycisku myszki oraz jeden na przyciśnięcie. Myszki jednoprzyciskowe są jednak mało popularne i używane raczej tylko w systemach MacOs. Dlatego też w pracy zdecydowano się użyć 4 gestów, by móc zasymulować 3 przyciskową mysz, bądź też 2 przyciskową z możliwością przypisania jednego gestu do dodatkowej akcji np.: zamknij (zazwyczaj Alt + F4).

Ze względu na charakter tworzonej aplikacji, możliwe jest wykonanie tylko jednego gestu w tym samym czasie. Typowo nie jest możliwe więc zasymulowania wciśnięcia dwóch przycisków myszy jednocześnie. Rozwiązaniem tego problemu może być implementacja dodatkowych gestów i zmapowanie ich na wciśnięcie dwóch klawiszy myszy na raz. Biorąc jednak pod uwagę rzadkość używania tego rodzaju komend zdecydowano się na pominięcie tego problemu i skupieniu się na lepszym działaniu aplikacji w typowych zastosowaniach.

Innym problemem jest gest typu naciśnij i przeciągnij. W pracy zaimplementowano obsługę tego typu akcji dla lewego przycisku myszy. Wykonywanie tej akcji prawym lub środkowym przyciskiem jest rzadkie, dlatego też aplikacja nie obsługuje tych komend. Dzięki takiemu rozwiązaniu komunikat puszczenia klawisza zostaje wysłany natychmiast po wykonaniu gestu. Powoduje to szybszą reakcję na gest (nie trzeba pokazać gestu neutralnego w celu wysłania komunikatu puszczenia klawisza myszy – jak w przypadku lewego przycisku).

4.3.1 Polski alfabet palcowy

W pracy zdecydowano się wybrać gesty z polskiego alfabet palcowego [36]. Zostały one wyselekcjonowane, ponieważ są dobrze znane oraz opisane w literaturze. Dodatkowo wykonanie wybranych gestów nie jest trudne i ich nauka nie wymaga wiele czasu.

Polski alfabet palcowy składa się z 36 gestów dłoni oznaczających kolejne litery alfabetu polskiego. Część z nich jest dynamiczna, jednak dla potrzeb rozwiązania w tej pracy nie były one brane pod uwagę, ze względu na zarezerwowanie ruchu dłoni do poruszania kursorem po ekranie.

Gest neutralny (niewykonujący żadnego zdarzenia) jest wyjątkiem i nie odpowiada żadnej literze z polskiego alfabetu palcowego. Został jednak dobrany jako najbardziej

naturalne ułożenie ręki podczas trzymania jej przed kamerą internetową. Pozostałe gesty zostały dobrane w celu zapewnienia prostego nauczenia się ich, jak również łatwego rozpoznania przez system.

Litera R została wybrana ze względu na jej znaczącą różnicę w kształcie w stosunku to wszystkich pozostałych liter. Jak pokazują wyniki zawarte w podrozdziale 4.5.5 system rozpoznaje ten gest z dużą poprawnością. Dodatkowo ułożenie ręki jest intuicyjnie podobne do kliknięcia.

Zaciśnięta dłoń (litera A), jest intuicyjnie przyjęta, jako przeciąganie ekranu, czyli środkowy przycisk myszy, dlatego też gest ten został użyty w systemie. Podobnych założeń użyto w pracy [48].

Gesty i oraz Y nie są odpowiednie przy dużych zniekształceniach w obszarze dłoni wniesionych podczas działania systemu w słabych warunkach oświetleniowych. Pojedyncze palce zostają często usunięte z obszaru, przez co symbol zamienia się w zupełnie inny i niemożliwa jest jego poprawna identyfikacja.

Gesty U i W są trudne w wykonaniu i wymagają dużego ruchu dłoni, co powoduje niechciany ruch kursora. Dla wygody użytkownika systemu ich rozpoznawanie nie zostało zaimplementowane.

W systemie nie powinny się znaleźć również gesty, które mogłyby zostać błędnie zaklasyfikowane podczas zmiany pozycji ręki w trakcie wykonywania innych gestów. Przykładem takiego gestu jest litera B. Podczas zmiany gestu z dłoni otwartej na gest R system mógłby błędnie zaklasyfikować zmianę tą jako gest B.

Gesty E, P, M i N wymagają ruchu całej dłoni, a nie pojedynczych palców przez co ich wykonanie jest męczące i wnosi niechcianą informację o ruchu całej dłoni podczas ich wykonywania.

Z pozostałych gestów tj, C, L, O, S i T został wybrany gest T jako najbardziej różniący się (Tabela 4.1)od wcześniej przyjętych już gestów zaimplementowanych w systemie. Wykonanie tego gestu skutkuje komunikatem wciśnięcia prawego przycisku myszy.

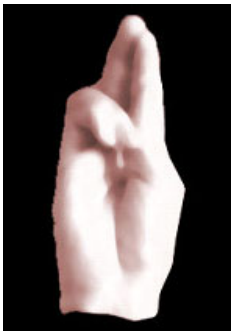
4.3.2 Gest neutralny (otwarta dłoń)

Podstawowym gestem jest otwarta dłoń (rysunek 4.2). Jest to gest, po wykonaniu którego, nie są wykonywane zdarzenia systemowe, chyba że wcześniej pokazywany był gest odpowiedzialny za kliknięcie lewego przycisku myszy. Służy on jako gest pomocniczy przy operacji *przeciągnij i upuść*. Dodatkowo trzymanie w ten sposób ręki z dużym prawdopodobieństwem nie spowoduje błędnego rozpoznania jednego z trzech pozostałych gestów. Wszystkie palce są wyprostowane i lekko rozchylone. Gest ten jest podobny do litery B, z tą różnicą, że palce są rozszerzone.

Gest	Moment 1	Moment 2	Moment 3	Moment 4
Dłoń otwarta	0,199312	0,00594	0,000497	0,000109
A	0,180162	0,00460	0,000687	3,15E-05
B	0,212416	0,01857	0,000273	1,76E-05
C	0,226635	0,01778	0,000361	3,85E-05
E	0,207315	0,01146	0,002259	0,000163
I	0,189291	0,00249	0,000830	0,000242
L	0,218537	0,01058	0,001709	0,000252
M	0,188192	0,00217	0,002219	1,52E-05
N	0,209240	0,00435	0,002778	0,000147
O	0,187093	0,00432	0,000648	1,11E-05
P	0,195818	0,00803	0,001647	0,000107
R	0,226559	0,02279	0,000672	0,000242
S	0,196295	0,00577	0,000790	1,05E-05
T	0,215321	0,01631	0,000756	2,65E-05
U	0,207091	0,01277	0,000523	0,000131
W	0,221144	0,01782	0,000281	3,89E-05
Y	0,231407	0,01444	0,000550	0,000784

Tabela 4.1: Zestawienie momentów różnych gestów z polskiego alfabetu palcowego

Rysunek 4.2: Gest neutralny



Rysunek 4.3: Gest R

Rysunek 4.4: Gest A

Rysunek 4.5: Gest T

4.3.3 Litera R

Kolejnym gestem jest litera R (rysunek 4.3). Dwa palce (wskazujący i środkowy) są wyprostowane, a pozostałe są zgięte. Kciuk jest umiejscowiony po wewnętrznej części dłoni. Wykonanie tego gestu skutkuje wysłaniem do systemu komunikatu naciśnięcia lewego przycisku myszy.

4.3.4 Zaciśnięta dłoń (litera A)

Zaciśnięta dłoń, czyli litera A (rysunek 4.4). W celu wykonania tego gestu należy zwyczajnie zacisnąć dłoń. Kciuk może być schowany pod innymi palcami, lub położony na nich. Wykonanie tego gestu powoduje wysłanie do systemu komunikatu naciśnięcia środkowego przycisku myszy.

4.3.5 Litera T

Ostatni gest to litera T (rysunek 4.5). Wykonuje się go poprzez złączenie palca wskazującego i kciuka tworząc literę T, a pozostałe palce są swobodnie wyprostowane. Po wykonaniu tego gestu następuje wysłanie komunikatu prawego przycisku myszy.

4.4 Detekcja dłoni na obrazie

Detekcja dłoni w obrazie to pierwszy krok algorytmu rozpoznania gestów. Krok ten ma znaczący wpływ na jakość całego procesu rozpoznawania. Projektując system rozpoznający gesty należy wziąć pod uwagę różnice anatomiczne w rozmiarach dłoni użytkowników. Dodatkowo gesty wykonywane przez różnych ludzi mogą znacząco się różnić, co również należy uwzględnić w tego typu systemach.

4.4.1 Przetwarzanie wstępne obrazu

Przed wykryciem obszarów, w których znajduje się dłoń, dokonywane jest przetwarzanie wstępne obrazu. W jego trakcie następuje poprawa jakości obrazu i dostosowanie parametrów do wykrywania obiektów. W programie zastosowano tylko jedną operację przetwarzania wstępnego. Jest to operacja otwarcia. Zastosowana na obrazie kolorowym działa jak filtr niskoczęstotliwościowy dzięki czemu następuje redukcja szumu.

W systemie nie zastosowano innych popularnych algorytmów przetwarzania wstępnego, jak wyrównanie histogramu czy poprawa ostrości lub kontrastu ze względu na wymagania szybkości działania aplikacji. W perspektywie rozwoju i poprawy funkcjonalności programu możliwe jest rozszerzenie tego modułu w celu poprawienia wyników ostatecznej detekcji kosztem szybkości działania.

4.4.2 Wykrywanie obszarów skóry

Wykrywanie obszarów skóry to jeden z ważniejszych etapów w rozpoznaniu dłoni. Gdy już w początkowej fazie detekcji obraz dłoni po segmentacji koloru będzie zawierać dużo zniekształceń i nieprawidłowości dalsza analiza będzie utrudniona, a czasem wręcz niemożliwa.

Pewnym sposobem na poprawienie wyników tego kroku jest stosowanie rękawiczek zawierających czujniki, bądź kolorowych rękawiczek w specyficznym, jaskrawym kolorze – łatwym do rozpoznania i segmentacji [51]. Używanie tego typu ułatwień poprawia jakość klasyfikacji, jednak narzuca użytkownikowi konieczność korzystania z dodatkowego sprzętu, który nie zawsze jest dostępny, a w pewnych zastosowaniach może być nieelegancki i niepraktyczny. Dodatkowo, rozwiązanie to wymaga większych kosztów, jakie wiążą się z zakupem rękawiczek oraz zmniejszają atrakcyjność rozwiązania poprzez zmuszenie użytkownika do nakładania ich podczas używania programu.

Istnieje wiele metod rozpoznawania [1] i wyszukiwania obszarów dłoni na podstawie informacji o kolorze, jak również informacji o samej jasności [28].

W pracy [29] została opisana metoda bazująca na badaniu histogramu poszczególnych obrazków. Zbiór uczący został przygotowany przez ludzi, którzy zaznaczali piksele skóry na obrazach testowych. W ten sposób powstał zbiór zawierający szereg pikseli, dzięki czemu możliwe było określenie dla każdego koloru osobno czy jest to kolor skóry. Wykrywanie czy dany piksel przedstawia skórę ludzką odbywa się poprzez sprawdzenie informacji z bazy danych dotyczącej dokładnie tego koloru. Etykietowanie każdego z kolorów jest jednak bardzo pracochłonne i wymaga dużych nakładów pracy ludzi.

W niniejszej pracy zastosowano rozwiązanie opierające się na segmentacji skóry z obrazu kolorowego uzyskanego z kamery internetowej. W celu uzyskania lepszego wyniku obok standardowej przestrzeni barw RGB – czyli trzykanałowego przedstawienia kolorów jako składowych kolorów czerwonego, zielonego i niebieskiego, użyte zostały przestrzenie barw HSV i YCbCr jako mniej wrażliwe na luminancję (jasność) [30].

Dostosowując system do różnych parametrów ekspozycji zastosowano rozwiązanie polegające na implementacji kilku niezależnych algorytmów dających różne wyniki dla różnego oświetlenia. Dzięki temu możliwy jest dobór algorytmu do aktualnie panujących warunków.

Poniżej zostały przedstawione różne sposoby segmentacji skóry (użytkownik ma możliwość wyboru, w celu dobrania sposobu dającego najlepsze rezultaty w warunkach, w których się znajduje).

Zastosowano następujące rozwiązania:

- badanie zależności kolorów w przestrzeni RGB (5 różnych funkcji),

- badanie zależności kolorów w przestrzeni HSV, oraz dodatkowe usprawnienie z przestrzeni RGB,
- badanie zależności kolorów w przestrzeni YCbCr, oraz dodatkowe usprawnienie w przestrzeni RGB.

Wszystkie z wymienionych funkcji są opisane szczegółowo w dalszej części pracy.

Metoda preselekcji pikseli

W celu lepszego rozpoznania czy badany piksel należy do skóry ludzkiej, został zaimplementowany algorytm preselekcji pikseli [50]. Usuwa (zaczernia) on piksele niebędące, z dużym prawdopodobieństwem, pikselami należącymi do obszaru skóry ludzkiej. Algorytm ten działa w przestrzeni RGB. Dopiero po wykonaniu preselekcji pikseli następuje rzeczywiste badanie jednym z dokładniejszych algorytmów, wspomnianych wcześniej. Dla wszystkich filtrów zakresy składowych RGB to 0 - 255.

W wyniku eksperymentów ustalono, że dla preselekcji niepoprawnych pikseli należy zastosować następujące sprawdzenia:

- niebieski powyżej 160, a zielony i czerwony kanał poniżej 180 – filtr ten ma wychwycić zbyt niebieskie piksele;
- zielony powyżej 160, a niebieski i czerwony kanał poniżej 180 – filtr ten ma wychwycić zbyt zielone piksele;
- wszystkie kanały poniżej 70 – ma to usunąć piksele będące zbyt ciemne;
- suma kanału czerwonego i zielonego powyżej 400, a niebieski poniżej 170 lub zielony powyżej 110, a niebieski poniżej 90 – filtr ten ma wychwycić piksele zbyt czerwonozielone (kolor żółtawy);
- niebieski kanał podzielony na sumę wszystkich kanałów większy od 0.4 – filtr ten ma usunąć zbyt niebieskie piksele w kontraście do innych;
- zielony kanał podzielony na sumę wszystkich kanałów większy od 0.4 – filtr ten ma usunąć zbyt zielone piksele w kontraście do innych;
- czerwony kanał mniejszy od 102, zielony w zakresie od 100 do 140, niebieski w zakresie od 110 do 160 – kolor podobny do oceanu. Filtr ten nie jest zawsze skuteczny – np. w oświetleniu niebiesko – zielonym, np. w przypadku pomalowania ścian na taki kolor, odbite światło może spowodować błędne działanie filtru.

Podczas obserwacji pikseli należących do skóry w programie graficznym, zauważono zależność, że zawsze składowa czerwona ma największą wartość, oraz że przeważnie składowa zielona ma większą wartość niż składowa niebieska stąd też propozycja kolejnego filtru:

- wartość kanału czerwonego musi być większa niż kanałów zielonego i niebieskiego.

