Akademia Górniczo-Hutnicza

im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

PRACA MAGISTERSKA

TOMASZ HUCZEK

AUTOMATYCZNE ROZPOZNAWANIE PUNKTÓW KONTROLNYCH GŁOWY SŁUŻĄCYCH DO 3D MODELOWANIA JEJ ANATOMII I DYNAMIKI

PROMOTOR:

dr Adrian Horzyk

Kraków 2009

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W PRACY.

.....

PODPIS

AGH

University of Science and Technology in Krakow

Faculty of Electrical Engineering, Automatics, Computer Science and Electronics

MASTER OF SCIENCE THESIS

TOMASZ HUCZEK

AUTOMATIC HEAD CONTROL POINT RECOGNITION USED TO THE 3D MODELLING OF ITS ANATOMY AND DYNAMICS

SUPERVISOR:

Adrian Horzyk Ph.D

Krakow 2009

Serdecznie dziękuję promotorowi
doktorowi Adrianowi Horzykowi za pomoc w podjęciu interesującej mnie tematyki pracy oraz czas poświęcony na dyskusje nad algorytmami podczas jej realizacji.

Spis treści

Spis rysunków

1. Wstęp

Celem niniejszej pracy było wykonanie systemu do analizy ludzkiej głowy, w szczególności twarzy, tak aby możliwe było stworzenie trójwymiarowego modelu odwzorowującego jej anatomię oraz dynamikę. Zamiarem było napisanie aplikacji, która pozwalałaby w czasie rzeczywistym analizować twarz osoby siedzącej przed kamerą internetową. Odtworzony obraz trójwymiarowy mógłby zostać użyty do prezentacji naszego wizerunku w Internecie, wykorzystany w multimedialnej grze bądź odwzorowywać nasze reakcje, emocje oraz mimikę twarzy podczas czatu. System z założenia nie wymaga od użytkownika żadnych ustawień ani kalibracji. Od momentu uruchomienia programu obraz wideo jest analizowany i przetwarzany.

Techniki opisane w pracy mogą okazać się pomocne w przypadku identyfikacji zaginionych osób. Często jedynym materiałem, jakim dysponuje policja są zdjęcia osób zaginionych. Na tej podstawie można wykorzystać opisane algorytmy do stworzenia trójwymiarowego modelu postaci, która jest poszukiwana. Dodając do tego informacje o tym, w jaki sposób dana osoba była ubrana, jakie przedmioty miała przy sobie w momencie gdy widziano ją po raz ostatni, jesteśmy w stanie stworzyć wysoce prawdopodobny model osoby poszukiwanej. Techniki takie mogą być bardzo pomocne dla służb specjalnych. Dodatkowo można wyobrazić sobie sytuację, w której tak stworzony model umieszczamy w Internecie. Dowolny użytkownik może dokładnie obejrzeć model postaci, co na pewno pozwoli na lepsze utrwalenie cech danej osoby niż pojedyncze statyczne zdjęcie.

Modelowanie trójwymiarowe twarzy służyć więc może nie tylko dla zabawy, ale być bardzo cennym narzędziem wykorzystywanym do bardzo poważnych celów.

Praca została podzielona na kilka rozdziałów, w których kolejno zostaną opisane metody wykorzystywane w dziedzinie rozpoznawania obrazów i detekcji twarzy. Rozdział pierwszy wraz ze wstępem wprowadza w tematykę pracy oraz przedstawia jej główny cel.

Rozdział drugi dotyczy algorytmów zastosowanych przy tworzeniu pracy oraz innych obecnie powszechnie wykorzystywanych. Zawiera opis zbioru metod wykorzystanych w ramach projektu, a także tych, które nie zostały zaimplementowanie i użyte, ale uznane zostały za ciekawe i dające zadowalające rezultaty.

Rozdział trzeci przedstawia istniejące rozwiązania, sposoby ich funkcjonowania oraz algorytmy w nich zastosowane. Podjęto próbę porównania ich z metodami stosowanymi w pracy, a także przedstawiono możliwości, do jakich można dane metody wykorzystać. Wrozdziale tym podkreślono również wady przedstawionych tam zastosowań.

Rozdział czwarty opisuje, w jaki sposób działa aplikacja napisana w ramach pracy dyplomowej. W części tej zawarty jest szczegółowy opis i kolejne etapy działania programu wraz z wyszczególnieniem zastosowanych rozwiązań. Rozdział ten jest wzbogacony licznymi ilustracjami prezentującymi wygląd aplikacji, przebieg algorytmów oraz rezultaty, jakie udało się uzyskać.

Rozdział piąty to próba przetestowania stworzonego systemu. Algorytmy poddane zostały sprawdzianowi skuteczności w różnych warunkach. Materiały wideo, które służyły za dane wejściowe dla algorytmów nagrane zostały w różnych warunkach, a osoby przedstawione na obrazach ujęte zostały w różny sposób, tak by ocenić działanie algorytmu w jak najszerszej skali. Rozdział ten porusza również temat możliwych udoskonaleń danego rozwiązania a także propozycje, w jakim kierunku można projekt dalej rozwijać.

Rozdział szósty to zakończenie oraz próba podsumowania tego, co udało się wykonać w ramach pracy dyplomowej.

1.1. Postawienie problemu

Celem pracy magisterskiej jest stworzenie systemu detekcji oraz analizy twarzy, tak aby możliwe było jej zamodelowanie trójwymiarowe. Trójwymiarowy model ma odwzorować anatomię twarzy oraz jej dynamikę.

1.2. Cele

Głównym założeniem pracy jest stworzenie systemu działającego w czasie rzeczywistym oraz w sposób automatyczny. Na rynku istnieje cała gama rozwiązań modelujących trójwymiarowe twarze, lecz odtworzenie cech trwa czasami kilkanaście minut, co przekreśla dane rozwiązanie w dziedzinie algorytmów działających w czasie rzeczywistym. Znaczna część rozwiązań wymaga ręcznego wyznaczania punktów charakterystycznych twarzy, co z kolei uniemożliwia zastosowanie ich do automatycznych procesów.

Aplikacja będąca celem pracy magisterskiej działa na dwóch niezależnych płaszczyznach komunikujących się ze sobą. Jedna odpowiada za analizę twarzy i wykrywanie punktów kontrolnych, druga zaś jest odpowiedzialna za tworzenie trójwymiarowego modelu na podstawie otrzymanych danych z procesu analizy.

Aby wyżej wymienione wymagania zostały spełnione należy poszukiwać algorytmów szybkich, wykorzystujących jak najmniej zasobów procesora. Analiza twarzy jest zagadnieniem dość skomplikowanym, więc decydując się na takie rozwiązanie musi zostać poniesiony koszt, jakim jest niedokładność. Szczegółowa analiza twarzy oparta o skomplikowane i czasochłonne algorytmy nie nadaje się do rozwiązań stosowanych w aplikacjach czasu rzeczywistego.

Druga część aplikacji zajmująca się modelowaniem trójwymiarowym również ma zadanie dość skomplikowane. Zadaniem tym jest animacja modelu twarzy – a właściwie głowy – składającej się z kilkuset wierzchołków podczas jednoczesnej analizy twarzy poprzez drugi moduł. W obecnych komputerach klasy PC karty graficzne potrafią odciążyć procesor z czasochłonnych obliczeń graficznych, więc praca częściowo się rozkłada na podzespoły CPU[1] oraz GPU[2] pozwalając aby aplikacja działała wydajniej.

Jednym z problemów, z jakimi należy się zmierzyć, to duży wpływ warunków otoczenia na trudność wstępnej analizy obrazu. Warunki otoczenia, czyli oświetlenie, jakość obrazu zależna od urządzenia wejściowego pobierającego obraz wideo, zakłócenia, ale także odmienność cech wyglądu różnych ludzi, rasy, karnacje i wiele innych.

Do innych problemów należą takie sytuacje jak: nieprawidłowe zlokalizowanie twarzy na obrazie poprzez wpływ innych, często ruchomych obiektów – przechodząca osoba w tle patrząca w stronę kamery, obraz wiszący na ścianie, a także ruch kamery i ruch samej osoby, której twarz jest analizowana.

System tworzony w ramach pracy magisterskiej powinien brać pod uwagę wszystkie z wymienionych wyżej czynników oraz w jak najlepszym stopniu spróbować zlikwidować wszystkie problemy i utrudnienia z nimi związane.

[1] *Central Processing Unit* - procesor komputera

[2] *Graphics Processing Unit* - procesor karty graficznej

2. Metody wykorzystane w pracy zastosowane w rozwiązaniu postawionego problemu

2.1. Rys historyczny

2.1.1. Rozpoznawanie obrazów

Rozpoznawanie obrazów sprowadza się do wykorzystania komputerów w analizie cyfrowych obrazów w sposób zbliżony do tego, w jaki rozpoznaje je człowiek za pomocą swojej inteligencji. Gałęzią informatyki, która zajmuje się tego typu zagadnieniami jest *sztuczna inteligencja*. Zgodnie z [3] obszar tej dziedziny ma bardzo słabo wytyczone granice, gdyż samo pojęcie sztucznej inteligencji jest trudne do jednoznacznego zdefiniowania. Rozpoznawanie obrazów jest jednym z głównych tematów poruszanych w ramach tej dziedziny.

2.1.2. Analiza i detekcja twarzy

Pojęcie analizy i detekcji twarzy jest pojęciem stosunkowo młodym. Rozpowszechnione zostało dzięki komputeryzacji, technikom cyfrowym, łatwym i powszechnym dostępem do urządzeń typu aparaty cyfrowe, kamery oraz kamery internetowe.

Analiza oraz detekcja twarzy jest bardzo ważnym zagadnieniem systemów bezpieczeństwa. Biometria, czyli nauka zajmująca się badaniem zmienności populacji organizmów analizuje cechy fizyczne organizmów takie jak siatkówka oka, linie papilarne palców, ale również bardzo ważnym aspektem jest analiza oraz detekcja twarzy. Nauka ta pozwala na wykorzystanie technik do kontroli dostępu do chronionych pomieszczeń lub autoryzacji użytkowników korzystających z określonych danych, programów czy urządzeń.