Dodatkowo dla skóry ludzkiej wszystkie składowe koloru są o podobnej wartości, to jest różnią się maksymalnie o około 10. Jednak, gdy w trakcie akwizycji mamy do czynienia ze słabym oświetleniem, zdarza się, że powyższe stwierdzenie jest niepoprawne. Filtr sprawdzający jedynie różnicę poszczególnych kanałów błędnie usuwałby piksele będące skórą. Dlatego też należy wprowadzić dodatkowy warunek na jasność pikseli. Podsumowując powyższe stworzono następujący filtr:

- absolutna różnica pomiędzy kanałem czerwonym i zielonym nie może być większa niż 10, pomiędzy zielonym i niebieskim nie może być większa niż 20, a pomiędzy czerwonym, a niebieskim nie większa niż 30. Dodatkowo filtr ten ma zastosowanie jedynie, gdy składowa czerwona jest większa niż 200.

Rozpoznawanie skóry w przestrzeniach RGB

Popularność przestrzeni RGB wpłynęła na dużą liczbę algorytmów pracujących w tej przestrzeni. Rozpoznawanie skóry w przestrzeni RGB jest jednak bardzo kontrowersyjne, ponieważ przestrzeń ta daje słabe wyniki w segmentacji zadanego zakresu koloru. Duże różnice we wszystkich składowych barw wnoszone są do tego systemu przez różne oświetlenie i jasność akwizycjonowanego obrazu.

Dlatego w systemie zdecydowano się na badanie stosunku poszczególnych składowych, a nie ich wartości. Dzięki temu uniezależnia się algorytm od jasności. Często bada się również stosunek poszczególnych składowych do sumy wszystkich składowych przekształcając w ten sposób przestrzeń RGB w znormalizowaną przestrzeń RGB [39].

Zaimplementowane funkcje zostały stworzone na podstawie rozwiązań wielu autorów. Wszystkie wartości liczbowe zostały dobrane empirycznie podczas testów w trakcie tworzenia aplikacji. Dopasowywanie odbywało się poprzez podpięcie każdej z wartości liczbowych do kontrolek w programie. Następnie po uruchomieniu następowało dopasowywanie każdej z wartości dla otrzymania jak najlepszych wyników wg subiektywnej oceny autora.

Wiele warunków zostało stworzonych metodą prób i błędów. Dzięki wnikliwej obserwacji charakterystyki koloru skóry w programie graficznym możliwe było sformułowanie dodatkowych zależności poprawiających działanie algorytmu. Kilka warunków jest

umieszczonych w wielu funkcjach ze względu na znaczący ich wpływ na poprawę segmentacji skóry. Funkcje dopasowano do różnych warunków oświetleniowych.

Poniżej przedstawiono kolejno każdą z funkcji. W systemie zaimplementowano pięć różnych funkcji wykrywających skórę w przestrzeni RGB. Funkcje te używane są zamiennie. Jak opisano dalej (podrozdział 4.4.5) wykorzystywanie różnych funkcji pozwala na lepsze dopasowanie systemu do aktualnie panujących warunków oświetlenia.

We wszystkich funkcjach przyjęto oznaczenia:

- r – składowa czerwona w zakresie 0 - 255;
- g – składowa zielona w zakresie 0 - 255;
- b – składowa niebieska w zakresie 0 - 255;
- sum – suma wszystkich trzech składowych.
- MAX(a,b)/MIN(a,b) - wartość maksymalna/minimalna ze zmiennych a i b.

Funkcja 1 Funkcja ta opiera się na badaniu zależności poszczególnych składowych. Wartości liczbowe zostały dobrane podczas testów.

```
if (r/b > 1.185 &&
        (r*b)/(r+g+b)^2 > 0.107 &&
        (r*g)/(r+g+b)^2 > 0.112)
        //jest to kolor skóry
else
        //nie jest to kolor skóry
```

Funkcja ta jest bardzo prosta, ale i wyniki przez nią osiągane nie są wystarczające do większości zastosowań. Niestety przy jej stosowaniu wiele obszarów niebędących skórą zostaje błędnie zaklasyfikowanych, przez co można jej logicznie używać tylko w połączeniu z usuwaniem informacji o tle i będąc ubranym w ubrania o kolorze niezbliżonym do koloru skóry.

Funkcja 2 Kolejna funkcja jest już bardziej zaawansowana. Wyniki tej funkcji dają bardzo dobre rezultaty dla skóry w sztucznym oświetleniu. Jednak występuje tu podobny problem jak w przypadku funkcji 1 tzn. błędna klasyfikacja obszarów niebędących skórą.

```
if (r > 95 && g > 40 && b > 20
        && MAX(MAX(r,g),b) - MIN(MIN(r,g),b) > 15
        && abs(r-g) > 15
        && r > g
```

```
        && r > b
        && (b/g < 1.249)
        && b/g > 0.5
        && (sum/(3*r) > 0.692)
        && (0.3333-b/sum > 0.029)
        && (g/(3*sum) < 0.124)
                || (3*b*r*r)/(sum*sum*sum) > 0.110
        && ((r*b + g*g ) / g*b) > 5000
        && sum/(3*r + (r-g)/sum) <2.7775)
        //jest to kolor skóry
else
        //nie jest to kolor skóry
```

Funkcja 3 Kolejna prosta funkcja, jednak jej rezultaty czasem okazują się lepsze od funkcje pierwszej i drugiej.

```
if (g / b - r / b <= -0.0905)
        && (sum / (3*r) + (r- g) / sum <= 0.9498)
        //jest to kolor skóry
else
        //nie jest to kolor skóry
```

Funkcja 4 Funkcja ta korzysta z zależności poszczególnych składowych do sumy wszystkich składowych. W ten sposób uniknięto wpływu jasności na działanie funkcji.

```
if ((b/g < 1.249)
        && (sum/(3*r) > 0.692)
        && (0.3333-b/sum > 0.029)
        && (g/(3*sum) < 0.124))
        //jest to kolor skóry
else
        //nie jest to kolor skóry
```

Funkcja 5 Funkcja piąta daje dobre wyniki tylko w specyficznych warunkach oświetleniowych. W większości przypadków rezultaty przez nią osiągane są niezadawalające.

```
if ( g/b - r/g <= -0.0905
        && (g*sum)/(b*(r-g)) > 3.4857
```

```
        && (sum*sum*sum)/(3*g*r*r) <= 7.397
        && sum/(9*r)-0.333 > -0.0976)
        //jest to kolor skóry
else
        //nie jest to kolor skóry
```

Rozpoznawanie skóry w przestrzeniach HSV

W celu poprawienia jakości rozpoznawania skóry na obrazie dobrym pomysłem jest pozbycie się wartości luminancji ze składowych koloru. Rozwiązaniem jest przejście w przestrzeń kolorów, która składową luminancji przechowuje osobno. Dzięki temu pozostałe składowe koloru nie są skorelowane z jasnością i na ich podstawie można dokładniej określić przestrzeń barw reprezentujących skórę.

W programie zaimplementowano funkcję działającą w przestrzeni HSV. W celu ulepszenia wyników dodano kilka warunków użytych w funkcjach z przestrzeni RGB. Poprawiło to klasyfikację i sprawiło, że funkcja osiąga często najlepsze wyniki w sztucznym oświetleniu.

Podczas badania zakresu wartości przestrzeni HSV, w jakich znajduje się barwa skóry ludzkiej osiągnięto następujące konkluzje:

- w rzeczywistości model HSV jest mniej podatny na zmianę naświetlenia podczas akwizycji,
- w celu wyodrębnienia skóry ludzkiej wartość barwy (składowej H) powinna mieścić się w zakresie od 244 – 267 (w skali 0-360),
- wartość nasycenia (składowej S) powinna znajdować się w zakresie od 34 do 80 (w przedziale 0-100),
- wartość jasności (składowej V) powinna zostać pominięta, jako nieposiadająca istotnych informacji dla metody wykrywania skóry ludzkiej, a stwarzająca dużo problemów przy różnym oświetleniu.

Rozpoznawanie skóry w przestrzeniach YCbCr

Kolejną przestrzenią, w której luminancja jest wydzielona jako jedna ze składowych koloru jest przestrzeń YCbCr. W systemie zaimplementowano funkcję opierającą się na tej przestrzeni. Również i w tym algorytmie dobrym rozwiązaniem okazało się dodanie kilku warunków z przestrzeni RGB. Ostatecznie powstała funkcja, która osiąga bardzo

dobre wyniki, a w niektórych warunkach akwizycji daje najlepsze rezultaty ze wszystkich zaimplementowanych algorytmów.

W trakcie badań najlepsze rezultaty funkcja osiągnęła, gdy:

- składowa Cb znormalizowana do przedziału 0 - 255 powinna przyjmować wartości od 110 do 141,
- składowa Cr znormalizowana do zakresu 0 - 255 powinna przyjmować wartości od 128 do 155,
- składowa Y została pominięta.

4.4.3 Usuwanie informacji o tle

Nawet najlepszy klasyfikator oparty tylko na informacji o kolorze pikseli nie jest w stanie zapewnić poprawnego rozpoznawania w każdych warunkach. W przypadku, gdy w otoczeniu jest dużo przedmiotów drewnianych, np. meble, lub też ściany pomieszczenia, w którym znajduje się użytkownik systemu, pomalowane są na kolor podobny do skóry ludzkiej, rozpoznawanie w oparciu tylko o kolor będzie wprowadzało do systemu dużo błędnych informacji.

W celu znaczącej poprawy klasyfikacji pojedynczych pikseli w systemie, został zaimplementowanych mechanizm eliminacji statycznych pikseli tła. Ma on na celu usunięcie z przetwarzanego obrazu tych pikseli, które z dużym prawdopodobieństwem przedstawiają tło, a nie użytkownika systemu [3].

Dla uzyskania funkcjonalności powyższego filtru, w pierwszej kolejności, należy dostarczyć do systemu informację o tle, na jakim będzie następowało rozpoznawanie dłoni. Użytkownik, przed przystąpieniem do pracy, powinien skalibrować program. Może to uczynić poprzez wykonanie poleceń automatycznej kalibracji po włączeniu programu lub ręcznie za pomocą klawiatury. Dodatek A zawiera szczegółowa instrukcję kalibracji.

Następnie każda kolejna klatka sekwencji uzyskanej z kamery jest porównywana piksel po pikselu z obrazem tła. Jeżeli piksel aktualnej klatki nie będzie różnił się od odpowiadającego mu piksela zawartego w obrazie tła, zostanie on zaklasyfikowany, jako nieprzedstawiający skóry ludzkiej. Różnica pomiędzy badaymi pikselami nie może przekraczać pewnej z góry ustalonej wartości. W tym kroku zostaną usunięte wszystkie piksele, przedstawiające obiekty statyczne. Przez usunięcie piksela, rozumie się zamienienie jego koloru na kolor czarny, czyli taki, który nie jest klasyfikowany przez żaden algorytm, jako należący do skóry ludzkiej.

Szum wysokoczęstotliwościowy nie jest brany pod uwagę ze względu, że nawet gdy zaszumione piksele zostaną przepuszczone przez algorytm, nie utworzą one obszaru cią-

głego, a tylko taki może być zaklasyfikowany jako dłoń. Dlatego też zdecydowano się na nieusuwanie szumu i drobnych zniekształceń jak przesunięcie pikseli w celu nie zmniejszania wydajności algorytmu.

Przefiltrowany przez powyższy algorytm obraz, trafia następnie do funkcji rozpoznającej skórę. Dalsze działanie systemu przebiega bez zmian.

Niestety, dla dynamicznie zmieniającego się tła, filtr ten będzie bezużyteczny, jako że informacja o tle dostarczonym do systemu będzie szybko się dezaktualizowała i konieczna byłaby ponowna kalibracja systemu.

Wpływ zmiany warunków oświetlenia

Podczas działania systemu zaobserwowano znaczący wpływ nawet minimalnej zmiany oświetlenia obrazu. Podczas normalnego użytkowania systemu pobrana informacja o tle w podstawowej wersji algorytmu traci ważność już po chwili. Sterownik kamery ma za zadanie utrzymywać ogólną jasność obrazu na podobnym poziomie, co z naszego punktu widzenia skutkuje częstą zmianą jasności pikseli w całym obrazie. Przykładowo mocne doświetlenie części tła, lub naszej dłoni spowoduje ogólne przyciemnienie całego obrazu uzyskanym z kamery. Jest to spowodowane wyrównywaniem histogramu obrazu na etapie akwizycji w sterowniku kamery internetowej.

Dla wyeliminowania powyższej zależności zastosowano usprawnienie algorytmu usuwania tła. Polega ono na automatycznej, wstępnej zmianie jasności pikseli obrazu referencyjnego (obrazu tła pobranego w trakcie kalibracji). Jest to czynione poprzez porównanie kilkudziesięciu pikseli znajdujących się w lewej górnej części obrazów. Na podstawie różnicy pomiędzy obrazem aktualnym, a referencyjnym następuje obliczenie uśrednionej zmiany jasności. Zmiana ta jest brana następnie pod uwagę w trakcie usuwania pikseli należących do tła. W ten sposób algorytm staje się bardziej niezależny od oświetlenia.

Powyższe usprawnienie pomaga w przypadku zmian jasności światła. Przy dużej zmianie barwy oświetlenia należy ponownie skalibrować system.

4.4.4 Używanie sieci neuronowej w rozpoznawaniu koloru

Do rozpoznawania skóry ludzkiej na obrazie można również użyć sieci neuronowych. Klasyfikatory oparte o sieć neuronową osiągają dużą dokładność. Są one jednak dużo wolniejsze od prostych porównań i zabiegów zaimplementowanych w systemie [34].

Dodatkowym problemem staje się dostarczenie odpowiedniego zbioru uczącego, mogącego pokryć jak najwięcej przypadków przy małym współczynniku błędnej klasyfikacji. W pracy [47] przedstawiono metodę rozpoznawaniu koloru skóry opartej o sieć neuronową działającą w przestrzeni kolorów TLS. W rozwiązaniu tym używane jest wiele obrazów

wejściowych przedstawiających skórę ludzi różnej narodowości i w różnych warunkach oświetlenia.

4.4.5 Badanie i wnioski używania różnych metod

W pracy zdecydowano się na segmentacje skóry poprzez proste algorytmy oparte na modelu skóry w różnych przestrzeniach kolorów. Wybór ten został dokonany ze względu na wysoką jakość uzyskanych wyników w stosunku do potrzebnej mocy obliczeniowej. Dodatkowym atutem wybranego rozwiązania jest szybkość segmentacji, co jest kluczowym czynnikiem podczas implementacji systemu czasu rzeczywistego.