Wyszukiwanie twarzy na obrazach jest dzisiaj bardzo powszechne i spotykane w bardzo wielu codziennych sytuacjach. Przykładem może być witryna *Google Images*, która analizuje indeksowane obrazy wyszukiwarki Google pod kątem występowania na nich twarzy i po włączeniu odpowiedniej opcji zwraca tylko te, na których zostały jakieś znalezione. Temat szerzej opisany jest w [4]. Z kolei w większości aparatów cyfrowych znaleźć można funkcję pozwalającą automatycznie ustawić ostrość w rejonie,

w którym znajduje się twarz. Tak samo kamerki internetowe dostarczane są z odpowiednim oprogramowaniem pozwalającym zlokalizować twarz na pozyskiwanym obrazie.

2.2. Podstawy przetwarzania obrazów

Aby móc rozpocząć przetwarzać obrazy należy zastanowić się jak są one przechowywane na urządzeniach cyfrowych. Obraz jest cyfrowym sposobem na zapisanie rzeczywistości. Ponieważ urządzenia cyfrowe mają swoje ograniczenia, nie możemy odwzorować rzeczywistości w sposób nieograniczony.

Pierwszym elementem jest kolor, który należy dyskretyzować i opisać cyframi. Aby to zrobić musimy jednoznacznie wyznaczyć zakres kolorów, jaki chcemy przedstawić na obrazie.

Innym ograniczeniem jest fakt, że na obrazie możemy zapisać obraz płaski, nie przestrzenny jak to jest w rzeczywistości. Do reprezentacji obrazu używa się najczęściej siatki kwadratowej, co ułatwia operacje graficzne i przechowywanie obrazu w pamięci komputera. Siatka obrazu (inaczej raster) przechowuje obraz o ograniczonej wielkości. O ilości elementów obrazu, jakie może on przechować decyduje jego rozdzielczość, która zazwyczaj ma wartość równą iloczynowi ilości elementów w poziomie oraz w pionie obrazu.

Elementami rastra są piksele, które mogą przyjmować wartości. Zakres tych wartości zależy od ilości bitów, jaka przypada na dany piksel. W zależności od potrzeb obraz może być binarny (2 bity na piksel), bądź kolorowy (do 32 bitów na piksel). Obrazy w skali szarości zazwyczaj są ośmiobitowe – wartość 0 oznacza kolor czarny, wartość 255 kolor biały, natomiast wartości pośrednie wyznaczają odpowiednie odcienie szarości rozpięte pomiędzy czarnym i białym.

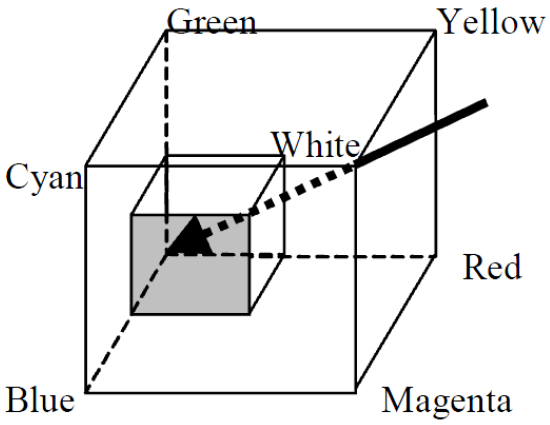
2.2.1. Przestrzenie barw

W technice cyfrowej obrazy przechowywane są w przeróżny sposób. Podstawową cechą obrazów jest kolor występujący na nich, będący nieodłączną ich częścią i cechujący się bardzo dużą skalą zróżnicowania. Należało więc w jakiś sposób zapisać informacje o kolorach występujących na obrazach. Podejść jest wiele i zależnie od potrzeb stosuje się różne rozwiązania.

RGB

RGB jest jednym z modeli przestrzeni barw, który opisany jest współrzędnymi R, G oraz B. Nazwy składowych pochodzą od angielskich nazw kolorów, z których składa się ten model – **R**ed, **G**reen i **B**lue, czyli czerwony, zielony oraz niebieski.

Model RGB pierwotnie stosowany był w technice analogowej, jednak obecnie jest również bardzo często stosowany także w technice cyfrowej. Na tym modelu kolorów oparte są współczesne aparaty fotograficzne, telewizory, monitory.



Rysunek 2.1: Model barwny RGB (*obrazek pochodzi z [1]*).

W informatyce RGB jest najczęściej stosowanym modelem kolorów. Występuje w większości plików graficznych, takich jak BMP, JPEG i innych. Zazwyczaj kolor zapisany jest na 24 bitach, czyli każdej składowej przypada 8 bitów. Z połączenia trzech barw składowych można uzyskać szeroką paletę kolorów, jednakże ta sama wartość może być różnie odwzorowana na różnych urządzeniach. Przykładowo na kiepskiej jakości matrycy monitora LCD kolory będą gorzej odwzorowane niż na monitorze lepszej jakości.

HSV

Model barwny HSV (ang. *Hue Saturation Value*) został zaproponowany w 1978 roku przez Alveya Raya Smitha. Według sposobu, w jaki widzi ludzkie oko wszelkie barwy wywodzą się ze światła białego, gdzie część widma zostaje wchłonięta, a część odbita od oświetlanych przedmiotów. Model HSV w taki właśnie sposób stara się opisać barwy.

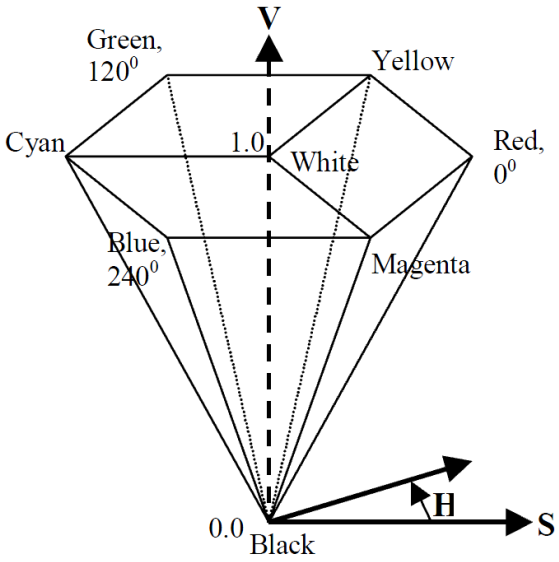
Składowe koloru w modelu HSV to litery wchodzące w skład nazwy, które pochodzą od: **H** – barwa światła (ang. *Hue*) wyrażona kątem na kole barw przyjmująca wartości od $0\,^\circ$ do $360\,^\circ$. Model można przedstawić jako stożek, którego podstawą jest koło barw – rysunek 2.2.

Składowa **S** to nasycenie koloru (ang. Saturation). Wyznacza ona długość promienia podstawy. Składowa **V** – (ang. Value) opisuje moc światła białego i na rysunku przedstawiona jest jako wysokość stożka.

2.2.2. Algorytmy przetwarzania obrazów

Algorytmów przetwarzania obrazów jest bardzo wiele. Część z nich została podzielona na różne grupy. Korzystając z [2] metody przekształceń można podzielić na:

- geometryczne,



Rysunek 2.2: Model barwny HSV (*obrazek pochodzi z [1]*).

- punktowe (bezkontekstowe),
- kontekstowe,
- widmowe (wykorzystujące transformatę Fouriera),
- morfologiczne.

W pierwszej kolejności zostaną omówione algorytmy, które zostały wykorzystane w pracy – głównie metody należące do grupy przekształceń punktowych, kontekstowych oraz morfologicznych.

Binaryzacja

Binaryzacja jest jedną z podstawowych operacji punktowego przetwarzania obrazu. Jak nazwa wskazuje polega na konwersji obrazu do postaci binarnej, czyli w przypadku obrazów polega na konwersji pikseli w skali szarości do pikseli o dwóch kolorach. Jest wiele metod na przeprowadzanie binaryzacji. Na podstawie [2] można wymienić najczęściej spotykane metody:

- **binaryzacja z dolnym progiem**

$$L'(x,y) = \begin{cases} 0 & \text{dla} \quad L(x,y) \leq a \\ 1 & \text{dla} \quad L(x,y) > a \end{cases} \tag{2.1}$$

gdzie:

s - wartość sąsiadujących punktów.

$L(x, y)$ - natężenie punktu obrazu źródłowego (wartości z zakresu $0 \ldots 2^B - 1$),

$L'(x, y)$ - wartość w punkcie obrazu wynikowego (wartości 1 lub 0),

a - próg binaryzacji.

- **binaryzacja z górnym progiem**

$$L'(x,y) = \begin{cases} 0 & \text{dla} \quad L(x,y) \geq a \\ 1 & \text{dla} \quad L(x,y) < a \end{cases} \tag{2.2}$$

- **binaryzacja z podwójnym ograniczeniem**

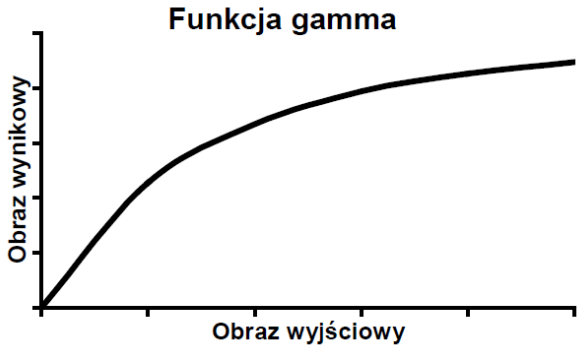
$$L'(x,y) = \begin{cases} 0 & \text{dla} \quad L(x,y) \leq a_1 \\ 1 & \text{dla} \quad a_1 < L(x,y) \leq a_2 \\ 0 & \text{dla} \quad L(x,y) > a_2 \end{cases} \tag{2.3}$$

gdzie:

a_1**,** a_2 - progi binaryzacji, $a_1 < a_2$.

Korekcja gamma

Korekcja gamma jest przekształceniem, które stosuje się do polepszenia jakości obrazu. Celem modulacji gamma jest redukcja nadmiernego kontrastu obrazu. Do tego celu wykorzystujemy funkcję postaci: $x \rightarrow x^\gamma$, gdzie γ to stały wykładnik będący zazwyczaj liczbą naturalną. Na rysunku 2.3 przedstawiono wykres funkcji gamma.

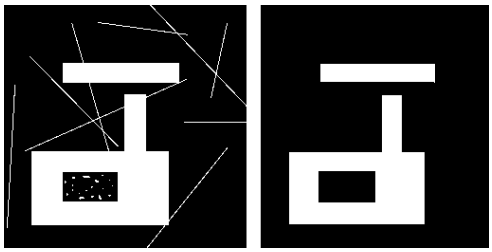


Rysunek 2.3: Funkcja gamma. (*obrazek pochodzi z [2]*).

Erozja

Erozja należy do przekształceń z grupy morfologicznych. Operuje ona na obrazach binarnych, więc jeżeli przetwarzamy obraz kolorowy bądź w skali szarości, należy wstępnie zastosować operację binaryzacji. Filtr ten polega na usunięciu wszystkich punktów obrazu o wartości 1, które posiadają przynajmniej jeden sąsiedni piksel o wartości 0. Erozja jest filtrem minimalnym, czyli takim operatorem, który każdemu pikselowi przypisuje minimum z wartości pikseli sąsiednich.

Główną właściwością erozji jest zdolność eliminacji drobnych szczegółów oraz wygładzania brzegu figury znajdującej się na obrazie.



Rysunek 2.4: Zastosowanie filtru erozji. Po lewej obraz źródłowy, po prawej rezultat zastosowania filtru.

Na rysunku 2.4 przedstawiono efekt działania algorytmu. Można zauważyć, że cienkie linie i drobne elementy i punkty zostały usunięte z obrazu wejściowego.

Wyrównanie histogramu

Aby zdefiniować pojęcie wyrównania histogramu należy zdefiniować najpierw pojęcie samego histogramu. Histogram obrazu jest operacją pozwalająca w pewien sposób scharakteryzować obraz. Histogram można przedstawić jako funkcję w następujący sposób:

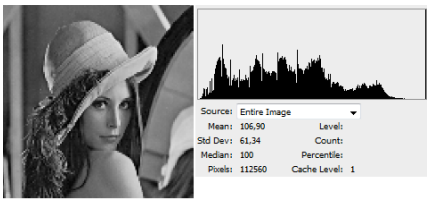
$$h(i) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} p(i/(x,y)) \quad i = 0, 1, \ldots, 2^B - 1; \tag{2.4}$$

gdzie:

$$p(i/(x,y)) = \begin{cases} 1 & \text{gdy} \quad L(x,y) = i, \\ 0 & \text{w przeciwnym przypadku.} \end{cases} \tag{2.5}$$

Funkcja $h(i)$ to nic innego jak opis ilości pikseli o danym natężeniu i występujących w obrazie. Histogram zazwyczaj przedstawiany jest w postaci wykresu, gdzie na osi pionowej zaznaczona jest ilość pikseli, a na osi poziomej natężenie danego piksela, co przedstawiono na rysunku 2.5.

Cytując [2]:



Rysunek 2.5: Obraz wejściowy oraz wykres prezentujący jego histogram.

"Wyrównanie histogramu polega na takim przekształceniu jasności poszczególnych punktów obrazu, aby ilość punktów o jasności leżącej w każdym z równych przedziałów histogramu była (w przybliżeniu) taka sama."

Operację wyrównania histogramu można przedstawić jako zwiększenie różnicy jasności pomiędzy takimi pikselami obrazu, które mają często występujące duże natężenia. Jeżeli założymy, że dla (i, j) takich, że spełniony jest warunek $h(i) > 0$ i $h(j) > 0$ oraz $h(i) = 0$ dla wszystkich $m < i < n$, to wtedy należy przemieścić m oraz n tak, aby minimalizować wartość Q:

$$Q = \left| \frac{\sum_{i=0}^{2^B-1} h(i)}{2^B - 1} - \frac{h(m)}{n - m} \right| \tag{2.6}$$

Konwolucja

Konwolucja (również nazywana splotem) należy do grupy filtrów liniowych, kontekstowych. Podstawowa definicja konwolucji zgodnie z [2] ma postać:

$$g(x) = (f \times h)(x) = \int_{-\infty}^{\infty} f(x - t)h(t)dx \tag{2.7}$$

gdzie: f, h to splatane funkcje.

W analizie komputerowej rozpatrujemy funkcje postaci $L(x, y)$, której dziedzina jest dyskretna oraz dwuwymiarowa. Upraszczając równanie 2.7 dla przypadku dyskretnego otrzymujemy:

$$L'(x, y) = (w \times L)(x, h) = \sum_{i,j \in K} L(x - i, y - j)w(i, j) \tag{2.8}$$

Filtr definiuje się jako tablice współczynników $w(i, j)$.

Konwolucja może spowodować wyjście poza zakres intensywności obrazu, dlatego też wprowadza się czynność normalizacji. Dla współczynników spełniających warunek $w(i, j) \geq 0$ stosuje się wzór:

$$L'(x, y) = \frac{1}{\sum_{i,j \in K} w(i, j)} \sum_{i,j \in K} L(x - i, y - j)w(i, j) \tag{2.9}$$

Jeżeli wartości współczynników mogą przyjmować również wartości ujemne, należy dodatkowo uwzględnić wartości maksymalne oraz minimalne przetworzonego obrazu, tak by znaleźć się w zakresie intensywności obrazu:

$$L''(x,y) = \frac{L'(x,y) - minL'(x,y)}{maxL'(x,y) - minL'(x,y)} 2^B \tag{2.10}$$

K jest otoczeniem punktu (x, y) i najczęściej przyjmuje się go jako okno wielkości 3×3. Wtedy tablica współczynników wygląda następująco:

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \tag{2.11}$$

Dla wyżej zaproponowanego przypadku funkcja realizująca algorytm konwolucji może być przedstawiona w ten sposób:

$$\begin{aligned} L'(x,y) = {} & w_1 L(x-1, y-1) + w_2 L(x-1, y) + w_3 L(x-1, y+1) \\ & + w_4 L(x, y-1) + w_5 L(x, y) + w_6 L(x, y+1) \\ & + w_7 L(x+1, y-1) + w_8 L(x+1, y) + w_9 L(x+1, y+1) \end{aligned} \tag{2.12}$$

Filtry dolnoprzepustowe

Jednym z podstawowych filtrów jest filtr usuwający zakłócenia obrazu. Do tego celu można użyć filtru uśredniającego. Uzyskujemy go wykonując konwolucję z macierzą współczynników o wartościach:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.13}$$

Po zastosowaniu powyższego filtru uzyskamy rezultat przedstawiony na rysunku 2.6.

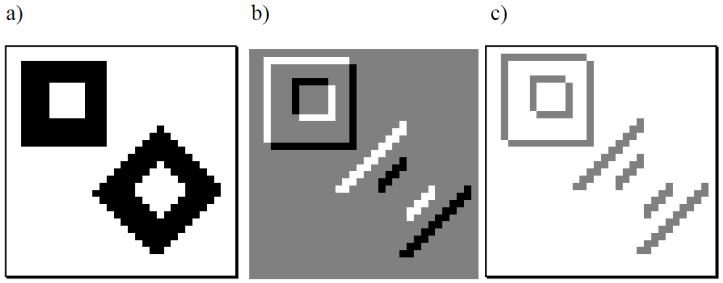
Filtry górnoprzepustowe

Filtry górnoprzepustowy służą w grafice komputerowej do uzyskiwaniu informacji, w których miejscach obrazu występują szybkie zmiany jasności. Dzięki temu mogą służyć do znajdowania konturów oraz krawędzi. Jednym z podstawowych filtrów górnoprzepustowych jest filtr Robertsa. Macierz współczynników przybiera postać:

$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{2.14}$$

Rysunek 2.6: Zostosowanie filtru dolnoprzepustowego.

Po zastosowaniu filtru na przykładowym obrazie można stwierdzić, że uzyskujemy pewną informację na temat niektórych krawędzi. Zilustrowano to na rysunku 2.7.



Rysunek 2.7: Zostosowanie filtru górnoprzepustowego. a) obraz wejściowy, b) rezultat działania filtru, c) zaznaczenie znalezionych krawędzi jednym kolorem.

Stosując bardziej urozmaicone maski (macierze współczynników) możemy uzyskać inne rezultaty (na przykład filtr *Sobela*). Począwszy od krawędzi skierowanych w odpowiednim kierunku bądź leżących pod odpowiednim kątem, do wszystkich krawędzi występujących na rysunku. Temat opisany jest obszernie w [2].

2.3. Canny edge detector

Algorytm Canny'ego jest jedną z metod realizacji segmentacji służącą do wykrywania krawędzi. Korzystając z [5] projekt Canny'ego bazuje na jednowymiarowym modelu krawędzi skokowej w ciągłej dziedzinie o amplitudzie h_E z dodatkowym, tak zwanym białym szumem Gaussa o standartowym odchyleniu σ[2].

Implementacja algorytmu opiera się na filtrach Gaussa oraz na filtrach różnicowych. Operacje te pozwolą na zaakcentowanie konturów obrazu. W kolejnych krokach algorytmu następuje selekcja istotnych

krawędzi, domknięcie krawędzi oraz lokalizacja konkretniej krawędzi w danym pikselu.

Rozmycie obrazu filtrem Gaussa

W pierwszym kroku stosujemy operację konwolucji (konwolucja została opisana w podrozdziale 2.2.2) z filtrem Gaussa. Operacja ta usunie szum z obrazu. Rozkład Gaussa charakteryzuje się parametrem σ, czyli odchyleniem standardowym. Maska tego filtru jest generowana dynamicznie podczas działania algorytmu i zależy od parametru σ.