Wykonano szereg testów mających ustalić jakość poszczególnych algorytmów segmentacji. Wyniki zamieszczono w tabelach 4.2-4.6. Plus i minus przy nazwie funkcji w tabeli oznacza, czy badana funkcja używa filtru wstępnego. Litery O, R, A, T odpowiadają wykonywanym gestom (O oznacza gest neutralny). Kolorem szarym zaznaczono funkcje dające najlepsze rezultaty.

W tabeli 4.2 wyniki funkcji YCbCr+, HSV+, RGB2+ i filtru wstępnego dla litery T osiągają bardzo dużą stopę błędu. Spowodowane jest to faktem, że w trakcie badania tego gestu obiekt z tła nie będący ręką spowodował błędne zaklasyfikowanie jako gest neutralny. Każdy z gestów był badany osobno. Podczas badania jednego gestu testowane były jednocześnie wszystkich funkcje segmentacji.

Na wykresach 4.6-4.10 pokazano uśrednione wartości rozpoznawania dla poszczególnych funkcji w różnych warunkach. Uzyskane wyniki świadczą o różnym działaniu poszczególnych funkcji w testowanych warunkach oświetleniowych. Jedynie w trakcie dnia, wewnątrz nieprześwietlonego pomieszczenia, wszystkie klasyfikatory uzyskiwały dobre wyniki (wykres 4.7). Dla pozostałych warunków można wyszczególnić tylko kilka klasyfikatorów uzyskujących zadowalające wyniki. Najgorsze wyniki zostały uzyskane w czasie nocy podczas gdy źródłem światła była jedynie lampka nocna (wykres 4.10).

Testy przeprowadzane były w otoczeniu rzeczywistym, tzn. w pomieszczeniu znajdowało się dużo obiektów mogących negatywnie wpłynąć na klasyfikacje. Były to drewniane szafy, książki itp. Test wykonany na zewnątrz został przeprowadzony przy użyciu laptopa z wbudowaną kamerą, trzymanego na kolanach, przez co niemożliwa była filtracja statycznego tła.

Wyniki pokazują, że nie można jednoznacznie określić najlepszej z funkcji. Nawet niewielkie zmiany w oświetleniu, mogą znacząco wpłynąć na działanie poszczególnych algorytmów. Dlatego w systemie zaimplementowano wszystkie z badanych procedur, oraz opracowane automatyczną metodę wybieranie tej, która daje najlepsze wyniki w panujących warunkach. Dodatkowo użytkownik ma możliwość zmiany algorytmu w trakcie

Gest	Poprawna klasyfikacja (%)				Nierozpoznane gesty (%)				Błędna klasyfikacja (%)			
Funkcja	O	R	A	T	O	R	A	T	O	R	A	T
HSV+	92,7	0,3	0	0	4,3	77,7	86,7	30,7	3	22	13,3	69,3
YCbCr+	3	0,3	0	0	96,7	77,7	86	31	0,3	22	14	69
RGB1+	0	0	0	0	99,7	93	99,7	99,7	0,3	7	0,3	0,3
RGB2+	3	0	0	0	96,7	78	89,7	33,7	0,3	22	10,3	66,3
RGB3+	0	0	0	0	99,7	93,3	99,3	99,7	0,3	6,7	0,7	0,3
RGB4+	0	0	0	0	99,3	94,7	93,7	99,7	0,7	5,3	6,3	0,3
RGB5+	0	0	0	0	99,3	93,7	95	99,7	0,7	6,3	5	0,3
HSV-	79	93	92	85,7	18,7	5,7	3,7	14	2,3	1,3	4,3	0,3
YCbCr-	75,3	93	93,7	92	23	5,7	3	7,7	1,7	1,3	3,3	0,3
Wstępny	92,3	0,3	0	0	4,3	77,7	86,3	30,7	3,3	22	13,7	69,3

Tabela 4.2: Zestaw wyników klasyfikatorów w warunkach: dzień, pełne słońce, na zewnątrz

Gest	Poprawna klasyfikacja (%)				Nierozpoznane gesty (%)				Błędna klasyfikacja (%)			
Funkcja	O	R	A	T	O	R	A	T	O	R	A	T
HSV+	93,3	89,3	81,6	91,3	5	9,4	13	7	1,7	1,3	5,4	1,7
YCbCr+	94,3	82,9	89,6	70,9	5,4	8,7	4,3	27,4	0,3	8,4	6	1,7
RGB1+	95	78,3	84,3	80,6	5	1	10	15,7	0	20,7	5,7	3,7
RGB2+	92,3	74,6	79,3	74,6	7,7	17,1	2,7	25,4	0	8,4	18,1	0
RGB3+	91	72,9	86,3	57,5	8,7	27,1	11,4	38,8	0,3	0	2,3	3,7
RGB4+	89	63,9	44,1	41,8	10,7	16,7	28,8	33,4	0,3	19,4	27,1	24,7
RGB5+	98,3	50,5	62,9	49,5	1,3	11,7	22,1	35,1	0,3	37,8	15,1	15,4
HSV-	90	51,8	81,3	46,5	5,7	16,1	7	34,1	4,3	32,1	11,7	19,4
YCbCr-	80,3	91,3	71,2	63,5	17,4	0,7	14,7	6	2,3	8	14	30,4
Wstępny	70,2	31,4	27,4	0	26,8	9,4	1,7	9,7	3	59,2	70,9	90,3

Tabela 4.3: Zestaw wyników klasyfikatorów w warunkach: dzień, słaba filtracja tła, wewnątrz, zasunięte żaluzje

Gest	Poprawna klasyfikacja (%)				Nierozpoznane gesty (%)				Błędna klasyfikacja (%)			
Funkcja	O	R	A	T	O	R	A	T	O	R	A	T
HSV+	88	85,7	24,3	1,3	11,3	13,7	25,3	23,3	0,7	0,7	50,3	75,3
YCbCr+	89,3	73	23,3	1	10,7	15,7	24,7	24	0	11,3	52	75
RGB1+	2,3	0	0	0	97,3	99,7	79	99,7	0,3	0,3	21	0,3
RGB2+	88	85,7	24,3	1,7	11,3	13,7	25,3	23	0,7	0,7	50,3	75,3
RGB3+	1,7	0	0	0	98	99,7	82,3	99,7	0,3	0,3	17,7	0,3
RGB4+	1	0	0	0	98,7	99,7	99,7	99,7	0,3	0,3	0,3	0,3
RGB5+	86	85,3	100	73,3	13	14,7	0	26,7	1	0	0	0
HSV-	87,7	96,3	93,7	81	11	3,7	3,7	19	1,3	0	2,7	0
YCbCr-	78	89,3	24,3	9,3	18,7	10,3	25,7	24,3	3,3	0,3	50	66,3
Wstępny	88	85,7	24,3	1,3	11,3	13,7	25,3	23,3	0,7	0,7	50,3	75,3

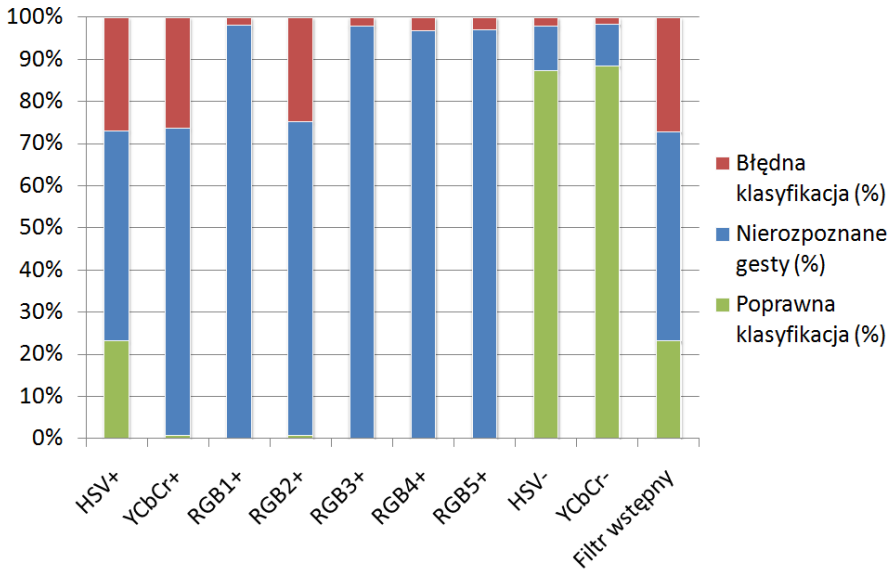
Tabela 4.4: Zestaw wyników klasyfikatorów w warunkach: dzień, dobra filtracja tła, wewnątrz odsłonięte żaluzje

Gest	Poprawna klasyfikacja (%)				Nierozpoznane gesty (%)				Błędna klasyfikacja (%)			
Funkcja	O	R	A	T	O	R	A	T	O	R	A	T
HSV+	97	83,6	81,9	73,9	3	10	15,4	20,1	0	6,4	2,7	6
YCbCr+	96,7	65,9	52,8	45,2	3,3	16,1	36,1	49,2	0	18,1	11	5,7
RGB1+	87,6	1,7	0	0,3	11,4	8	7	40,8	1	90,3	93	58,9
RGB2+	99,7	84,6	78,9	72,6	0	14	20,7	20,4	0,3	1,3	0,3	7
RGB3+	93,6	27,1	28,1	1,3	4,7	54,8	27,8	49,8	1,7	18,1	44,1	48,8
RGB4+	79,3	7,4	13	1,7	20,1	27,1	32,4	39,1	0,7	65,6	54,5	59,2
RGB5+	0	0	0	0	100	100	100	100	0	0	0	0
HSV-	92	93	64,9	72,2	4	3,3	34,8	22,7	4	3,7	0,3	5
YCbCr-	97,3	90,3	63,5	68,9	2	4	31,4	29,4	0,7	5,7	5	1,7
Wstępny	78,6	74,2	70,9	74,2	18,4	21,4	10	25,8	3	4,3	19,1	0

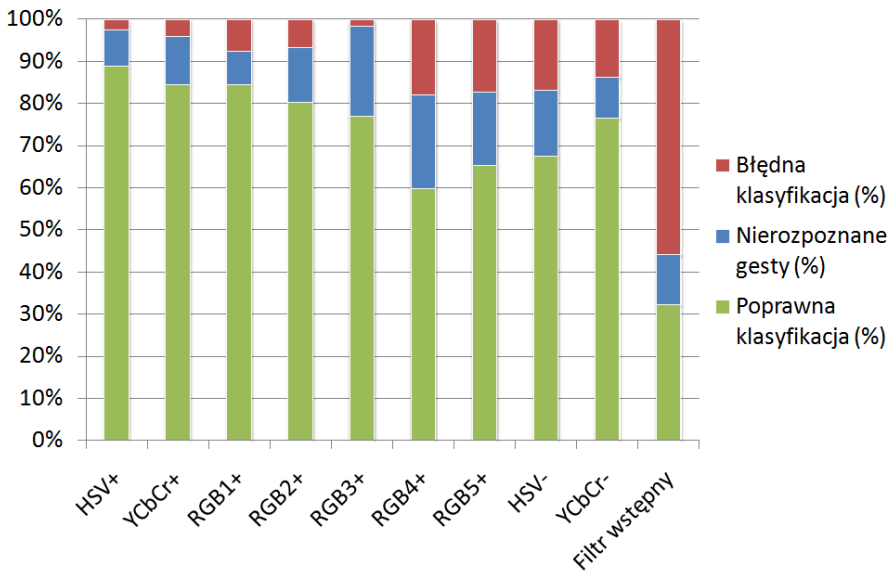
Tabela 4.5: Zestaw wyników klasyfikatorów w warunkach: noc, przy świetle lamp sufitowych (dobre oświetlenie rozproszone)

Gest	Poprawna klasyfikacja (%)				Nierozpoznane gesty (%)				Błędna klasyfikacja (%)			
Funkcja	O	R	A	T	O	R	A	T	O	R	A	T
HSV+	72,3	67	80,7	44,3	27,7	30,3	18,7	54,7	0	2,7	0,7	1
YCbCr+	64,7	5,3	3,3	0,3	34,3	10,7	54,3	94,3	1	84	42,3	5,3
RGB1+	0	0	0	0	99,7	99,7	99,7	99,7	0,3	0,3	0,3	0,3
RGB2+	87	3,3	16,3	1	12,3	17,7	77,7	98,7	0,7	79	6	0,3
RGB3+	10	2,7	0	0	89,7	5,7	64	71,3	0,3	91,7	36	28,7
RGB4+	0	0	0	0	99,3	99,7	99,7	99,7	0,7	0,3	0,3	0,3
RGB5+	84,7	74,3	60,3	48	13,7	15	34	51,7	1,7	10,7	5,7	0,3
HSV-	78,3	46,3	11	26	19	24	72,3	72,7	2,7	29,7	16,7	1,3
YCbCr-	72	67	81	44,3	27,7	30,3	18,3	54,7	0,3	2,7	0,7	1
Wstępny	72,3	67	80,7	44,3	27,7	30,3	18,7	54,7	0	2,7	0,7	1

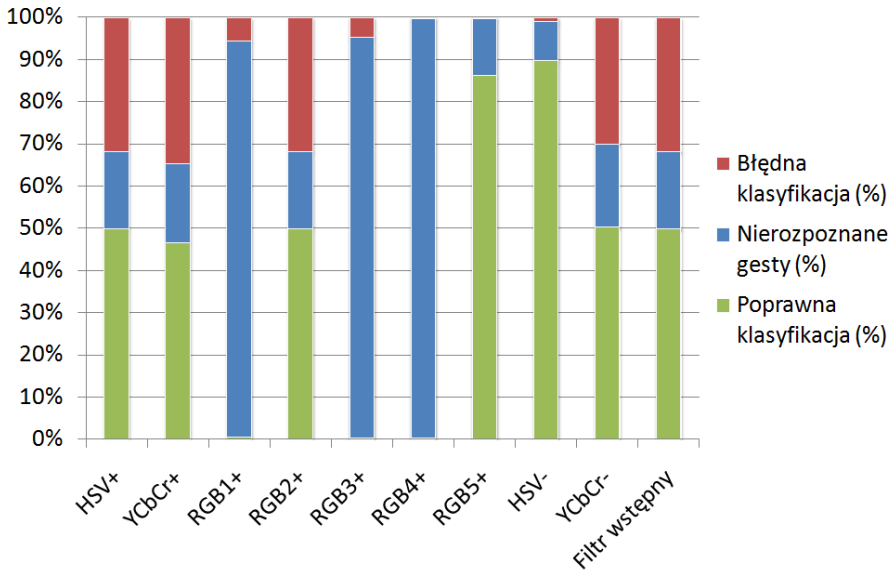
Tabela 4.6: Zestaw wyników klasyfikatorów w warunkach: noc, przy świetle lamki nocnej (mocne oświetlenie punktowe)