Przetwarzanie operatorem gradientu kierunkowego

Kolejnym krokiem algorytmu jest analiza zmienności funkcji $g(j,k)$, która opisuje obraz wejściowy. Do tego celu można zastosować filtr konturowy Sobela z rotacją maski, tak aby wyeliminować czułość kierunkową filtru. Stosuje się 8 masek dla kierunków odpowiednio: 2 w poziomie (o przeciwnych zwrotach), 2 w pionie oraz po dwa w kierunkach pod kątem $45°$ i $135°$. Dla każdego kierunku wyznacza się macierz średnich natężeń pikseli, które oznaczymy odpowiednio S_1 dla kierunku poziomego (wierszy), S_2 dla kierunku pionowego (kolumn), S_3 dla kierunku pod kątem $45°$ oraz S_4 dla kąta $135°$.

Mapa kierunków

Tablice S_1 oraz S_2 zawierają składowe wektorów gradientu obrazu. Dla każdego piksela wyznaczamy kąt wektora gradientu z osią poziomą OX:

$$\theta(j,k) = arctg\frac{S_2(j,k)}{S_1(j,k)}. \tag{2.15}$$

Ten krok algorytmu jest szczególnie ważny przy procesie konstrukcji krawędzi o szerokości jednego piksela.

Eliminacja ciemniejszych pikseli

Ten etap algorytmu polega na eliminacji punktów o mniejszej intensywności. Analiza jest przeprowadzana na każdym pikselu obrazu wraz z jego otoczeniem (najbliższymi sąsiadami). Korzystając z informacji gradientów o potencjalnej orientacji krawędzi, do której dany piksel może przynależeć sprawdzamy czy dany piksel jest jaśniejszy niż piksele nieleżące na prostej wyznaczonej przez gradient. Jeżeli jest jaśniejszy, pozostaje bez zmian, jeżeli natomiast któryś z sąsiednich pikseli jest jaśniejszy, bieżący piksel pozostaje gaszony.

Progowanie krawędzi

W tym kroku następuje sprawdzenie wartości gradientu każdego piksela obrazu i w zależności od ustalonych parametrów algorytmu *HT* (ang. *high threshold*) oraz *LT* (ang. *low threshold*) pozostawiamy dany piksel na obrazie lub go gasimy. Następnym etapem jest domknięcie krawędzi, które może być wy-

konane na kilka sposobów – zależnie od potrzeb. Najprostszym sposobem jest wykorzystanie algorytmu morfologicznego z odpowiednią maską.

Podsumowanie

Algorytmy służące do segmentacji obrazu są niezwykle cenne i bardzo pomocne przy rozpoznawaniu obrazów. Algorytm Canny'ego daje bardzo dobre rezultaty i przy odpowiednim dobraniu parametrów algorytmu można bardzo dokładnie wyznaczyć kontury badanego obiektu.

2.4. Transformacja Hough

Transformacja Hough (ang. Hough Transform, HT) została wprowadzona w 1962 roku przez Paula Hougha, jako metoda wykrywania wzorców na obrazach binarnych. Była jedną z pierwszych prób automatycznego rozpoznawania śladów podczas prac nad obserwacjami ruchów cząsteczek elementarnych w komorze pęcherzykowej. Po początkowym braku zainteresowania z czasem pozyskiwała coraz większą popularność.

W roku 1972 Richard Duda oraz Peter Hart zastosowali transformację Hougha do detekcji krzywych analitycznych – wcześniej była wykorzystywana jedynie do wykrywania prostych (po raz pierwszy zacytowana w pracy Rosenfelda: *"Picture Processing by Computer"* w 1969 roku). Z czasem powstało wiele odmian transformaty możliwych do zastosowania w wielu różnych dziedzinach, takich jak: analiza danych astronomicznych, sejsmologii, biometrii, biomedycynie i innych.

Transformacja Hougha jest stosowana na obrazach binarnych najczęściej po wcześniejszym użyciu odpowiednich filtrów wykrywających krawędzie oraz filtrów morfologicznych, tak aby uwypuklić charakterystyczne kształty, które nas interesują.

Wykrywanie okręgów

W ogólnym przypadku równanie krzywej zapisujemy jako:

$$f(x, y, a_1, a_2 \dots a_n) = 0 \tag{2.16}$$

gdzie $a_1, a_2 \dots a_n$ są pewnymi parametrami.

Zgodnie z [6] transformacja Hougha polega na tym, że pojedynczemu punktowi (x, y) odpowiada równanie o zmiennych $a_1, a_2 \dots a_n$ w przestrzeni kartezjańskiej. Dla dwóch parametrów otrzymujemy krzywą, dla trzech jest to już powierzchnia.

Szczególnym zastosowaniem transformaty Hough jest wykrywanie okręgów. Zapisując równanie parametryczne koła otrzymujemy:

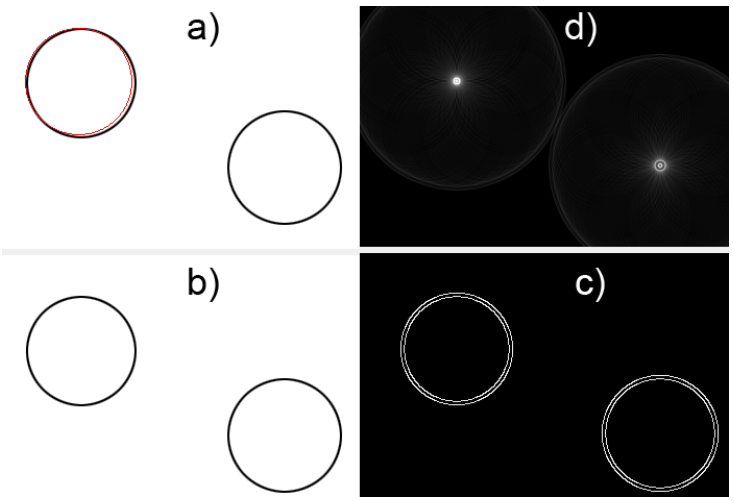
$$(x-a)^2 + (y-b)^2 = r^2 \tag{2.17}$$

Parametry a i b równania (2.17) opisują środek okręgu, natomiast parametr r odpowiada za jego promień. Ponieważ algorytm wyznaczania transformaty Hougha dla trzech parametrów jest obliczeniowo skomplikowany i wymaga sporych nakładów pamięciowych, można poszukiwać okręgu o zadanym promieniu. Wtedy poszukiwana funkcja ma postać:

$$a = f(b) \tag{2.18}$$

Po wyznaczeniu transformacji Hougha dla obrazu wejściowego, na którym poszukiwane są kształty spełniające funkcję (2.18) otrzymano zbiór punktów. Punkty w przestrzeni Hougha spełniające zadane równanie określają środek znalezionego wzorca – w opisywanym przypadku okręgu o zadanym promieniu.

Na rysunku 2.8 przedstawiony został przykład wyszukiwania okręgów przy pomocy transformacji Hougha. Na rysunku d) wyznaczone zostały punkty w przestrzeni Hougha opisujące środki znalezionych okręgów o zadanym promieniu, w tym przypadku $r = 50$.



Rysunek 2.8: Kolejne etapy wyszukiwania okręgów: a) obraz wejściowy, b) obraz po zastosowaniu binaryzacji, c) wyszukiwanie krawędzi, d) prezentacja transformacji Hougha.

2.5. Algorytmy wykorzystywane do lokalizacji oraz śledzenia twarzy

2.5.1. Metody szablonowe

Metody szablonowe polegają na poszukiwaniu zadanego wzorca w postaci obrazu na innym, zazwyczaj większym obrazie. Przykładem może być tutaj poszukiwanie twarzy na obrazie zawierającym kilka osób. Poszukiwanym wzorcem jest mniejszy fragmentu obrazu, który zawiera jedną konkretną poszukiwaną twarz.

Algorytmy dopasowujące wzorzec dzielą się na dwie zasadnicze grupy: FBM (ang. *feature-based matching*) oraz ABM (ang. area-based matching). FBM polega na poszukiwaniu punktów charakterystycznych obrazu (cech) w szablonie i poszukiwanie tych cech na obrazie docelowym. Metody te mają dużą tolerancję na obroty obrazu, zmiany wielkości oraz perspektywy. W momencie pojawiania się zakłóceń rezultaty metody drastycznie się pogarszają. ABM natomiast, jako szablon przechowuje zbiór pikseli obrazu poszukiwanego. Metoda jest bardziej złożona obliczeniowo i wymagająca pod względem pamięciowym. Jej mocną stroną jest niewrażliwość na liniowe zmiany jasności obrazu. Tolerancja na zakłócenia również przedstawia się lepiej, niż w przypadku metod FBM.

Normalized Grey-scale Correlation

Jedną z podstawowych metod należących do grupy ABM jest metoda NGC (ang. *Normalized Grey-scale Correlation*). Metoda polega na obliczaniu współczynnika dopasowania w każdym możliwym punkcie obrazu przeszukiwanego.

Oznaczmy macierz szablonu jako $T_{i,j}$ o wymiarach m na n oraz macierz obrazu przeszukiwanego jako $S_{i,j}$ o rozmiarach p na q. Niech $\lambda(i,j)$ będzie współczynnikiem dopasowania szablonu T w miejscu (i,j) obrazu przeszukiwanego S. Zgodnie z [7] współczynnik wyznaczamy następująco:

$$\lambda(i,j) = \frac{\sum_{x=0}^{n-1}\sum_{y=0}^{m-1}(S_{i+x,j+y} - \bar{S}_{i,j})(T_{x,y} - \bar{T})}{\sqrt{\sum_{x=0}^{n-1}\sum_{y=0}^{m-1}(S_{i+x,j+y} - \bar{S}_{i,j})^2 \sum_{x=0}^{n-1}\sum_{y=0}^{m-1}(T_{x,y} - \bar{T})^2}} \quad 0 \le i < p-n, \quad 0 \le j < q-m \tag{2.19}$$

gdzie:

$$\bar{S}_{i,j} = \frac{1}{m \times n}\sum_{x=0}^{n-1}\sum_{y=0}^{m-1} S_{i+x,j+y}, \tag{2.20}$$

oraz:

$$\bar{T}_{i,j} = \frac{1}{m \times n}\sum_{x=0}^{n-1}\sum_{y=0}^{m-1} T_{i,j}. \tag{2.21}$$

Rozwiązaniem algorytmu jest punkt o współrzędnych (i, j), dla którego współczynnik $\lambda(i, j)$ osiągnął wartość największą (maksymalną wartością współczynnika $\lambda(i, j)$ jest 1).

Zaletą metody jest stosunkowo dobra dokładność, jednakże cena, jaką trzeba zapłacić to duża ilość obliczeń. Współczynnik $\lambda(i, j)$ musi zostać wyliczony dla każdego punktu na obrazie S. Przy dużych rozmiarach obrazu S operacja ta może trwać długo.

Możliwe optymalizacje algorytmu

Aby przyspieszyć poszukiwanie wzorca na obrazie stosuje się tak zwane piramidy obrazu. Polega to na tym, że tworzone są kopie obu obrazów, zarówno szablonu T jak i obrazu przeszukiwanego S w mniejszych rozmiarach (zazwyczaj o wielkości dwa razy mniejszej od poprzedniego). Tworzymy kilka poziomów piramid zależnie od wielkości oryginalnych obrazów. Wstępne przeszukiwanie wzorca rozpoczynamy na obrazach najmniejszych, a następnie po wyznaczeniu współczynników $\lambda(i, j)$ dla obrazów mniejszych, przeszukujemy obrazy większe, ale tylko w rejonach gdzie współczynnik $\lambda(i, j)$ przekroczył zadaną ustaloną wartość. W ten sposób ograniczamy ilość obliczeń dla dużego obrazu oryginalnego S przyspieszając jednocześnie działanie algorytmu.

2.5.2. Algorytm KLT

Algorytm KLT (*Kanade-Lucas-Tomasi*) służy do wyznaczania punktów charakterystycznych – dalej nazwanych *cechami* – obrazu a następnie śledzenia tych punktów w kolejnych klatkach obrazu. Punkty zostają wybrane w sposób optymalny, aby ich śledzenie było stosunkowo proste. Głównym założeniem algorytmu jest fakt, że dobra cecha to taka, którą łatwo śledzić. Jeżeli wyznaczona wcześniej cecha nie zostanie wykryta w bieżącej klatce obrazu zostaje wyznaczona nowa, tak aby ilość cech opisujących obraz była stała.

Wyznaczenie cech

Niech macierz Z 2.22 opisuje lokalną intensywność:

$$Z = \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix} \tag{2.22}$$

gdzie:

$g(x, y)$ jest funkcją intensywności obrazu w punktach x oraz y. Kandydatem na cechę jest fragmentu obrazu małej wielkości, na przykład 25×25 pikseli, dla którego środkowych pikseli wartości własne macierzy Z, λ_1 oraz λ_2 spełniają warunek:

$$min(\lambda_1, \lambda_2) > \lambda \tag{2.23}$$

gdzie wartość parametru λ określa odległość pomiędzy kolejnymi cechami. Spośród kandydatów na cechy wybieranych jest N_f o najwyższych współczynnikach.

Śledzenie cech

Wyznaczone cechy w pierwszej klatce obrazu są następnie w kolejnych klatkach śledzone. Niech funkcja $g(x, y)$ wyznacza intensywność punktu obrazu z poprzedniej klatki, natomiast funkcja $j(x, y)$ wyznacza intensywność punktu obrazu klatki bieżącej. Oznaczmy cechę z poprzedniej klatki jako: $\mathbf{u} = [x, y]$. Poszukiwany jest punkt $v = u + d$ w nowej klatce obrazu minimalizując:

$$\epsilon(d) = \sum_{x,y \in W} (g(x, y) - j(x + d_x, y + d_y))^2 \tag{2.24}$$

gdzie d jest otoczeniem punktu **u**. Po przekształceniach 2.24 otrzymujemy układ równań:

$$Zd = b, \tag{2.25}$$

$$\nabla g = [g_x \, g_y] \,, \tag{2.26}$$

$$\Delta g = g(x, y) - j(x + d_x, y + d_y), \tag{2.27}$$

$$Z = \sum_{W} \begin{bmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{bmatrix}, \tag{2.28}$$

$$b = \sum_{W} \begin{bmatrix} \Delta g(x, y) g_x \\ \Delta g(x, y) g_y \end{bmatrix}. \tag{2.29}$$

Rozwiązanie d określa przesunięcie punktu **u** pomiędzy klatką bieżącą a poprzednią. Jeżeli macierz nie ma rozwiązań, punkt przestaje być cechą.

Podsumowanie

Wadą rozwiązania jest to, że dla dużych zmian na obrazie pomiędzy klatkami stracimy większość cech z poprzedniej iteracji. Rozwiązaniem jest zastosowanie piramidy coraz mniejszych kopii obrazu. Zwiększa to jednak zasoby potrzebne podczas wykonywania algorytmu, szczególnie dla dużych obrazów. W dokumentach [8] oraz [9] przedstawiono zastosowanie algorytmu KLT do śledzenia ruchomych obiektów na obrazie wideo. Zapoznając się z rezultatami tam opisanymi można stwierdzić, że algorytm radzi sobie bardzo dobrze z wyznaczaniem cech charakterystycznych oraz ich śledzeniem. Wadą tego rozwiązania jednak jest brak kontroli nad tym, które punkty są wybierane jako charakterystyczne.

2.5.3. CamShift

Algorytm CamShit (*Continuously Adaptive Mean Shift*) służy do wyznaczenia współrzędnych oraz rozmiarów okna zawierającego twarz na danym obrazie. Metoda została opracowana przez G. Bradskiego i opisana w pracy [1].

Wyznaczenie momentów

Pierwszym etapem algorytmu jest wyznaczenie momentów: momentu zerowego $M00$ 2.30, pierwszego momentu dla współrzędnej x oraz y ($M10$ oraz $M01$) 2.31 i drugiego momentu dla współrzędnej x i y ($M20$ i $M02$) 2.32:

$$M_{00} = \sum_X \sum_Y P(x,y) \tag{2.30}$$

$$M_{10} = \sum_X \sum_Y xP(x,y) \quad M_{01} = \sum_X \sum_Y yP(x,y) \tag{2.31}$$

$$M_{20} = \sum_X \sum_Y x^2P(x,y) \quad M_{02} = \sum_X \sum_Y y^2P(x,y) \tag{2.32}$$

gdzie $P(x,y)$ wyraża prawdopodobieństwo tego, czy dany piksel (x,y) należy do wnętrza poszukiwanego obrazu twarzy.

Współrzędne szukanego obiektu wyliczamy ze wzoru:

$$x_c = \frac{M_{10}}{M_{00}} \quad y_c = \frac{M_{01}}{M_{00}} \tag{2.33}$$

Długość L oraz szerokość w okna wyznaczamy ze wzorów 2.34:

$$l = \left(\frac{(a+c) + (b^2 + (a-c)^2)^{\frac{1}{2}}}{2} \right)^{\frac{1}{2}} \quad w = \left(\frac{(a+c) + (b^2 - (a-c)^2)^{\frac{1}{2}}}{2} \right)^{\frac{1}{2}} \tag{2.34}$$

gdzie parametry a, b oraz c wynoszą:

$$a = \frac{M_{20}}{M_{00}} - x_c^2 \quad b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c \right) \quad c = \frac{M_{20}}{M_{00}} - y_c^2 \tag{2.35}$$

Miary prawdopodobieństwa

Aby móc zastosować powyższy algorytm należy wyznaczyć odpowiednią funkcję $P(x,y)$, która stwierdzi czy dany piksel (x,y) należy do poszukiwanego obiektu. Do tego celu użyte zostają miary prawdopodobieństwa przynależności piksela do barwy twarzy. Przykładowa miara została przedstawiona w równaniu 2.36:

$$P = \begin{cases} cos^2\left(abs\frac{H_t - Hue}{180}\frac{\pi}{2}\right) & \text{dla } |H_t - Hue| < 180 \\ cos^2\left(\frac{360-(absH_t - Hue)}{180}\frac{\pi}{2}\right) & \text{w przeciwnym wypadku} \end{cases} \tag{2.36}$$

gdzie:

- P - wartość prawdopodobieństwa przynależności piksela do obiektu w zakresie od 0 do 1,
- Ht - wartość barwy piksela odpowiadająca kolorowi skóry twarzy w modelu HSV,
- Hue - wartość barwy danego piksela obrazu w modelu HSV.

Podsumowanie

Aby uniknąć zakłóceń spowodowanych przez wystąpienie obszarów białych oraz czarnych na obrazie, można brać również pod uwagę wartość składowej S (ang. *saturation*) koloru w modelu HSV, jednak podstawową wadą rozwiązania jest oparcie detekcji na podstawie koloru obiektu. Przy zmiennych warunkach oświetleniowych, różnorodności osób występujących na obrazach poddanych analizie algorytmem camshift, metoda ta wymaga znacznych modyfikacji parametrów miar prawdopodobieństwa. Nie sposób stworzyć miarę uniwersalną, która sprawdzi się w różnorodnych warunkach.