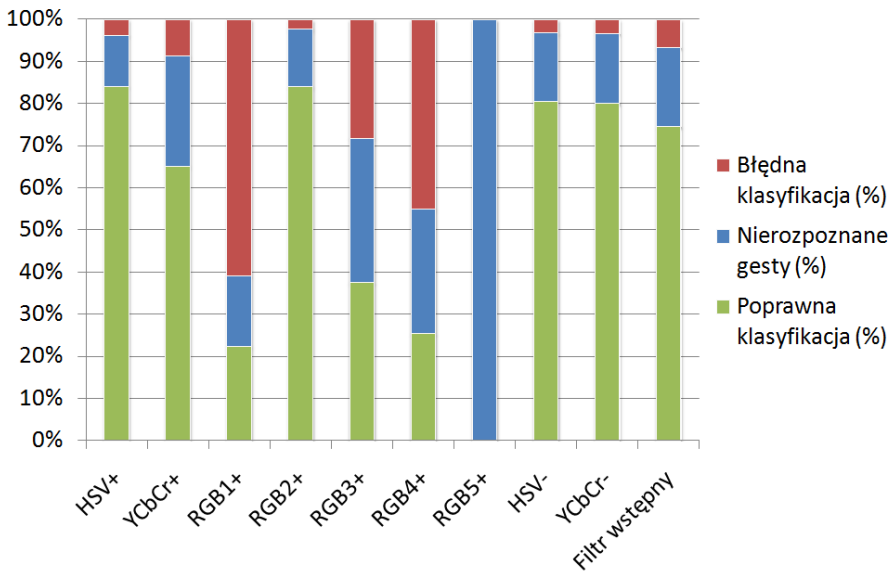
Wykres 4.6: Zestawienie wyników klasyfikatorów w warunkach: dzień, pełne słońce, na zewnątrz



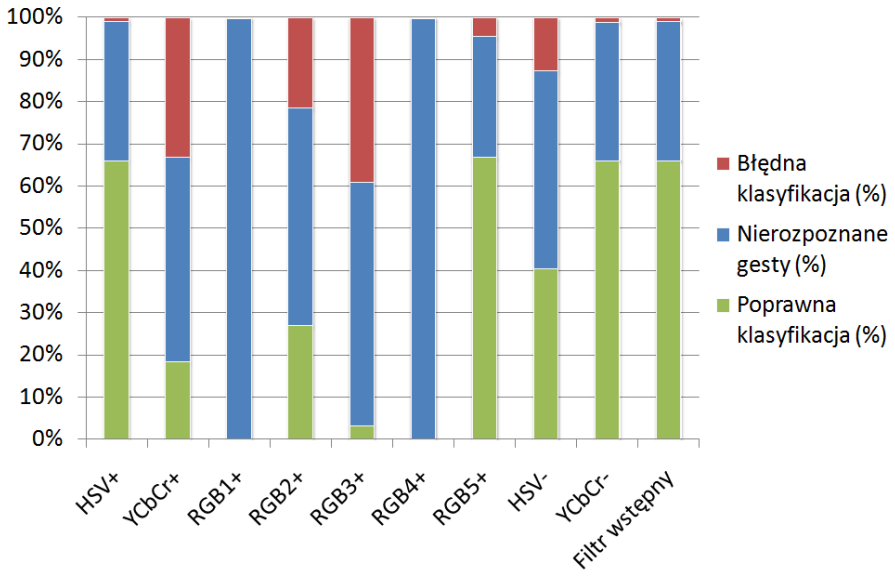
Wykres 4.7: Zestawienie wyników klasyfikatorów w warunkach: dzień, słaba filtracja tła, wewnątrz, zasunięte żaluzje



Wykres 4.8: Zestawienie wyników klasyfikatorów w warunkach: dzień, dobra filtracja tła, wewnątrz odsłonięte żaluzje



Wykres 4.9: Zestawienie wyników klasyfikatorów w warunkach: noc, przy świetle lamp sufitowych (dobre oświetlenie rozproszone)



Wykres 4.10: Zestawienie wyników klasyfikatorów w warunkach: noc, przy świetle lamki nocnej (mocne oświetlenie punktowe)

działania systemu.

Dalsza obserwacja testów pokazuje, że algorytmy osiągały niski współczynnik błędnej klasyfikacji, przeważnie rzędu kilku procent. Ma to pozytywny wypływ na działanie programu, ponieważ w przypadku gdy nie nastąpi poprawna klasyfikacja, algorytm nie zwraca błędnego wyniku, przez co nie są wykonywane gesty, które nie zostały pokazane.

4.5 Rozpoznawanie gestów dłoni

Po rozpoznaniu obszarów skóry następuje detekcja dłoni. W tym celu obliczane są podstawowe cechy wszystkich wykrytych płaszczyzn. Następnie obliczone cechy poddaje się analizie w celu wyodrębnienia obszaru, który z największym prawdopodobieństwem jest dłonią.

Z wyodrębnionego obszaru oblicza się pozostałe cechy i na ich podstawie określony zostaje wykonywany gest. Ustalenie gestu odbywa się poprzez zebranie wyników z kilku niezależnych klasyfikatorów. W przypadku, gdy obszar dłoni jest zniekształcony, a klasyfikatory dają różne wyniki gest nie zostaje określony, dzięki czemu unika się błędnie wykonywanych komend.

4.5.1 Wstępne ustalenie gestu

W pierwszej kolejności liczone są geometryczne momenty bezwładności. Na podstawie pierwszego momentu m_{00} określany jest rozmiar obszaru. Nie może być on zbyt duży, gdyż skutkiem będzie błędna klasyfikacja. Przyczyną niepoprawnej klasyfikacji może być fakt, że dłoń znajduje się za blisko kamery i występować będzie dużo zakłóceń związanych ze słabą akwizycją obrazu dla typowych kamer internetowych z niewielkiej odległości. Dodatkowo, gdy w kadrze pojawi się obiekt o podobnym kolorze do skóry i złączy się on z dłonią, obszar ten będzie zbyt duży i całkowicie zniekształcony, dlatego też w takich przypadkach klasyfikacja tego typu obszarów jest bezcelowa.

Dla pozostałych obszarów liczona jest odległość Mahalanobisa pomiędzy wartością średnią dla każdego gestu obliczoną na podstawie zbioru uczącego, a obecnie badanymi kształtami. Do dalszego rozpoznawania zostaje wybrany obszar, którego odległość Mahalanobisa jest najmniejsza. Jeżeli odległość ta jest większa od ustalonego progu w danej klatce nie następuje rozpoznanie gestu.

Obliczanie odległości Mahalanobisa pomiędzy badaną próbką pozwala na uniknięcie błędnej klasyfikacji obszarów nieprzypominających dłoni. Użycie wartości średniej cech nie jest dokładnym rozwiązaniem, jednak w większości przypadków w zupełności wystarcza by odfiltrować błędne kształty.

4.5.2 Użycie kilku klasyfikatorów

Dla uzyskania lepszych wyników rozpoznawania zastosowano kilka równoległych klasyfikatorów. Rozwiązanie takie jest lepsze, ponieważ osiąga się mniejszą stopę błędów, co w zastosowaniu przy sterowaniu myszką jest bardzo istotne [32].

Negatywnym aspektem zbierania wyników z kilku klasyfikatorów jest obniżenie wydajności algorytmu. Jednak w przypadku klasyfikacji kilku cech zysk na dokładności będzie nieporównywalnie większy w stosunku do straty na wydajności. W trakcie testowania programu okazało się, że operacje związane z klasyfikacją gestu nie zajmują dużo czasu w stosunku do całego algorytmu rozpoznawania dłoni i detekcji ruchu.

W pracy zostały użyte następujące klasyfikatory:

- klasyfikator Bayesa,
- klasyfikator jednego i trzech najbliższych sąsiadów,
- trzy klasyfikatory SVM,
- odległość Mahalanobisa.

Zbiór testowy dla klasyfikatora Bayesa i klasyfikatora najbliższych sąsiadów zawierał wszystkie próbki gestów. Do klasyfikatora SVM przygotowano zbiory uczące zawierające zestawy par gestów. Klasyfikator ten w podstawowej postaci potrafi rozróżniać tylko 2 klasy. W trakcie testowania aplikacji okazało się, że maszyna wektorów nośnych daje dobre wyniki tylko pomiędzy gestami w zestawieniu z gestem R. Pozostałe gesty okazały się słabo separowane liniowo, przez co użycie dla nich klasyfikatora SVM stało się wątpliwe. Wyniki klasyfikatora można zobaczyć w tabelach 4.8-4.11.

4.5.3 Badanie kolejnych klasyfikacji

Dodatkowym udoskonaleniem jest zbieranie informacji o kolejnych rozpoznanych gestach. Dopiero w przypadku, gdy wyniki ze wszystkich klasyfikatorów będą zgodne przez trzy kolejne klatki sekwencji wideo gest jest wykonywany. Pozwala to wyeliminować przypadkowo źle rozpoznane gesty [3].

Uwaga do wyników

Gest T uzyskał najgorszy wynik. Wiąże się to z tym, że często następowała błędna klasyfikacja z gestem neutralnym. W przypadku, gdy gest neutralny nie wykonuje żadnej operacji błędy takie nie będą krytyczne i nie będą miały znaczącego wpływu na działanie aplikacji.

Gest	Poprawna klasyfikacja	Brak klasyfikacji	Negatywna klasyfikacja	3 negatywne klasyfikacje pod rząd
Neutralny	84,4%	14,7%	0,9%	0,1 %
R	79%	20,7%	0,3%	0,0 %
A	81,3%	17,1%	1,6%	4,3%
T	70%	18%	12%	0,0 %

Tabela 4.7: Wyniki testów dla 3 kolejnych klasyfikacji gestów

4.5.4 Eliminacja twarzy z rozpoznawania dłoni

Podczas badania obszarów wydzielonych przez algorytmy segmentacji skóry częstym przypadkiem była błędna klasyfikacja obszarów głowy jako dłoni. Dla uniknięcia tego efektu, w programie zastosowano asynchroniczne wykrywanie twarzy. Informacja o znalezionej twarzy trafia do klasyfikatora obszarów i eliminuje te, w których środek ciężkości (obliczony podczas wyznaczania geometrycznych momentów bezwładności) znajduje się wewnątrz obszaru twarzy.

Detekcja twarzy opiera się na implementacji zawartej w bibliotece OpenCV. Algorytm ten jest jednak czasochłonny, dlatego też obliczenia z nim związane wydzielono do osobnego wątku, a samo wykrywanie twarzy następuje około 30 razy rzadziej od wykrywania i analizy dłoni. Optymalizacja taka była możliwa, ponieważ ruch głowy jest stosunkowo niewielki.

Klasyfikator z biblioteki OpenCV opiera się na cechach Haara dla twarzy ludzkiej obliczonych wcześniej i dostarczonych wraz z implementacją biblioteki. Więcej na temat tego klasyfikatora można znaleźć na stronie Wiki biblioteki OpenCV dotyczącej wykrywania twarzy [13].

4.5.5 Badanie i wnioski używania różnych metod

W celu zbadania zachowania poszczególnych klasyfikatorów wykonano testy dla wszystkich rozpoznawanych przez system gestów. Wyniki okazały się najlepsze dla klasyfikatora wektorów nośnych.

Testy wykonano przy warunkach dobranych do poprawnej, ale nie idealnej segmentacji skóry, czyli starano się uzyskać najbardziej zbliżone warunki do poprawnego działania systemu w trakcie użytkowania.

W tabelach 4.8-4.11 przedstawiono wyniki dla poszczególnych gestów. Najsłabiej klasyfikowany okazał się gest przedstawiający literę A czyli zaciśnięta pięść. W wyniku

Klasyfikator	Procent poprawnej klasyfikacji
Bayesa	99%
Najbliższego sąsiada	89%
Trzech najbliższych sąsiadów	93%
SVM (Neutralny; R)	99,3%
SVM (Neutralny; T)	0,3%
SVM (Neutralny; K)	30,3%

Tabela 4.8: Wyniki testów klasyfikacji gestu neutralnego

Klasyfikator	Procent poprawnej klasyfikacji
Bayesa	100%
Najbliższego sąsiada	90,6%
Trzech najbliższych sąsiadów	97,6%
SVM (Neutralny; R)	100%
SVM (T; R)	100%
SVM (A; R)	100%

Tabela 4.9: Wyniki testów klasyfikacji gestu R

Klasyfikator	Procent poprawnej klasyfikacji
Bayesa	100 %
Najbliższego sąsiada	84,6 %
Trzech najbliższych sąsiadów	82,9 %
SVM (T; R)	97,9 %
SVM (T; Neutralny)	0,9 %
SVM (T; A)	13,2 %

Tabela 4.10: Wyniki testów klasyfikacji gestu T

Klasyfikator	Procent poprawnej klasyfikacji
Bayesa	67,2 %
Najbliższego sąsiada	68,3%
Trzech najbliższych sąsiadów	68,9 %
SVM (A; R)	96,9 %
SVM (A; Neutralny)	0,9 %
SVM (A; T)	7,2 %

Tabela 4.11: Wyniki testów klasyfikacji gestu A

tych badań potwierdzone zostało zmapowany w systemie gestu A na najmniej użyteczną dla użytkownika operacja, o czym więcej w podrozdziale 4.7.

Najlepiej rozpoznawany przez system jest gest R. Ma to związek z jego zupełnie odmiennym, podłużnym kształtem, co skutkuje łatwo rozróżnialnymi niezmiennikami momentowymi zobacz 4.3.1.

Podstawowym wnioskiem z badania różnych metod rozpoznawania dłoni jest założenie, że znacznie lepiej użyć kilku różnych klasyfikatorów i połączyć ich wyniki, niż używać jednego nawet najlepszego klasyfikatora. Do podobnych wniosków doszli autorzy książki [32].

4.6 Detekcja i śledzenie ruchu

Oprócz rozpoznawania gestów ważnym elementem systemu jest detekcja ruchu. Może być ona rozwiązana na wiele różnych sposobów. Jednym z bardziej znanych i prostych w implementacji algorytmów jest algorytm badający różnicę pomiędzy dwoma kolejnymi klatkami sekwencji obrazu i analizie kierunku ruchu na podstawie tej różnicy. Został on opisany w m.in. [46] [7]. Jest to jednak algorytm niedokładny i niemożliwe jest jego zastosowanie, gdy „gubimy" obszar dłoni (np. poprzez nie znalezienie jej na obrazie). W pracy został zastosowany algorytm będący niejako sumą wielu pomysłów opisany poniżej.

4.6.1 Omówienie autorskiego algorytmu

W celu śledzenia ruchu dłoni został zaimplementowany autorski pomysł opierający się na algorytmie [4], będącym ulepszeniem procedury wyznaczania optycznego przepływu Lucasa Kanade.

Algorytm z pracy [4] korzysta z iteracyjnej postaci metody Lucasa Kanade. Koncep-

cja polega na kilkukrotnym wyznaczaniu przepływu optycznego dla pikseli na obrazach o różnych rozdzielczościach. Przed rozpoczęciem działania algorytmu Lucasa Kanade, oba obrazy sekwencji są kilkukrotnie zmniejszane. Następnie, w kolejnych krokach brane są obrazy od najmniejszego i liczony jest przepływ optyczny pikseli, dla których szukane jest przemieszczenie. Wyniki z mniejszych rozdzielczości są używane w kolejnych etapach przetwarzania, jako wartości wejściowe dla obliczeń na obrazach o większych rozdzielczościach. Ulepszenie to ma na celu lepszą detekcję dynamicznego ruchu.