2.6. Sieci neuronowe

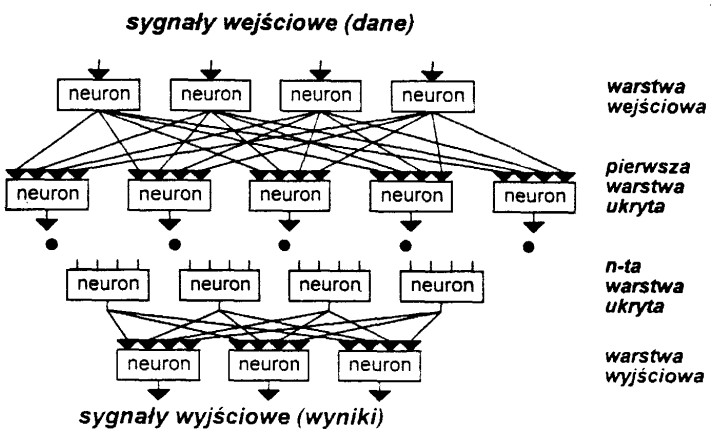
Sieci neuronowe powstały na bazie uproszczonego modelu mózgu. W ich skład wchodzi duża liczba elementów przetwarzających informacje. Ich ilość może sięgać nawet kilkudziesięciu tysięcy, a w szczególnych przypadkach nawet więcej. Elementy te nazywane są neuronami. Ich funkcje są bardzo uproszczone w stosunku do rzeczywistych komórek. Sieć neuronowa powstaje przez połączenia neuronów o parametrach (tak zwanych wagach) modyfikowanych w trakcie uczenia sieci. Sposoby oraz parametry połączeń neuronów określają rodzaj sieci. Rozwiązaniem zadań stawianych sieci są sygnały pojawiające się na jej wyjściach w zależności od sygnałów wejściowych.

Sieci neuronowe wykorzystywane współcześnie z reguły mają budowę warstwową. Wyróżniamy następujące warstwy: wejściowa, wyjściowa oraz warstwy ukryte. Ogólny schemat sieci przedstawiono na rysunku 2.9.

2.6.1. Rodzaje sieci

Sposoby połączeń pomiędzy neuronami oraz wzajemne ich współdziałanie spowodowało powstanie wielu różnych rodzajów sieci. Każdy rodzaj sieci jest ściśle powiązany ze sposobem doboru wag.

Zgodnie z [10], do najczęściej stosowanych rodzajów sieci zaliczane są:



Rysunek 2.9: Ogólny schemat sieci wielowarstwowej.

- sieć jednokierunkowa jednowarstwowa,
- sieć jednokierunkowa wielowarstwowa,
- sieci rekurencyjne.

W publikacjach takich jak [11] oraz [12] znaleźć można także obszerne informacje na temat takich sieci jak:

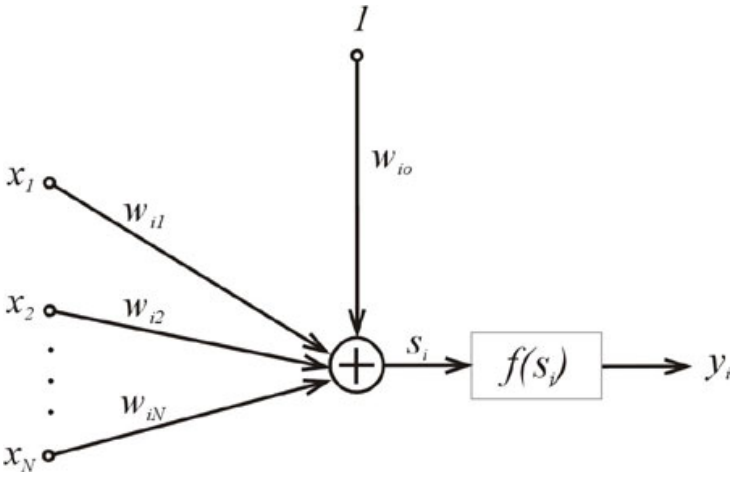
- nieliniowe sieci neuronowe,
- sieci CP,
- sieci rezonansowe,
- sieć Hopfielda.

2.6.2. Sieć jednokierunkowa wielowarstwowa

Sieć jednokierunkowa wielowarstwowa jest rodzajem sieci, która cieszy się największym zainteresowaniem i jest jedną z najczęściej stosowanych. Przepływ sygnałów odbywa się w jednym kierunku – od wejścia do wyjścia. Model neuronu sigmoidalnego składa się z elementu sumacyjnego, do którego wchodzą sygnały wejściowe x_1, x_2, ..., x_N (rysunek 2.10). Sygnał wyjściowy sumatora y_i wyznacza się w sposób następujący:

$$y_i = \sum_{j=1}^{N} W_{i,j} x_j = W_{i0} \tag{2.37}$$

Funkcja aktywacji $f(x)$ jest ciągła i przyjmuje postać funkcji unipolarnej $(0, 1)$ lub bipolarnej $(-1, 1)$. Pierwsza z nich ma postać:



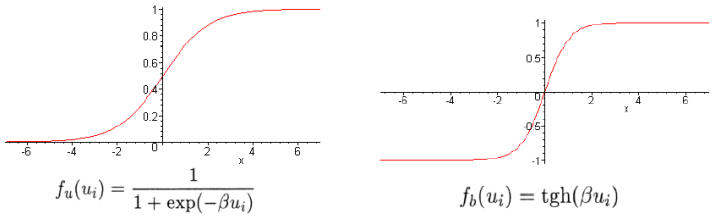
Rysunek 2.10: Model neuronu sigmoidalnego.

$$f(x) = \frac{1}{1 + e^{-\beta t}}, \tag{2.38}$$

natomiast bipolarna ma postać:

$$f(x) = tanh(\beta x). \tag{2.39}$$

Obie funkcje przedstawiono na rysunku 2.11.



Rysunek 2.11: Unipolarna oraz bipolarna funkcja aktywacji.

Parametr β dobierany jest przez użytkownika. Jego wartość wpływa na kształt funkcji aktywacji. Zazwyczaj przyjmuje się dla uproszczenia $\beta = 1$.

Funkcja celu

Uczenie sieci polega na optymalizacji funkcji celu, minimalizując błąd między wartościami jakie są żądane, a aktualnymi wartościami na wyjściu sieci.

Wprowadzone zostają oznaczenia:

- $\mathbf{d} = [d_1, d_2, \ldots, d_M]^T$ - wektor znanych wartości żądanych na wyjściu,

- $U = \{(x(j), \mathbf{d}(j)) | j = 1 \dots p\}$ - p-elementowy zbiór uczący,
- $\mathbf{W} = [W_1, W_2, \dots, W_n]^T$- wektor wag danej sieci,
- E - funkcja celu.

Funkcja celu ma postać:

$$E = \frac{1}{2} \sum_{i=1}^{M} (y_i - d_i)^2 \tag{2.40}$$

dla jednej pary uczącej $(\mathbf{x}, \mathbf{d})$. Dla ciągu uczącego składającego się z p elementów otrzymuje się:

$$E = \frac{1}{2} \sum_{j=1}^{p} \sum_{i=1}^{M} (y_i(j) - d_i(j))^2. \tag{2.41}$$

Metody gradientowe używane do optymalizacji funkcji celu wymagają, aby funkcje były ciągłe i różniczkowalne. Funkcja celu 2.41 spełnia te warunki, ponieważ funkcje aktywacji są ciągle i różniczkowalne.

Minimalizacja funkcji celu

Gradient funkcji celu wyznacza się w sposób:

$$\nabla E = \left[\frac{\partial E}{\partial W_1}, \frac{\partial E}{\partial W_2}, \dots, \frac{\partial E}{\partial W_n} \right]^T. \tag{2.42}$$

Kierunek normalizacji $\mathbf{p}(\mathbf{W}(k))$ wyznacza się na podstawie gradientu funkcji celu w każdym kroku. Kolejnym krokiem jest aktualizacja wag sieci zgodnie z:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + \eta p(\mathbf{W}(k)), \tag{2.43}$$

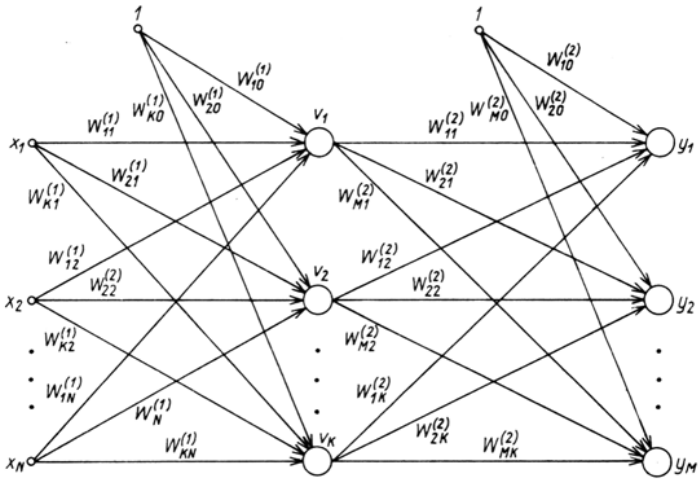
gdzie η jest współczynnikiem uczenia.

Metoda propagacji wstecznej błędu

Cytując za [10]:

> "W uproszczeniu przyjęto, że celem uczenia sieci jest określenie wartości wag neuronów wszystkich warstw sieci w taki sposób, aby przy zadanym wektorze wejściowym x uzyskać na wyjściu sieci wartości sygnałów wyjściowych y_i, równające się z dostateczną dokładnością wartościom żądanym d_i dla $i = 1, 2, \dots, M$."

Metoda wstecznej propagacji błędu jest najczęściej stosowaną metodą uczenia sieci. Wykorzystuje ona gradientowe metody optymalizacji przy doborze wag. Nazwa "wsteczna propagacja"wzięła się od



Rysunek 2.12: Sieć neuronowa.

sposobu, w jaki obliczane są błędy w poszczególnych warstwach sieci. Sygnał błędu rozprzestrzenia się od warstwy wyjściowej do warstwy wejściowej.