Sam algorytm Lucasa Kanade opiera się na rozwiązywaniu równania przepływu optycznego pikseli przy założeniu, że jest on stały w najbliższym sąsiedztwie badanego piksela.

Implementacja algorytmu z artykułu [4] znajduje się w bibliotece OpenCv i pozwala na śledzenie punktu w sekwencji obrazów. W celu zaadoptowania go do śledzenia dłoni musiały zostać rozwiązane następujące problemy:

1. ustalenie punktów początkowych w obszarze dłoni;
2. wyeliminowanie błędów związanych z niewyszukaniem przesunięcia dla starego punktu;
3. zgubienie punktu z obszaru dłoni.

W celu rozwiązania powyższych problemów zdecydowano się na śledzenie dużej ilości pojedynczych punktów (100) i uśrednianie wartości przemieszczenia każdego z nich. Dodatkowo, jeżeli kilka punktów zostaje usuniętych lub zgubionych algorytm może działać nadal opierając się na wynikach zebranych z analizy pozostałych punktów.

Do rozwiązania problemu pierwszego użyte zostały wyniki rozpoznawania obszaru dłoni (zobacz podrozdział 4.4). W tym celu podczas poprawnego wykrycia dłoni tworzone są punkty o współrzędnych leżących wewnątrz obszaru, który można określić używając momentów przestrzennych [5]. Intuicyjną metodą wyznaczania środka jest obliczanie środka geometrycznego [26]. Dla określenia środka dłoni, jak również jej pola używa się poniższych wzorów:

$$Pole = m_{00} \tag{4.1}$$

$$x = m_{10}/m_{00} \tag{4.2}$$

$$y = m_{01}/m_{00} \tag{4.3}$$

Dane te służą do wyznaczenia prostokąta, we wnętrzu którego losowane są punkty, używane następnie do śledzenia ruchu (rysunki 4.11 i 4.12). Za każdym razem po poprawnej detekcji dłoni następuje korekcja nieprawidłowych punktów, opisana dalej.

Rysunek 4.11: Wyznaczanie środka ciężkości i ograniczającego rozmieszczenie punktów kontrolnych na podsawie konturu dłoni.

Rysunek 4.12: Wyznaczone punkty na tle dłoni.

W celu zminimalizowania wpływu wykonywania gestu na ruch dłoni wyżej opisany prostokąt jest wyznaczany w dolnej części dłoni. Dzięki temu gesty wykonywane głównie palcami mają znacznie mniejszy wpływ na ogólny ruch kursora.

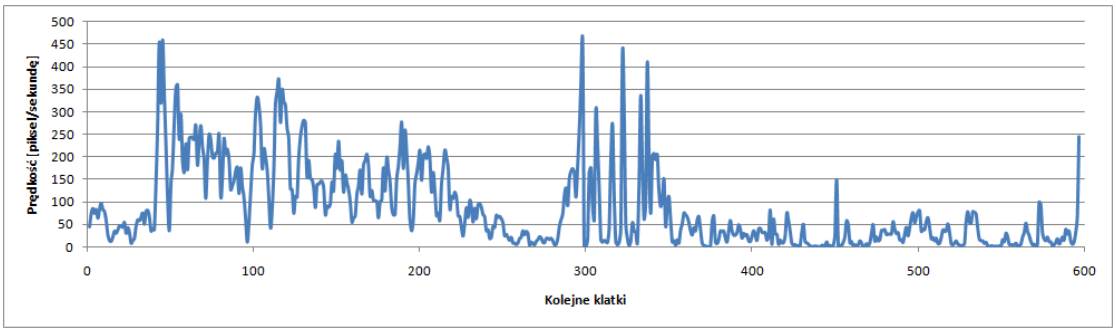
Algorytm zaimplementowany w bibliotece OpenCv w trakcie próby detekcji przemieszczenia zwraca również informacje o tym, czy udało się znaleźć nowy punkt odpowiadający staremu. Jeżeli próba taka zakończy się niepowodzeniem punkt jest usuwany i nie jest dalej brany pod uwagę. Rozwiązuje to problem opisany w punkcie drugim.

Ostatni problem jest rozwiązany poprzez wspomnianą wcześniej korekcje punktów podczas poprawnej identyfikacji dłoni. Każdy z punktów, który został wcześniej ustalony, przechodzi test czy zawiera się wewnątrz dłoni. Jeżeli nie, to na jego miejsce zostaje wylosowany punkt z wnętrza obszaru dłoni. Dodatkowo zawsze podczas poprawnej klasyfikacji zostają „uzupełnione" punkty usunięte wcześniej, w skutek nie odnalezienia im odpowiednika. W ten sposób punkty stale znajdują się wewnątrz dłoni.

Eliminacja ruchu podczas wykonywania gestów

W trakcie gdy dłoń jest otwarta i użytkownik nie wykonuje żadnych ruchów, poruszanie kursora jest płynne i gładkie. Jeżeli jednak w momencie gdy użytkownik zatrzyma rękę i wykona gest problemem stają się niewielkie przesunięcia punktów kontrolnych wywołujący niezamierzony i niepożądany ruch kursora.

Dla uniknięcia powyższego oprócz wyboru obszaru dłoni do losowania punktów opisanego wyżej, zaimplementowano algorytm redukcji chaotycznych ruchów. Polega on na obliczeniu ile punktów ma różny kierunek ruchu w stosunku do uśrednionego ruchu wszystkich punktów kontrolnych. W przypadku, gdy użytkownik wykonuje np.: zaciśnięcie dłoni punkty kontrolne zbiegają się do środka, co skutkuje różnymi wektorami kierunku ruchu. Jeżeli ilość punktów z innymi wektorami od wektora przekroczy zadany próg ruch nie jest wykonywany.



Wykres 4.13: Prędkość ruchu dłoni

Dzięki temu ulepszeniu możliwe jest wykonywanie gestów bez znaczącej zmiany pozycji kursora, dzięki czemu praca z systemem jest wygodniejsza i dokładniejsza.

Filtracja dolnoprzepustowa ruchu

Ruch dłoni może być wykonywany ze skończoną prędkością, dlatego też w systemie zastosowano odfiltrowywanie ruchu, który został zaklasyfikowany jako zbyt szybki. Ruch taki może nastąpić na przykład przy błędzie rozpoznania, lub podczas zakłóceń wniesionych w etapie akwizycji. W celu odfiltrowania zbyt szybkiego ruchu zastosowano filtr dolnoprzepustowy, który eliminuje ruch ze zbyt dużą prędkością.

W celu ustalenia wartości progowej dla prędkości wykonano testy z jaką prędkością porusza się dłoń przy normalnej pracy. Dla rozdzielczości 320 na 240 było to średnio 86 pikseli na sekundę, przy maksymalnym zarejestrowanym przemieszczeniu 460 pikseli na sekundę. Ostatecznie progowa prędkość została ustalona na 350 pikseli na sekundę. Na wykresie 4.13 przedstawiono wykres prędkości w zależności od czasu testowania programu.

Wnioski z badania algorytmu

Podsumowując algorytm ten sprawdza się bardzo dobrze, nawet jeżeli nie w każdej klatce sekwencji z kamery nastąpi wyszukanie i poprawna identyfikacja dłoni. Dzięki temu ruch myszki staje się płynniejszy i dużo dokładniejszy, niż gdyby użyć jedynie przemieszczenia środka ciężkości dłoni. Ustalając liczbę punktów kontrolnych do śledzenia na dość dużą wartość, uzyskujemy dobre rezultaty nawet po długim czasie nie odnalezienia dłoni na obrazie. Algorytm ten jest jednak bardzo wrażliwy nawet na małe ruchu dłoni przez co precyzja podczas wykonywania gestów jest stosunkowo niewielka. Rozpatrując jednak jego wady i zalety zdecydowano się na użycie tej metody w programie, jako bardzo zadowalającej dla typowych zastosowań.

4.7 Przełożenie wyników na gesty myszki

Gesty wykonywane za pomocą myszy zostały podzielone na dwie płaszczyzny. Pierwszą z nich jest ruch kursora myszy, a drugą bardziej rozbudowaną jest wykonywanie kliknięć, czyli przełożenie gestów dłoni na komunikaty systemowe symulujące klikanie przycisków.

4.7.1 Ruch kursora

Poruszana przed kamerą dłoń powoduje ruch kursora myszki. Przełożenie ruchu jest dokonywane relatywnie do zmiany położenia dłoni. Brana pod uwagę jest jedynie zmiana położenia w poziomie i pionie, a nie odległości od kamery. Dzięki takiemu rozwiązaniu nie jest wymuszane na użytkowniku trzymanie ręki w odpowiedniej odległości w stosunku do kamery.

W systemie zaimplementowano ustawianie przełożenia ruchu ręki na ruch kursora na ekranie. W trakcie działania programu użytkownik ma możliwość zwiększenia i zmniejszenia czułości w zależności od indywidualnych potrzeb. Więcej na temat kalibracji systemu można przeczytać w instrukcji obsługi programu (Dodatek A).

4.7.2 Gesty kliknięcia przycisków

Wykonanie kliknięć myszki odbywa się poprzez pokazywanie gestów. Mapowanie gestów na odpowiednie przyciski zostało opisane w podrozdziale 4.3. Poniżej opisano dokładnie sekwencje w jaki sposób następuje wysyłanie komunikatów do systemu.

Lewy przycisk myszy

W przypadku pokazania gestu R. Następuje wysłanie komunikatu wciśnięcia lewego przycisku myszy. Komunikat o puszczeniu przycisku następuje dopiero, gdy zostanie wykonany inny gest. Rozwiązanie takie umożliwia operacje przeciągania elementów czy np.: zaznaczania tekstu.

Środkowy przycisk myszy

Gest A, czyli zaciśnięta dłoń powoduje wciśnięcie i natychmiastowe puszczenie środkowego przycisku myszy. Odmienne zachowanie niż w przypadku przycisku lewego jest spowodowane rzadkim używaniem długiego wciśnięcia środkowego przycisku, a natychmiastowe puszczenie przyspiesza reakcje programów w systemie operacyjnym.

Prawy przycisk myszy

Wykonanie gestu T powoduje wysłanie do systemu komunikatu kliknięcia i natychmiastowego puszczenia prawego przycisku myszy. Podobnie jak w przypadku środkowego przycisku pokazanie gestu związanego z prawym przyciskiem powoduje wysyłanie dwóch komunikatów systemowych. Dzięki temu pojawianie się menu kontekstowych, związanych z prawym przyciskiem myszy, następuje szybciej, ponieważ nie trzeba pokazać innego gestu w celu puszczenia przycisku.

4.8 Testy

W trakcie powstawania aplikacji przeprowadzono wiele testów modułowych mających na celu dostosowanie każdego parametru w systemie dla uzyskania jak najlepszego wyniku końcowego rozpoznawania. Testy te były czynione przed ukończeniem programu i miały wpływ na ustalenie stałych wartości używanych w algorytmach.

Po zakończeniu modułu segmentacji i analizy zostały przeprowadzone testy końcowe. Pokazują one jakość działania w rzeczywistym otoczeniu. Szereg przeprowadzonych testów udowadnia wysoką jakość powstałego oprogramowania. Zostały one przeprowadzone w różnych warunkach, zarówno w dzień jak i w nocy. Testowanie końcowe polegało na pokazaniu gestu i sprawdzeniu czy system poprawnie go rozpoznawał.

Zastosowanie wielu algorytmów rozpoznawania skóry jest spowodowane faktem, iż nie można określić jednoznacznie, który algorytm segmentacji skóry jest najlepszy. W zależności od panujących warunków oświetleniowych, odmienne algorytmy dawały najlepsze wyniki (porównaj z podrozdziałem 4.4.5).

Testowanie jakości przełożenia ruchu ręką na ruch kursora odbywało się głównie empirycznie. Przebadano zachowanie programu dla ruchu wolnego, typowego i bardzo szybkiego. Dodatkowo przebadano zachowanie się programu gdy ręka wychodzi poza kadr kamery, a następnie wraca. Stwierdzono, że zachowanie systemu jest zgodne z intuicją i nie powoduje błędów.

Podczas wielu testów okazało się, że program działa błędnie gdy dłoń znajduje się blisko twarzy (występują błędne rozpoznania gestów), dlatego też zalecane jest pokazywanie gestów i ruch dłoni z dala od głowy użytkownika.

4.9 Wnioski

Stworzony program spełnia wymagania interfejsu człowiek-komputer. Za jego pomocą możliwa jest komunikacja człowieka z komputerem z użyciem jedynie kamery internetowej jako urządzenia wejściowego. Program działa w czasie rzeczywistym tzn. obliczenia

są wykonywane na bieżąco i decyzja jest podejmowana na podstawie aktualnie wykonywanych gestów i ruchu dłoni.

Aplikacja umożliwia płynne poruszanie kursora na ekranie oraz wykonywanie wystarczającej ilości gestów w celu interakcji z komputerem. Funkcjonalność programu odpowiada funkcjonalności myszy komputerowej.

Rozpoznawana przez system jest pojedyncza dłoń, dlatego też należy wyeliminować z wizji kamery wszystkie inne ruchome przedmioty o kolorze podobnym do skóry ludzkiej. System działa zdecydowanie lepiej gdy kamera jest statyczna. Jest to spowodowane użyciem funkcji filtrującej statyczne tło.

Precyzja ruchu myszką pozwala na nawigację w systemie, obsługę dokumentu tekstowego, przełączanie okien itp. Nie jest to jednak dokładność wymagana do obsługi programów graficznych. Średnio sprawdza się przy przeglądaniu Internetu (zbyt mała dokładność kliknięć). Ze względu na fakt, że ruch ręką jest zawsze przekładany na ruch kursora system nie sprawdza się również w grach - nie ma możliwości powrotu ręki bez wykonania ruchu kursora.

Mimo niskiej rozdzielczości pracy systemu (320 na 240 pikseli) osiągnięto dużą dokładność przemieszczenia kursora. Spowodowane jest to badaniem przemieszczeń wielu punktów przestrzeni, przez co dokładność ruchu jest zdecydowanie większa. System pozwala na pracę w rozdzielczość 1280 na 800 z dokładnością do jednego piksela. Praca na większych rozdzielczościach możliwa jest dzięki zmiennemu przełożeniu ruchu ręki na ruch kursora.

Wykonywanie gestów obarczone jest jednak niechcianym przemieszczeniem kursora podczas ruchu palców. Dlatego też klikanie jest dużo mniej dokładne aniżeli sam ruch. W systemie zastosowano wiele usprawnień mających poprawić tę niedogodność jednak nie eliminują one problemu całkowicie. W perspektywie dalszego rozwoju systemu powinno znaleźć się więc jego rozwiązanie.