Dla sieci z rysunku 2.12 przyjmijmy funkcję celu:

$$E = \frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{M}(y_i(j) - d_i(j))^2 \tag{2.44}$$

Różniczkując funkcję celu otrzymujemy składowe gradientu:

$$\begin{aligned} E &= \frac{1}{2}\sum_{k=1}^{M}\left[f\left(\sum_{i=0}^{K} W_{ki}^{(2)} v_i\right) - d_k\right]^2 \\ &= \frac{1}{2}\sum_{k=1}^{M}\left[f\left(\sum_{i=0}^{K} W_{ki}^{(2)} f\left(\sum_{j=0}^{K} W_{ij}^{(1)} x_j\right)\right) - d_k\right]^2. \end{aligned} \tag{2.45}$$

Dla warstwy ostatniej otrzymujemy:

$$\frac{\partial E}{\partial W_{ij}^{(2)}} = (y_i - d_i)\frac{df(u_i^{(2)})}{du_i^{(2)}} v_j. \tag{2.46}$$

Wprowadzając oznaczenie:

$$\delta_i^{(2)} = (y_i - d_i)\frac{df(u_i^{(2)})}{du_i^{(2)}}, \tag{2.47}$$

otrzymuje się:

$$\frac{\partial E}{\partial W_{ij}^{(2)}} = \delta_i^{(2)} v_j. \tag{2.48}$$

Dla warstwy ukrytej wykorzystujemy rezultat obliczeń dla warstwy wyjściowej:

$$\frac{\partial E}{\partial W_{ij}^{(1)}} = \delta_i^{(1)} x_j \tag{2.49}$$

gdzie:

$$\delta_i^{(1)} = \sum_{k=1}^{M} \delta_k^{(2)} W_{ki}^{(2)} \frac{df(u_i^{(1)})}{du_i^{(1)}}. \tag{2.50}$$

Poprzez analogię wzory rozszerza się na sieć posiadającą więcej warstw ukrytych. Wagi uaktualniamy mając wyliczone wszystkie składowe gradientu ze wzoru:

$$\Delta \mathbf{W} = -\eta \nabla E(\mathbf{W}). \tag{2.51}$$

Podsumowanie

Głównymi zaletami zastosowania wstecznej propagacji błędu są:

- zmniejszenie złożoności obliczeniowej,
- możliwość przetwarzania równoległego.

Do głównych wad metody należy przede wszystkim uzależnienie jej działania od wybranych wartości początkowych wag. Wybranie niewłaściwych wartości bądź punktu początkowego może spowodować wyznaczenie minimum lokalnego, z którego algorytm nie zdoła wyjść. Odnalezienie wtedy minimum globalnego może się nie udać. Można w pewnym stopniu zapobiec tego typu problemom:

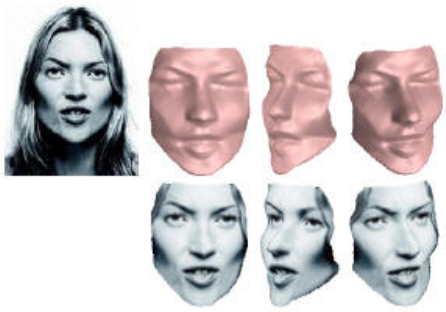
- wagi początkowe powinny mieć wartości takie, aby sygnał wyjściowy nieliniowej częsci neuronu był mniejszy od jedności,
- kilkakrotne rozpoczynanie uczenia rozpoczęte dla różnych wartości wag.

3. Istniejące rozwiązania

3.1. Przegląd istniejących rozwiązań

3.1.1. Molding Face Shapes by Example

W dokumencie [13] można znaleźć opis projektu **Molding Face Shapes by Example**. Program służy do tworzenia trójwymiarowych modeli głowy na podstawie statycznego obrazu twarzy. Algorytmy opierają się na teorii odbić lambertowskich. Na podstawie prawa Lamberta założyć można, że natężenie światła docierające do obserwatora zależne jest wyłącznie od $cos\alpha$, gdzie α to kąt, pod jakim pada światło na dany obiekt. Na tej podstawie podjęto próbę rekonstrukcji powierzchni obiektu przedstawionego na płaskim obrazie do trójwymiarowej, przestrzennej bryły.



Rysunek 3.1: Rezultaty działania programu.

Na rysunku 3.1 przedstawiono rysunki demonstrujące działanie aplikacji.

Wady rozwiązania

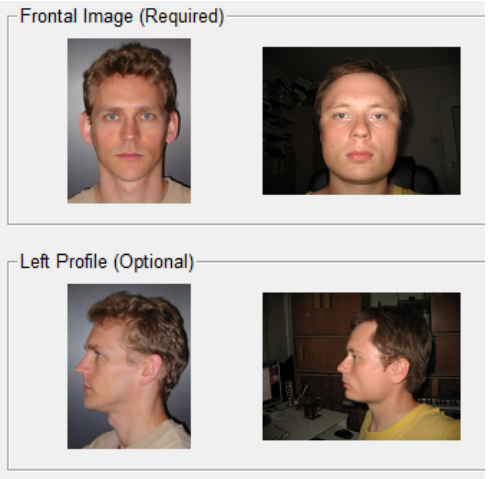
Opisane wyżej rozwiązanie w żaden sposób nie uwzględnia cech anatomicznych twarzy, nie dokonuje analizy ani nie wykrywa żadnych punktów kontrolnych. Stworzony na jej podstawie obiekt jest bryłą trójwymiarową – zbiorem punktów w przestrzeni. Aby móc dokonać analizy tak stworzonego modelu potrzebne są jeszcze szczegółowe informacje o tym, co kryje się za tymi punktami. W przeciwnym wypadku nie można wykorzystać tych punktów do wykrywania gestów, czy mimiki twarzy.

3.1.2. Facegen Modeller

Facegen Modeller, którego można znaleźć pod adresem [14], jest komercyjną aplikacją, za którą trzeba zapłacić 299$ USD. Jest to dość rozbudowany system służący do tworzenia trójwymiarowych modeli twarzy. Aplikacja posiada szeroki wachlarz możliwości, w jaki można modyfikować stworzoną twarz. Pozawala na symulowanie procesów starzenia, dowolnie manipulować elementami twarzy, tak by uzyskać przeróżne gesty oraz miny. Posiada szereg zdefiniowanych ekspresji twarzy i sposobów wyrażania uczuć. Aplikacja pozwala również nanosić cechy etniczne bądź rasowe na daną twarz.

Tworzenie modelu twarzy na podstawie zdjęcia

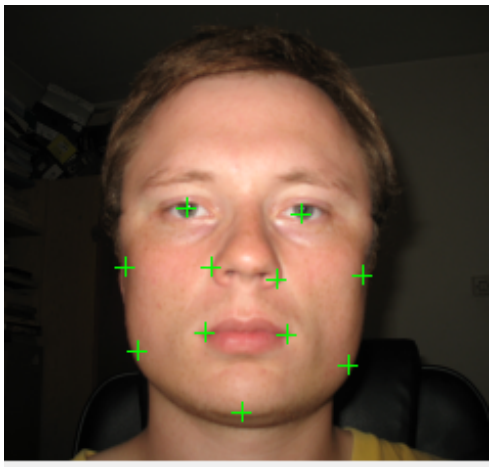
Aplikacja posiada możliwość stworzenia modelu twarzy na podstawie obrazu, bądź obrazów twarzy. Na rysunku 3.2 przedstawiono interfejs użytkownika, w którym ładujemy zdjęcia osoby, której twarz chcemy zamodelować. Opcjonalnie można załadować zdjęcia z profilu, co poprawia jakość tworzonego modelu.



Rysunek 3.2: Wczytanie zdjęć twarzy z przodu oraz z profilu.

Kolejnym krokiem jest opisanie wczytanych zdjęć przez punkty kontrolne. Za pomocą myszki przesuwamy szereg zdefiniowanych punktów w z góry ustalone miejsca. Zdefiniowane punkty opisują na przykład kąciki ust, podstawę nosa, brodę. Czynność tą wykonujemy dla każdego zdjęcia, jakie załadowaliśmy. Proces ten przedstawiony został na rysunku 3.3.

Po wykonaniu powyższych czynności system rozpoczyna tworzenie modelu trójwymiarowego. Proces ten trwa stosunkowo długo i może zająć do kilkunastu minut. Na rysunku 3.4 przedstawiono zrekonstruowany model trójwymiarowy na podstawie zdjęć wczytanych do programu. Model jest pewnego rodzaju przybliżeniem, które rozmywa część szczegółów anatomicznych. Dodatkowo tekstura powierzchni twarzy sprawia wrażenie rozmytej.



Rysunek 3.3: Ręczne zaznaczanie cech na obrazie twarzy.

Rysunek 3.4: Zrekonstruowana głowa trójwymiarowa.

Korzystając z opcji dostępnych w programie można spróbować modyfikować stworzony model na różne sposoby. Jednym z nich jest próba symulacji starzenia się. Ustawiając w parametrach programu wiek osoby na 50 lat otrzymujemy model przedstawiony na rysunku 3.5.

Podsumowanie

Szereg dostępnych opcji programu jest bardzo duży. Aplikacja pozwala na bardzo wiele manipulacji nad stworzonym obiektem, nie mniej jednak sam proces konstruowania modelu pozostawia wiele do życzenia. Główną wadą rozwiązania jest czas, jaki jest potrzebny na stworzenie modelu. Ta cecha aplikacji powoduje, że zastosowane rozwiązanie nie nadaje się do procesów stosowanych w aplikacjach działających w czasie rzeczywistym. Drugą wadą rozwiązania jest konieczność ręcznego wprowadzania punktów kontrolnych. Wstępne wykrycie punktów kontrolnych, a następnie ewentualna ręczna ich korekta byłoby dobrym pomysłem na kolejną wersję aplikacji.

Rysunek 3.5: Zastosowanie przekształceń do zasymulowania zmiany wieku.

3.1.3. FaceShop

Na stronie firmy Abalonellc [15] znaleźć można oprogramowanie o nazwie FaceShop. Niestety dostępnych jest bardzo mało informacji technicznych na temat tego projektu. Jedyne, co można znaleźć, to filmiki demonstracyjne aplikacji. Aplikacja służy do tworzenia trójwymiarowych modeli głowy na podstawie obrazów statycznych twarzy. Niestety z tego, co można zobaczyć na filmach demonstracyjnych, proces tworzenia modelu jest niesłychanie trudny i czasochłonny. Użytkownik zmuszony jest do ręcznego przesuwania punktów kontrolnych dla każdego elementu twarzy, zaznaczania charakterystycznych miejsc, a nawet ręcznego wskazywania łuków występujących na twarzy (takich jak łuki brwiowe, zarysy policzków).