System działa z dużą szybkością dzięki zastosowaniu niewymagających obliczeniowo algorytmów oraz niewielkiej rozdzielczości. Na komputerze z procesorem Intel Core 2 Duo 2 GHz system jest w stanie przetworzyć nawet do 60 obrazów na sekundę.

Rozdział 5

Podsumowanie

Celem niniejszej pracy było opracowanie interfejsu użytkownika z wykorzystaniem kamery internetowej. Dla uzyskania powyższego założenia został napisany i przebadany program umożliwiający interakcję człowieka z maszyną przez wykonywanie gestów i ruchów dłoni.

W implementacji systemu znalazły się algorytmy segmentacji koloru, wydzielania cech i ich analizy. W związku z dużymi wymaganiami stawianymi przed programem zastosowano równoległe rozwiązania umożliwiające pracę systemu w różnych warunkach oświetleniowych. W pewnych przypadkach znane i popularne algorytmy okazały się niewystarczające do spełnienia założonych wymagać, toteż zaproponowano pewne ulepszenia i zmiany, które w znaczący sposób przyczyniły się do uzyskania lepszych wyników.

W pracy znalazło się również omówienie systemów rozpoznawania obrazów w aspekcie interfejsów człowiek-komputer. Szczegółowo opisano również poszczególne etapy rozpoznawania skupiając się na zastosowaniach użytych w systemie.

Modułowa budowa powstałego programu umożliwia łatwy, późniejszy rozwój i rozszerzenie dowolnego etapu rozpoznawania, począwszy od segmentacji dłoni, a na decyzji systemu w postaci komunikatów systemowych skończywszy.

Projektowanie systemu przebiegało krokowo i można wydzielić kilka kolejnych etapów powstawania kompletnego programu. Poniżej przedstawiono kolejno moduły programu odpowiedzialne za poszczególne etapy rozpoznawania gestów.

Akwizycja obrazu i przetwarzanie wstępne to pierwszy etap. Pobierany jest obraz z kamery o rozdzielczości 320 na 240 pikseli z prędkością około 50 klatek na sekundę. Podczas testów rozdzielczość ta okazała się w zupełności wystarczająca do działania systemu. Pobrany obraz ulega przetwarzaniu morfologicznemu i trafia do kolejnego modułu programu.

Dzięki wykorzystaniu obrazu o niewielkiej rozdzielczości działanie programu możliwe

jest na obecnych komputerach. Osiągnięcie niewielkich wymagań systemowych przy wysokiej jakości działania systemu jest niewątpliwym sukcesem i pozwala na użytkowanie systemu przez szeroką gamę odbiorców.

Kolejny moduł odpowiedzialny jest za rozpoznawanie skóry i wydzielanie obszarów dłoni z obrazu. Zaimplementowano w nim szereg niezależnych algorytmów detekcji w różnych przestrzeniach barw. Ma to na celu umożliwienie korzystania z systemu w różnych warunkach oświetleniowych. Na stworzenie tego modułu poświęcono najwięcej czasu gdyż poprawna segmentacja skóry oznacza bezbłędne i stabilne działanie programu.

W następnym module zawarto rozpoznawanie gestu dłoni. Używane są w nim algorytmy z dziedziny uczenia maszynowego. W pracy zostały przebadane i zastosowane min. klasyfikator Bayesa, maszyna wektorów nośnych czy klasyfikator najbliższych sąsiadów. Przygotowane w trakcie pracy zbiory uczące pozwalają na rozpoznawanie 4 gestów, w tym gest neutralny.

Wykorzystanie autorskiego algorytmu i wielu pomocniczych rozwiązań pozwoliło na uzyskanie dobrze działającej detekcji ruchu. Dzięki innowacyjnym pomysłom ruch ręką powoduje płynny i bardzo dokładny ruch kursora. Dodatkowo zastosowano usprawnienie pozwalające na wykonywanie gestu przy niewielkim zaburzeniu w pozycji kursora na ekranie.

Ostatni moduł odpowiada za przełożenie uzyskanych wyników na komunikaty systemowe dotyczące myszki. Funkcjonalność programu jest równa trzy-przyciskowej myszce komputerowej przy użyciu jedynie kamery internetowej i ludzkiej dłoni. Zastosowano rozwiązanie pozwalające na działanie programu nawet po minimalizacji okien, dzięki czemu możliwa jest normalna praca z komputerem z wykorzystaniem zaimplementowanego systemu.

Ostateczne testy aplikacji dały zadowalający skutek i umożliwiły sterowanie komputerem bez użycia myszki.

Każdy z modułów jest stworzony w sposób umożliwiający jego rozbudowę. Kolejnymi etapami w rozwoju programu może być rozpoznawanie ruchu gestów trójwymiarowych oraz rozpoznawanie dłoni pomimo posiadania przez użytkownika krótkiego rękawka. Obecna budowa systemu umożliwia również dodanie dodatkowych gestów.

Bibliografia

[1] Crystal Muang Ahmed Elgammal and Dunxu Hu. Skin detection a short tutorial. *Department of Computer Science, Rutgers University, Piscataway, NJ, 08902, USA*.

[2] Nicu Sebe Alejandro Jaimes. Multimodal human–computer interaction: A survey. *IDIAP, Switzerland University of Amsterdam, The Netherlands*, 2007.

[3] Tamas Sziranyi Attila Licsara. User-adaptive hand gesture recognition system with interactive training. *Department of Image Processing and Neurocomputing, University of Veszpre´m, Egyetem u. 10, Hungary*, 2005.

[4] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm. *Intel Corporation Microprocessor Research Labs*.

[5] Hua Li Dong Xu. Geometric moment invariants. *Institute of Computing Technology, Chinese Academy of Sciences, Beijing China*, 2007.

[6] Rajeev Sharma Ediz Polat, Mohammed Yeasin. Robust tracking of human body parts for collaborative human computer interaction. *Computer Science and Engineering Department, 220 Pond Lab., Pennsylvania State University, University Park, PA 16802, USA*, 2002.

[7] Abdolhossein Sarrafzadeh Farhad Dadgostar. An adaptive real-time skin detector based on hue thresholding: A comparison on two motion tracking methods. *Institute of Information and Mathematical Sciences, Massey University, Albany, Private Bag 102 904, North Shore Mail Centre, Auckland, New Zealand*, 2005.

[8] Adrian Kaehler Gary Bradski. *Learning OpenCV*. O'Reilly Media, 2008.

[9] Younglae J. Bae-Hyun Seung Yang Ho-Sub Yoon, Jung Soh. Hand gesture recognition using combined features of location, angle and velocity. *Image Processing Div. / Computer and Software Technology Lab., ETRI 161, Kajung-Dong, Yusung-Ku,*

Taejon, 305-350, South Korea - Department of Computer Science / Korea Advanced Institute of Science and Technology, 373-1 Kusung-Dong, Yusung-Ku, Taejon, 305-701, South Korea, 2000.

[10] *http : //johnnylee.net/projects/wii*. Strona johnny'a lee autora projektów związanych z kontrolerem wii. WWW.

[11] *http : //opencv.willowgarage.com/wiki/*. Strona wiki bilbioteki opencv. WWW.

[12] *http : //opencv.willowgarage.com/wiki/ApplicationGSOC*2010. Strona na temat aplikacji współpracujących z biblioteką opencv. WWW.

[13] *http : //opencv.willowgarage.com/wiki/FaceDetection*. Strona wiki dotyczący wykrywania twarzy w bibliotece opencv. WWW.

[14] *http : //software.intel.com/en − us/articles/intel − integrated − performance − primitives − intel − ipp − open − source − computer − vision − library − opencvfaq*. Strona na temat technologii procesorowej intela związanych z biblioteką opencv. WWW.

[15] *http : //www.cognotics.com/opencv/docs/*1.0/*ref/opencvref_cvaux.htm*. Dokumentacja opencv - opis modułu cvaux. WWW.

[16] *http : //www.cognotics.com/opencv/docs/*1.0/*ref/opencvref_cv.htm*. Dokumentacja opencv - opis modułu cv. WWW.

[17] *http : //www.cognotics.com/opencv/docs/*1.0/*ref/opencvref_cxcore.htm*. Dokumentacja opencv - opis modułu cxcore. WWW.

[18] *http : //www.cognotics.com/opencv/docs/*1.0/*ref/opencvref_highgui.htm*. Dokumentacja opencv - opis modułu highgui. WWW.

[19] *http : //www.cognotics.com/opencv/docs/*1.0/*ref/opencvref_ml.htm*. Dokumentacja opencv - opis modułu ml. WWW.

[20] *http : //www.cs.iit.edu/˜agam/cs*512/*lect − notes/opencv − intro/opencvintro. html*. Wprowadzenie do programowania w opencv - dokumentacja. WWW.

[21] *http : //www.cs.iit.edu/˜agam/cs*512/*lect − notes/opencv − intro/opencvintro. html#SECTION*00023000000000000000. Dokumentacja opencv - konwencja nazewnictwa. WWW.

[22] *http : //www.emgu.com/wiki/index.php/Main_Page*. Oficjalna strona projektu emgu. WWW.

[23] *http : //www.nintendo.pl/konsola_wii.php*. Oficjalna strona nintendo poświęcona projektowi wii. WWW.

[24] *http : //www.opensource.org/licenses/bsdlicense.php*. Licencja bsd - opis. WWW.

[25] *http : //www.vicon.com/*. Oficjalna strona firmy vicon (vicon motion systems and peak performance inc). WWW.

[26] Mgr inż. Szymon Myśliński. Rozpoznawanie obrazów dłoni za pomocą gramatyk klasy etpl(k) w systemie wizyjnym analizy języka migowego. *Katedra Systemów Informatycznych UJ*, 2009.

[27] Jordi Gonzalez-Francisco J. Perales Javier Varona, Antoni Jaume-i-Capó. Toward natural interaction through visual recognition of body gestures in real-time. *University of the Balearic Islands, Department of Mathematics and Computer Science, Edificio Anselm Turmeda Campus UIB, Ctra. Valldemossa Palma de Mallorca, Spain*, 2009.

[28] Thomas Jean-Sebastien Pierrard. Skin detail analysis for face recognition. *University of Basel, Department of Computer Science, Bernoullistrasse 16, CH - 4056 Basel, Switzerland*, 2006.

[29] Michael J. Jones and James M. Rehg. Statistical color models with application to skin detection. *Cambridge Research Laboratory, Compaq Computer Corporation, One Cambridge Center, Cambridge.*

[30] Juan A. Gómez-Pulido-Juan M. Sánchez-Pérez Jose M. Chaves-González, Miguel A. Vega-Rodríguez. Detecting skin in face recognition systems: A color spaces study. *Univ. Extremadura, Dept. Technologies of Computers and Communications, Escuela Politécnica, Campus Universitario s/n, 10071, Cáceres, Spain*, 2009.

[31] Parag C. Pendharkar Judy Yorka. Human–computer interaction issues for mobile computing in a variable work contex. *Solutions, Hagerstown, USA*, 2004.

[32] Ludmila I. Kuncheva. *Combining Pattern Classifiers Methods and Algorithms*, chapter 3, 4. INC., PUBLICATION.

[33] LI Jiuxian XIA Siyu XIA LiangzhengFace. Face detection based on self-skin segmentation and wavelet support vector machine. *Department of Automatic Control Engineering, Southeast University, Nanjing, China*, 2006.

[34] David Gonzalez Ortega Jose Fernando Diez Higuera Mario Martinez-Zarzuela, Francisco Javier Diaz Pernas and Miriam Anton Rodiguez. Real time gpu-based fuzzy

art skin recognition. *Higher School of Telecommunications Engineering, University of Valladolid (Spain)*, 2007.

[35] Jurij F. Tasic Marko Tkalcic. Colour spaces - perceptual, historical and application background. *Faculty of electrical engineering Univesity of Ljubljana*, 2003.

[36] Joanna Marnik. The polish finger alphabet hand postures recognition using elastic graph matching. *Rzeszów University of Technology, ul. W. Pola 2, 35-235 Rzeszow.*

[37] Joanna Marnik. Rozpoznawanie znaków polskiego alfabetu palcowego. *Politechnika Rzeszowska, Katedra Informatyki i Automatyki*, 2003.

[38] Bernd Volpel Martin Kreutz and Herbert Janissen. Scale invariant image recognition based on higher order autocorrelation geatures. *UniversititsstraBe*, 1996.

[39] S. AramvithD N. Soontranon and T. H. Chalidabhongst. Face and hands localization and tracking for sign language recognition. *Department of Elecuical Engineering Chulalongkorn University Bangkok 10330 Thailand*, 2004.

[40] F. Safaei P. Premaratne and Q. Nguyen. Moment invariant based control system using hand gestures. *School of Electrical, CScale invariant image recognition Computer Telecommunications Engineering, The University of Wollongong, North Wollongong, 2522, NSW, Australia, Research School of Information Sciences and Engineering, Australian National University, Canberra, ACT, Australia*, 2006.

[41] Laurence Perron. What gestures to perform a collaborative storytelling? *FT R and D 2 Avenue Pierre-Marzin F- 22300 Lannion*, 2007.

[42] Vadim Pisarevsky. Introduction to opencv. *Intel Corporation, Software and Solutions Group China*, 2007.

[43] Naveen Aggarwal Pragati Garg and Sanjeev Sofat. Vision based hand gesture recognition. *World Academy of Science, Engineering and Technology*, 2009.

[44] Michał Krawczyk Prof. M. Domański, dodatkowo: Marcin Grochociński. Rozpoznawanie twarzy wg, standardu mpeg 7. Prezentacja.

[45] Przemysław Korohoda Ryszard Tadeusiewicz. *Komputerowa analiza i przetwarzanie obrazów.* Wydawnictwo Fundacji Postępu Telekomunikacji Kraków, 1977.

[46] Maciej Smiatacz. Automatyczna lokalizacja i śledzenie obiektów na obrazie. *Politechnika Gdańska*, 2005.

[47] Abdesselam Bouzerdoum Son Lam Phung and Douglas Chai. Skin segmentation using color and edge information. *School of Engineering and Mathematics, Edith Cowan University Perth, Western Australia, Australia*, 2003.

[48] A. Lanitis T.F. Cootes T. Ahmad, C.J. Taylor'. Tracking and recognizing hand gestures, using statistical shape models. *Department of Medical Biophysics. University of Manchester*, 1996.

[49] Volker Kr:uger Thomas B. Moeslund, Adrian Hilton. A survey of advances in vision-based human motion capture and analysis. *Laboratory of Computer Vision and Media Technology, Aalborg University, Denmark*, 2006.

[50] Filipe Tomaz, Tiago Candeias, and Hamid Shahbazkia. Improved automatic skin detection in color images. *Universidade do Algarve, FCT, Campus de Gambelas - 8000 Faro, Portugal*, 2003.

[51] Thomas S. Huang Vladimir I. Pavlovic, Rajeev Sharma. Visual interpretation of hand gesture for human computer interaction a review. *Department of Electrical and Computer Engineering, and The Beckman Institute for Advanced Science and Technology University of Illinois at Urbana-Champaign 405 N. Mathews Avenue, Urbana, IL 61801, USA*.

[52] Guangyou XU Hui ZHANG Yu HUANG, Yuanxin ZHU. Spatial temporal features by image registration and warping for dynamic gesture recognition. *Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, P R China*, 1998.

Dodatek A

Instrukcja obsługi programu

W celu używania programu wymagana jest znajomość jego funkcji i właściwości. Po uruchomieniu programu przed rozpoczęciem użytkowania konieczna jest kalibracja systemu opisana poniżej. Spis funkcji zaimplementowanych w systemie zamieszczono po rozdziale dotyczącym kalibracji.

Dla poprawnego użytkowania programu użytkownik powinien być ubrany w rzeczy z długim rękawem o kolorze niezbliżonym do koloru skóry np. czarna koszula. Twarz, jeżeli znajduje się w kadrze, powinna być zwrócona w kierunku kamery w celu poprawnej jej lokalizacji przez system. Oświetlenie powinno być jednostajne, a dłoń nie powinna być prześwietlona.

A.1 Automatyczna kalibracja systemu

Po uruchomieniu systemu pojawi się okno automatycznej kalibracji (rysunek A.1). W celu poprawnego działania systemu należy wykonać wszystkie kroki kalibracji. Na rysunkach A.8 - A.10 zamieszczono kolejne zrzuty ekranu dla poszczególnych etapów kalibracji wraz z opisem.

A.1.1 Etap pierwszy – pobranie statycznego tła

W tym etapie następuje pobranie statycznego tła do systemu. Zapisywane tło powinno być niezmienne przez cały okres działania aplikacji. W przypadku gdy tło ulegnie zmianie, lub diametralnie zmieni się oświetlenie możliwe jest ponowne pobranie tła poprzez naciśnięcie klawisza c na klawiaturze. Etap pierwszy przedstawiono na rysunku A.1

A.1.2 Etapy od drugiego do piątego – pokazywanie gestów

Etapy od drugiego do piątego służą do automatycznego wyboru funkcji rozpoznawania skóry. Przedstawiono je na rysunkach A.2, A.3, A.4 i A.5. Dłoń należy trzymać w kadrze kamery, jednak nie na tle twarzy. Dla ułatwienia, w dodatkowym oknie wyświetlana jest miniatura gestu który powinien być pokazywany w danym momencie. W trakcie pobierania informacji o gestach pokazane zostaje okno z informacją procentową o postępie rysunek A.7.

A.1.3 Zakończenie kalibracji

Po zakończeniu kalibracji wyświetlane jest okno z podpowiedziami o funkcjach programu (rysunek A.6). Naciśnięcie dowolnego przycisku powoduje przejście do programu.

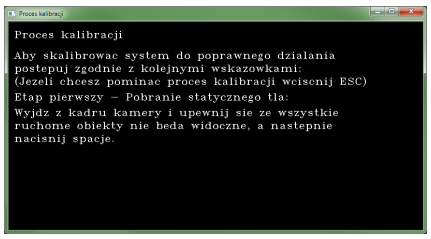
A.2 Ręczna kalibracja systemu

Po uruchomieniu programu można pominąć automatyczną kalibrację systemu (poprzez naciśnięcie Esc). W takim przypadku należy w pierwszej kolejności wybrać algorytm rozpoznawania skóry. W tym celu umieszczamy dłoń w obrębie widoku kamery i wybieramy algorytm za pomocą cyfr na klawiaturze (0-9). Dopasowanie algorytmu czyni się poprzez obserwowanie podglądu obrazu po segmentacji i zmianie algorytmu w taki sposób, by dłoń widoczna na ekranie była cała w kolorze białym, a jej krawędzie były gładkie i regularne. Rysunki A.11 i A.12.

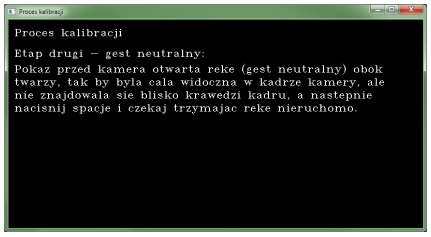
Kolejnym etapem jest zapamiętanie w systemie statycznego tła w celu eliminacji go z etapu wyszukiwania dłoni. W tym celu należy umieścić ręce i twarz poza kadrem kamery i nacisnąć przycisk *c* na klawiaturze.

Po usunięciu tła, można ponownie sprawdzić wybór algorytmu segmentacji skóry, w celu jak najlepszego dopasowania go do panujących warunków oświetleniowych, w których się znajdujemy.

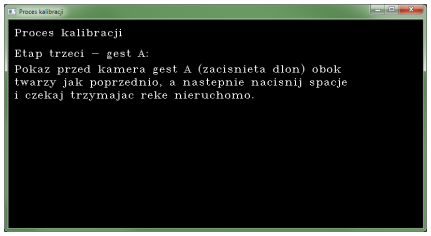
Jeżeli po etapie kalibracji dłoń widoczna na poglądzie w oknie po segmentacji posiada nieregularne krawędzie należy postarać się o lepsze oświetlenie pomieszczenia (niekoniecznie jaśniejsze). System nie będzie poprawnie działał w nocy bez oświetlenia, ale również wtedy, gdy dłoń przed kamerą będzie prześwietlona. Prześwietlenie dłoni może nastąpić, gdy źródło światła skierujemy bezpośrednio na nią, a oświetlenie tła będzie znikome. Optymalnym jest oświetlenie rozproszone, umiarkowane o naturalnym kolorze białym.



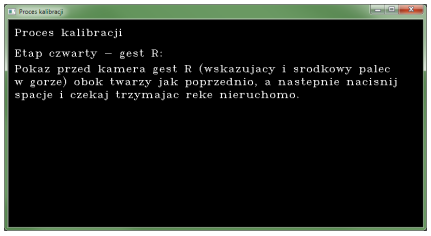
Rysunek A.1: Kalibracja etap pierwszy



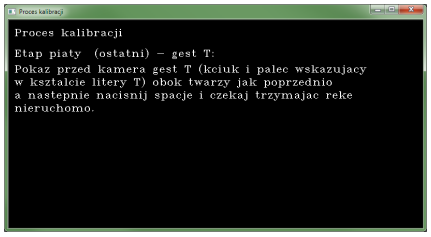
Rysunek A.2: Kalibracja etap drugi



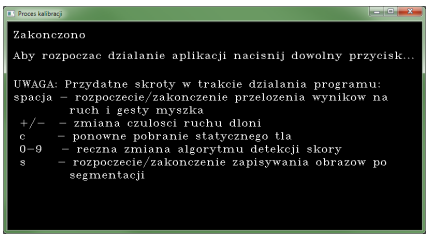
Rysunek A.3: Kalibracja etap trzeci



Rysunek A.4: Kalibracja etap czwarty

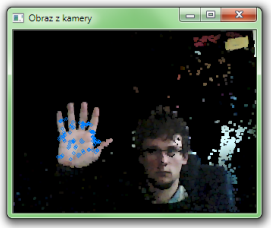


Rysunek A.5: Kalibracja etap piąty

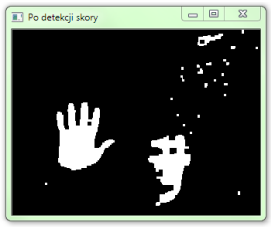


Rysunek A.6: Kalibracja zakończenie

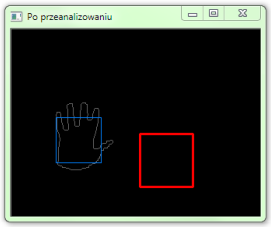
Rysunek A.7: Kalibracja postęp kalibracji



Rysunek A.8: Widok z kamery



Rysunek A.9: Widok po przetworzeniu



Rysunek A.10: Widok po przeanalizowaniu

Rysunek A.11: Prawidłowa segmentacja

Rysunek A.12: Nieprawidłowa segmentacja

A.3 Spis funkcji programu

W tabeli A.1 przedstawiono spis funkcji programu, wraz z krótkim opisem.

Program po uruchomieniu pokazuje na ekranie 4 okna. Jedno okno to konsola, w której można śledzić kiedy nastąpiło rozpoznanie jakiego gestu, a kolejne to obraz z kamery przed i po przetworzeniu wraz z zaznaczonymi dodatkowymi informacjami.

Poniżej przedstawiono każde z okien:

- Obraz z kamery po przepuszczeniu przez filtr usuwający tło rysukek A.8. W tym oknie mamy możliwość zaobserwowania obrazu z kamery, po przepuszczeniu go przez filtr, który usuwa piksele będące tłem. W przypadku, gdy do programu nie został jeszcze zapisany obraz tła (przez wciśnięcie c), będzie wyświetlony tu obraz bezpośrednio z kamery. Dodatkowo w oknie tym zaznaczono na niebiesko punkty służące do detekcji ruchu.

- Obraz po detekcji skóry rysunek A.9. Na tym oknie możemy zaobserwować, które piksele zostały uznane przez algorytm rozpoznawania za należące do skóry. Są one zaznaczone na biało. Pozostałe piksele są koloru czarnego.

- Obraz po przeanalizowaniu rysunek A.10. Ostatnie z okien przedstawia obrys wykrytej dłoni wraz z zaznaczeniem jej niebieskim kwadratem, oraz zaznaczoną na czerwono wykrytą twarz, jeżeli występuje ona w kadrze.

Nazwa	Skrót	Opis
Przełączanie mapowania na myszkę	spacja	Włączanie i wyłączanie przełożenia wyników programu na ruch i gesty myszy.
Zwiększenie czułości ruchu	+	Zwiększa czułość przełożenia ruchu ręką na ruch kursora. Po zwiększeniu czułości taki sam ruch dłoni powoduje większy ruch kursora myszki na ekranie.
Zmniejszenie czułości ruchu	-	Zmniejsza czułość przełożenia ruchu ręką na ruch kursora. Po zmniejszeniu czułości taki sam ruch dłoni powoduje mniejszy ruch kursora myszki na ekranie.
Ustawienie funkcji detekcji skóry	0-9	Zmienia funkcję rozpoznawania skóry w różnych przestrzeniach koloru. Odpowiednio: 0. Sam filtr wstępny 1. Funkcja HSV z filtrem wstępnym 2. Funkcja YCbCr z filtrem wstępnym 3. Funkcja RGB1 z filtrem wstępnym 4. Funkcja RGB2 z filtrem wstępnym 5. Funkcja RGB3 z filtrem wstępnym 6. Funkcja RGB4 z filtrem wstępnym 7. Funkcja RGB5 z filtrem wstępnym 8. Funkcja HSV bez filtru wstępnego 9. Funkcja YCbCr bez filtru wstępnego
Zapisanie tła w programie	c	Zapisuje w programie statyczne tło w celu eliminacji błędnej interpretacji części tła jako dłoni.
Zapisywanie obrazów po segmentacji	s	Rozpoczyna/Kończy zapisywanie obrazów binarnych po segmentacji na dysku w folderze uruchomienia programu. Mogą one potem służyć jako zbiór uczący.

Tabela A.1: Zbiór funkcji programu

Dodatek B

Omówienie biblioteki OpenCV

B.1 Część Opisowa

B.1.1 Opis biblioteki

OpenCV (Open Source Computer Vision) jest biblioteką programistyczną zawierającą zestaw funkcjonalności do przetwarzania grafiki komputerowej oraz uczenia maszynowego w czasie rzeczywistym. Biblioteka została zaimplementowana w języku C, jednak od wersji 2.0 udostępnia interfejs w języku Python oraz C++. Biblioteka posiada rozszerzenia umożliwiające kodowanie w innych językach programowania np. C#/Visual Basic o nazwie EmguCV [22].

Obecnie biblioteka jest dostępna w wersji 2.1 (dane na 17.07.2010) [11]. Jest to produkt zyskujący na popularności, o czym może świadczyć ponad dwa miliony ściągnięć ze strony projektu, oraz ponad czterdzieści tysięcy zarejestrowanych użytkowników na grupach dyskusyjnych dotyczących biblioteki. Ma zastosowanie w wielu dziedzinach, od sztuki graficznej i korekcji zdjęć, poprzez tworzenie wzorców danych i eksploracji wiedzy, do zaawansowanej robotyki. Implementacja zawiera ponad pięćset zoptymalizowanych algorytmów i około dziesięć razy tyle funkcji pomocniczych przydatnych przy ich używaniu [12].

Początkowo twórcą OpenCV był Intel, obecnie jej kod jest otwarty na zasadach licencji BSD (wolna do zastosowań deweloperskich i komercyjnych pod warunkiem informacji o autorach oryginalnego kodu i treści licencji) [24]. Dzięki udziałowi firmy Intel, w projekcie mamy dostępny interfejs do programowania procesorów Intela za pomocą IPP (Intel's Integrated Performance Primitives) [14].

OpenCV został zaprojektowany w celu stworzenia wspólnej infrastruktury dla systemów wizyjnych oraz przyspieszenia wykorzystania percepcji maszyny w produktach komercyjnych. Głównym naciskiem projektantów było użycie biblioteki w aplikacjach

czasu rzeczywistego. W celu zapewnienia jak najlepszej wydajności biblioteka wykorzystuje instrukcje MMX i SSE kiedy są dostępne. Obecnie trwają prace nad interfejsem CUDA. Używając SWIG, OpenCV posiada interfejsy do Matlab oraz Octave [12].

Wiele znanych firm korzysta z OpenCV do swoich zastosowań. Dla przykładu są to np.: Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota. Przykładowe przeznaczenia OpenCV to łączenie zdjęć z kamer ulicznych, wykrywanie włamań w Israelu lub monitorowania urządzeń kopalni w Chinach. Z godnych wspomnienia zastosowań należy przytoczyć system do wykrywania tonących ludzi stosowany w Europie, system kontroli etykiet na produktach w fabrykach na całym świecie czy system szybkiego wykrywania twarzy używany w Japonii [42].

Biblioteka OpenCV jest niezależna zarówno od systemu operacyjnego, menadżera okien jak i sprzętu. Poprawnie zainstalowany sprzęt (np.: kamera wideo) nie wymaga dodatkowej konfiguracji do współdziałania z biblioteką.

B.1.2 Opis biblioteki

Biblioteka OpenCV posiada następujące funkcjonalności [20]:

- Udostępnia funkcje do zarządzania obrazami i pamięcią. Alokacja pamięci potrzebnej do przetwarzania struktur odbywa się wewnątrz biblioteki i programista musi jedynie zadbać o wywołanie funkcji zwalniającej pamięć po obiekcie (obrazie, uchwycie do kamery), który przestaje już używać. Dostępne są również kopiowanie, konwersja, ustawianie właściwości obrazów oraz zapisywanie po zmianach.

- Obsługa wejścia wyjścia jest oparta na odbieraniu komunikatów systemowych i przetwarzaniu ich. Jako wejście mogą służyć zarówno obrazy i sekwencje wideo z dysku jak również informacje pobrane na bieżąco z kamery zainstalowanej w komputerze. Wyjście z programu może zostać zapisane, jako pliki obrazów, filmy jak również zostać wyświetlone na ekranie.

- Biblioteka obsługuje operacje na macierzach i wektorach. Implementuje zestaw podstawowych działań matematycznych jak np.: odwracanie macierzy, transpozycja, mnożenie, obliczanie wyznaczników, jak również bardziej zaawansowane algorytmy matematyczne, dla przykładu służące do obliczania równań liniowych.

- W OpenCV mamy dostępny zestaw podstawowych dynamicznych struktur danych. Są to listy, kolejki, zbiory drzewa i grafy. Zostały one napisane w sposób efektywny w celu szybkiego dostępu nawet na słabszym sprzęcie (np.: sterowniki, oprogramowanie robotów).

- W bibliotece został zaimplementowany niezależnych od środowiska graficzny interfejs użytkownika. Interfejs ten nie jest bogaty funkcjonalnie (np. brak przycisków) jednak w zupełności wystarczy do większości zastosowań biblioteki (wyświetlanie obrazów, wideo, obsługa suwaków). Za pomocą OpenCV możliwe jest również przechwycenie komunikatów od klawiatury i myszy.

- OpenCV oferuje szereg podstawowych operacji na obrazach. Będą to np. filtrowania (górno i dolnoprzepustowe), rozmazywanie, wyostrzanie, detekcja krawędzi, obliczanie mnożników, obliczanie dyskretnej transformaty Fouriera, obliczanie histogramu obrazka i wiele innych.

- Następną, jednak jedną z ważniejszych funkcjonalności jest analiza strukturalna. W bibliotece znajdziemy funkcje odpowiedzialne za przetwarzanie połączonych komponentów, dopasowywanie wzorców, wyznaczanie otoczek wypukłych, obliczanie momentów obiektów na obrazku, dopasowanie liniowe, Triangulacja Delone i inne.

- Za pomocą biblioteki OpenCV możemy również dokonać kalibracji kamery 3d, wyszukać i śledzić wzorce kalibracji oraz obsługiwać wiele kamer jednocześnie.

- Z bardziej zaawansowanych funkcjonalności należy wspomnieć o analizie ruchu. Implementacja dostarcza nam funkcje zarówno do wykrywania samego ruchu, jak i badania gradientu kierunku, czy śledzenia obiektów.

- W wykrywaniu obiektów z pomocą przyjdzie ukryty model Markova (Hidden Markov Model) czy metoda znajdowania rzeczywistych i urojonych części wartości wektorów własnych macierzy obiektów (Eigen-methods).

- Do zaznaczania oraz opisywania obrazów twórcy biblioteki stworzyli zestaw funkcji rysujących. Do dyspozycji są funkcje rysujące linie, figury geometryczne oraz wypisujące tekst.

Wraz z instalacją biblioteki zostają zainstalowane przykładowe programy prezentujące funkcjonalność biblioteki. Warto się z nimi zapoznać – dostępny jest również skomentowany kod źródłowy każdego programu.

Biblioteka OpenCV posiada szerokie wsparcie dla statystycznego uczenia maszynowego. Implementuje takie mechanizmy jak: drzewa decyzyjne, algorytm najbliższego sąsiedztwa, naiwna klasyfikacja Bayesa, maszyna wektorów nośnych, algorytm maksymalizacji wartości oczekiwanej, drzewa losowe, sieci neuronowe i inne [19].

B.1.3 Moduły

Biblioteka składa się z czterech podstawowych modułów. Są to cv, cvaux, cxcore, highgui. Dokowo biblioteka posiada moduł ml, który rozszerza funkcjonalność poprzednich czterech o funkcje uczenia maszynowego.

cv

Główny moduł, w którym znajdziemy funkcję do operacji na obrazach, analizy strukturalnej, wykrywania i śledzenia ruchu, rozpoznawania wzorców oraz kalibracji kamery i rekonstrukcji 3D. [16]

cvaux

Moduł zawierający funkcje pomocnicze oraz eksperymentalne. Znajdziemy w nim implementacje ukrytego modelu Markov'a, śledzenia trójwymiarowego, funkcjonalność Eigen-Objects (PCA), funkcje służące do morfingu [15].

cxcore

Moduł odpowiedzialny za struktury danych, oraz wsparcie algebry liniowej, znajdziemy w nim implementację tablic, funkcje matematyczne, dynamiczne struktury, funkcje rysujące, funkcjonalność zapewniającą zapis i odczyt danych z dysku (również RTTI) oraz funkcje systemowe [17].

Highgui

Moduł implementujący funkcje interfejsu użytkownika. W module zawarty kod odpowiedzialny za tworzenie, wyświetlanie, zmienianie ustawień okien; ładowanie, zapisywanie obrazków i wideo oraz funkcję konwertującą obrazki [18].

ml

Moduł odpowiedzialny za uczenie maszynowe. W tym module znajdziemy następujące algorytmy: normalną klasyfikację Bayes'a, klasyfikację najbliższych sąsiadów, maszynę wektorów nośnych, drzewa decyzyjne, metodę Boosting, drzewa losowe, algorytm maksymalizacji wartości oczekiwanej oraz sieci neuronowe [19].

B.2 Podejście programistyczne

W tej części zostały opisane wybrane podstawowe funkcjonalności biblioteki OpenCV na podstawie implementacji w języku C. Dodatkowo omówiona została konwencja nazewnictwa i przedstawiony został kod dwóch prostych programów (ładujący i wyświetlający obrazek, oraz przechwytujący i wyświetlający obraz z kamery).

B.2.1 Konwencja nazewnictwa

OpenCV zapewnia intuicyjną i łatwą do opanowania konwencję nazewnictwa [21]:

Funkcje

`cvActionTargetMod(...)`

gdzie:

- Action – akcja np.: `Set`, `Create`, `Load`, `Dispose`, `Dilate`, `Warp`, `Smooth`, `Integral`, `Init`, `Move`, `Find`
- Target – cel np.: `Contour`, `Image`, `Polygon`, `SubPix`, `Affine`, `Window`, `Capture`
- Mod – opcjonalny dodatkowy parameter, np.: typ argumentu

Przykłady: `cvDestroyWindow(..)`, `cvShowImage(..)`, `cvReleaseCapture(..)`, `cvWarpPerspective(..)`

Typy macierzy

`CV_<głębia>(S|U|F)C<liczba_kanałów>`

gdzie:

- `<głębia>` wyrażona liczba bitów na jedną komórkę macierzy (np.: piksel) np.: 8, 16, 24, 32
- `(S|U|F)` typ zmiennej
  - `S` całkowitoliczbowa ze znakiem
  - `U` całkowitoliczbowa bez znaku
  - `F` zmiennopozycyjnych
- `<liczba_kanałów>` liczba kanałów na jedną komórkę macierzy wyrażona w liczbie

Przykłady:

- `CV_8UC1` oznacza macierz liczb 8-bitowych, całkowitoliczbowych bez znaku z jednym kanałem
- `CV_32FC2` oznacza macierz 32-bitową, liczb zmiennopozycyjnych z dwoma kanałami

Typy obrazów

`IPL_DEPTH_<głębia>(S|U|F)`

`<głębia>` i `(S|U|F)` jak wyżej

Pliki nagłówkowe

Pliki nagłówkowe w bibliotece OpenCV odpowiadają modułom biblioteki omówionym wyżej.

Moduł	Plik nagłówkowy
cv	`<cv.h>`
cvaux	`<cvaux.h>`
highgui	`<highgui.h>`
cxcore	`<cxcore.h>`
ml	`<ml.h>`

Uwaga: plik `cxcore.h` jest załączany w `cv.h`, a więc nie ma konieczności załączania go, jeżeli załączyliśmy plik `cv.h`.

B.2.2 Przykładowe funkcje biblioteki

Załadowanie obrazu z pliku

```
IplImage* cvLoadImage(const char* filename,
                      int flags=CV_LOAD_IMAGE_COLOR)
```

Funka przyjmuje, jako parametr nazwę pliku, oraz złożenie następujących flag:

- `CV_LOAD_IMAGE_GRAYSCALE` obraz zostanie załadowany w skali szarości
- `CV_LOAD_IMAGE_COLOR` obraz zostanie załadowany w kolorze z 8-bitową głębią (chyba, że ustwaiono następną flagę)
- `CV_LOAD_IMAGE_ANYDEPTH` obraz zostanie z dowolną głębią w skali szarości
- `CV_LOAD_IMAGE_ANYCOLOR` obraz zostanie załadowany z dowolną głębią i kolorem

Zwalnianie obrazu

```
void cvReleaseImage(IplImage** image)
```

Funkcja przyjmuje, jako parametr wskaźnik do obrazu,

Kopiowanie obrazu

```
IplImage* cvCloneImage(const IplImage* image)
```

Funkcja przyjmuje, jako parametr wskaźnik do obrazu, który ma zostać zkopiowany. Funkcja wewnątrz alokuje potrzebną pamięć na nowy obraz.

Zapisanie obrazu na dysku

```
int cvSaveImage(const char* filename, const CvArr* image)
```

Funkcja przyjmuje, jako parametry nazwę pliki i uchwyt do obrazka, który ma zostać zapisany. Funkcja umożliwia zapis tylko 8bitowych obrazków z trzema kanałami w kolejność BGR.

Zmiana koloru obrazu

```
void cvCvtColor(const CvArr* src, CvArr* dst, int code)
```

Funkcja, jako pierwszy parametr przyjmuje 8, 16 lub 32 bitowy obrazek. Jako drugi obrazek docelowy, będący tego samego typu, a mogący różnić się jedynie liczbą kanałów (np. w konwersji z kolorowego obrazu na obraz w skali szarości), code jest kodem, w jaki sposób na nastąpić konwersja (z jakiej przestrzeni kolorów na jaką). Jest on konstruowany następująco:

`CV_<przestrzeń>2<przestrzeń>` gdzie `<przestrzeń>` może być następująca:

- `RGB` – kanały czerwony, zielony, niebieski (w takiej kolejności). Zakres R, G, B jest zależny od typu obrazka i tak odpowiednio 0..255 dla 8bitowego, 0..65535 dla 16bitowego, 0..1 dla zmiennopozycyjnego.
- `XYZ` – przestrzeń CIE XYZ
- `YCrCb`, `HSV`, `HLS` – przestrzenie jak wskazuje nazwa
- `Lab` – przestrzeń CIE L*a*b*
- `Lub` – przestrzeń CIE L*u*v*

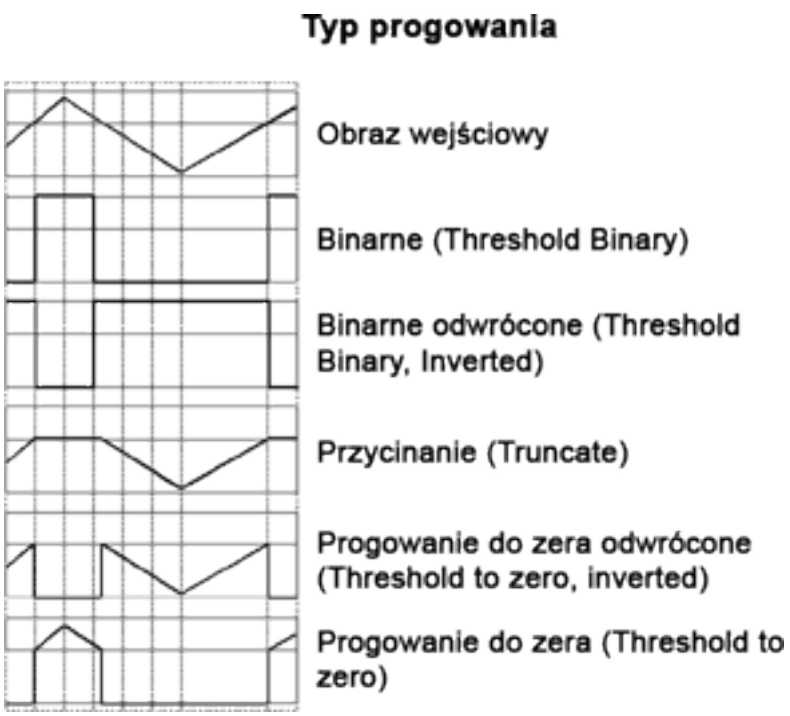
- `BayerBG` – służąca do konwersji z postaci surowej zdjęcia najczęściej dla zdjęć z matryc CCD lub CMOS. 'BG' może być zastąpione GB, GR, RG w zależności od matrycy.
- Gray – skala szarości

Progowanie

```
void cvThreshold(const CvArr* src, CvArr* dst, double threshold,
                 double max_value, int threshold_type)
```

Funkcja ta jako parametry wejściowe przyjmuje odpowiednio:

- `src` – obrazek źródłowy
- `dst` – obrazek docelowy
- `threshold` – poziom progowania
- `max_value` – wartość maksymalna używana do progowania binarnego i binarnego odwróconego
- `threshold_type` – typ progowania będący jednym z poniższych:



W celu zapoznania się z niektórymi, powyższymi poleceniami można przeglądnąć kod przykładowych programów podanych niżej.

B.2.3 Przykładowe programy

Program wyświetlający obrazek

```
/* Program wyświetlający obrazek o nazwie img.jpg,*
 * znajdujący się w tym samym katalogu co program.*
 * Autor: Damian Turczyński                        */
#include <cv.h>
#include <highgui.h>

int _tmain(int argc, char *argv[]){
    //Załaduj obraz
    IplImage* img = cvLoadImage("img.jpg");
    // Stwórz okno
    cvNamedWindow("Przykladowe okno", CV_WINDOW_AUTOSIZE);

    // Wyświetl okno
    cvShowImage("Przykladowe okno", img );
    // Czekaj na klawisz
    cvWaitKey(0);
    // Zwolnij obraz
    cvReleaseImage(&img );
    return 0;
}
```

Program przechwytujący obraz z kamery

```
/* Program wyświetlający przechwycony obraz z kamery.*
 * Autor: Damian Turczyński                           */
#include <cv.h>
#include <highgui.h>

int _tmain(int argc, char *argv[]){
    CvCapture* capture = cvCaptureFromCAM( CV_CAP_ANY );
    if( !capture ) {return -1;}

    //Stwórz okno do wyświetlania
    cvNamedWindow("Kamera_internetowa", CV_WINDOW_AUTOSIZE);
```

```
    //Pobierz w pętli obraz z kamery
    IplImage* frame = NULL;
    while(frame = cvQueryFrame( capture )){
        //Wyświetl pobrany obraz
        cvShowImage( "Kamera_internetowa", frame );
        if( cvWaitKey(10) == 27 ) break;
    }
    cvReleaseCapture(&capture);
    return 0;
}
```