Rezultaty procesu tworzenia modelu poprzedzone tak precyzyjnymi ręcznymi modyfikacjami są bardzo dobre, jednakże aby zrobić to dobrze potrzebna jest bardzo dobra znajomość programu oraz sporo czasu.

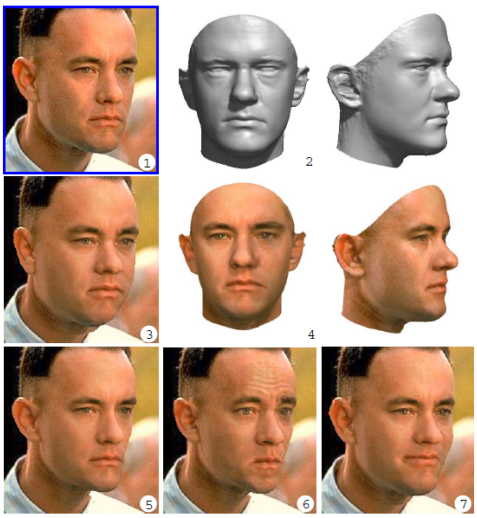
Podsumowanie

Aplikacja daje bardzo dobre rezultaty, ale proces wstępny, który wymaga bardzo dużo ręcznego wprowadzania danych powoduje, że zastosowane tam rozwiązanie nie nadaje się do automatycznych systemów. Kolejną dyskwalifikującą cechą aplikacji jest czas, w jakim model trójwymiarowy głowy jest tworzony.

3.1.4. A Morphable Model For The Synthesis Of 3D Faces

W dokumencie [16] opisano rozwiązanie, które – podobnie jak wyżej opisane – pozwala tworzyć modele trójwymiarowe głowy. Rozwiązanie to oparte jest na morfingu. W tym przypadku dostępnych jest nieco więcej informacji technicznych na temat zrealizowanego projektu.

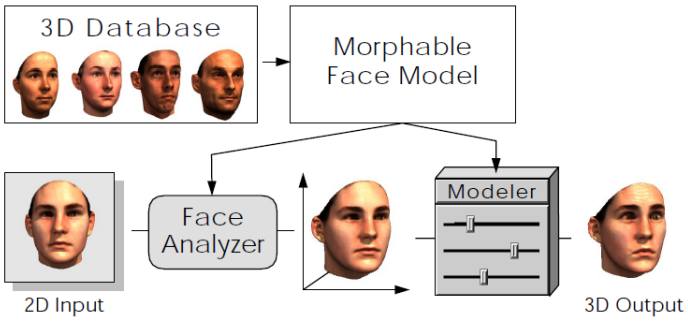
Na rysunku 3.6 przedstawiono demonstracyjne rezultaty działania systemu.



Rysunek 3.6: Kolejne etapy działania programu.

Zastosowane rozwiązanie

Rysunek 3.7 przedstawia ogólny schemat działania algorytmu. Program zawiera zestaw trójwymiarowych modeli twarzy sporządzonych na podstawie 200 osób (100 mężczyzn i 100 kobiet) w młodym wieku. Każdy model składa się z około 70000 wierzchołków. Na podstawie tych danych stworzony został morficzny model głowy, który płynnie może przechodzić z jednego modelu do kolejnego.



Rysunek 3.7: Schemat działania algorytmu.

Algorytm tworzący model trójwymiarowy działa na zasadzie dopasowania modelu trójwymiarowego do statycznego płaskiego obrazu twarzy. Poprzez przekształcenia morficzne przechodzi poprzez bazę wszystkich stworzonych wcześniej modeli i sprawdza, w jakim stopniu dopasowanie się udało.

W ten sposób stworzony model można następnie dowolnie modyfikować, zmieniać mimikę twarzy czy też symulować gesty. Można nadawać cechy specyficzne dla danej płci, cechy etniczne i inne.

Podsumowanie

Wadą rozwiązania jest fakt, że nigdy nie zrekonstruujemy cech, których nie ma na bazowych modelach. Rezultaty aplikacji są bardzo obiecujące i należą do najlepszych ze wszystkich opisanych w tym rozdziale rozwiązań. Przekształcenia morficzne są jednak dość skomplikowaną dziedziną, jednocześnie będąc bardzo czasochłonną i skomplikowaną obliczeniowo. Z tego też powodu nie nadaje się do stosowania w aplikacjach czasu rzeczywistego.

4. Opis zaprojektowanego systemu detekcji punktów kontrolnych oraz modelowania twarzy

4.1. Analiza twarzy

4.1.1. Kolejne etapy analizy twarzy

Analiza rozpoczyna się od lokalizacji twarzy na obrazie wejściowym. Klatka obrazu wstępnie zawiera twarz, lecz nie można od razu określić jej położenia ani orientacji. Jedyne, co założono to fakt, że twarz się tam znajduje i jest tylko jedna.

Do wyznaczenia dokładnego położenia twarzy na obrazie wykorzystana zostanie metoda dopasowania szablonu. Szablonem będzie zbiór obrazów testowych zawierających twarze kilkudziesięciu osób w różnym wieku, różnej postury oraz różnej karnacji, aby możliwe było jak najlepsze dopasowanie dla różnych osób.

Gdy uzyskana zostanie szukana pozycja kolejnym krokiem będzie wstępna analiza twarzy do określenia charakterystycznych obszarów twarzy, które poddane zostaną dalszej, szczegółowej analizie. Wstępna analiza opiera się na zastosowaniu obliczeń gradientów.

Kolejnym etapem detekcji jest dokładna analiza oczu. Po wstępnej analizie wyznaczony zostaje obszar, w którym należy poszukiwać dalszych szczegółów. Analiza oczu opiera się na wyznaczeniu gradientów oraz na zastosowaniu filtrów graficznych do znalezienia charakterystycznych cech.

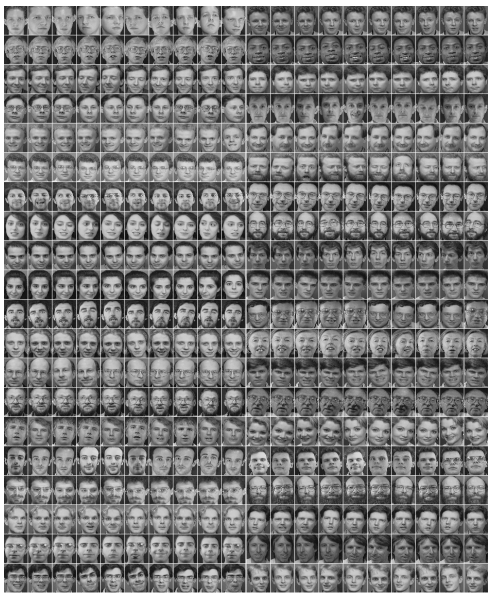
Fragment obrazu zawierający usta zostanie – podobnie jak ten zawierający oczy – poddany analizie poprzez obliczenie gradientów oraz zastosowanie filtrów graficznych, aby umożliwić łatwiejsze wyznaczenie punktów charakterystycznych.

Aby wyznaczyć punkty charakterystyczne opisujące geometrię brwi zastosowano serię filtrów graficznych. Następnie dodatkowe obliczenia, które pomogą wyznaczyć punkty skrajne będące punktami opisującymi ich anatomię.

4.1.2. Lokalizacja twarzy

Lokalizacja twarzy jest pierwszym etapem działania aplikacji. Celem tego etapu jest wyznaczenie regionu, w którym znajduje się twarz na obrazie wejściowym, który poddany zostanie dalszej analizie.

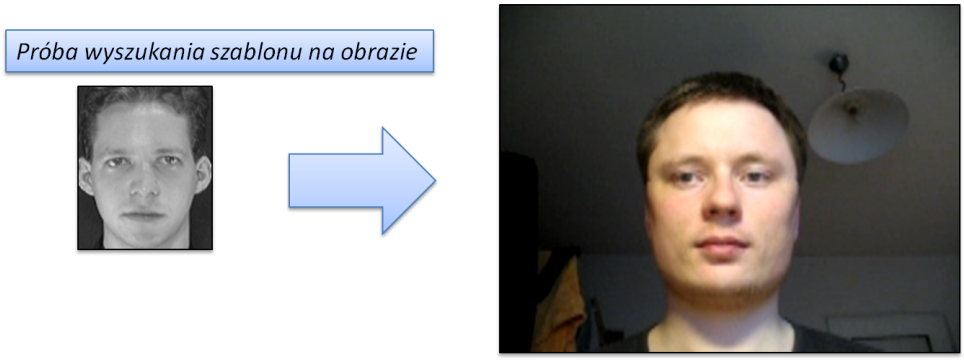
Obszar zawierający twarz wyznaczany jest na podstawie dopasowania szablonu. Algorytm korzysta ze zbioru kilkudziesięciu twarzy, na podstawie którego tworzy szablony i stara się dopasować je do obrazu wejściowego. Część bazy twarzy została przedstawiona na obrazku 4.1. Baza szablonów pochodzi z *AT&T Laboratories Cambridge* i można ją pobrać z [17]. Każda twarz jest niewielkich rozmiarów, co pozwala na zaakcentownie jedynie głównych cech charakterystycznych twarzy. Prowadzi to do tego, że współczynnik dopasowania będzie miał większą wartość dla obrazów znacznie różniących się detalami. Zdjęcia znajdujące się w bazie były robione w różnych warunkach oświetleniowych, a osoby przedstawione na nich ustawiane były w różnych pozycjach, pod różnym kątem. Osoby wykonują różne gesty twarzy – uśmiechają się, zamykają oczy, a także znaleźć można w bazie osoby w okularach. Taka różnorodność zwiększa szanse na trafność dopasowania szablonu, a tym samym na prawidłowe znalezienie twarzy na obrazie.



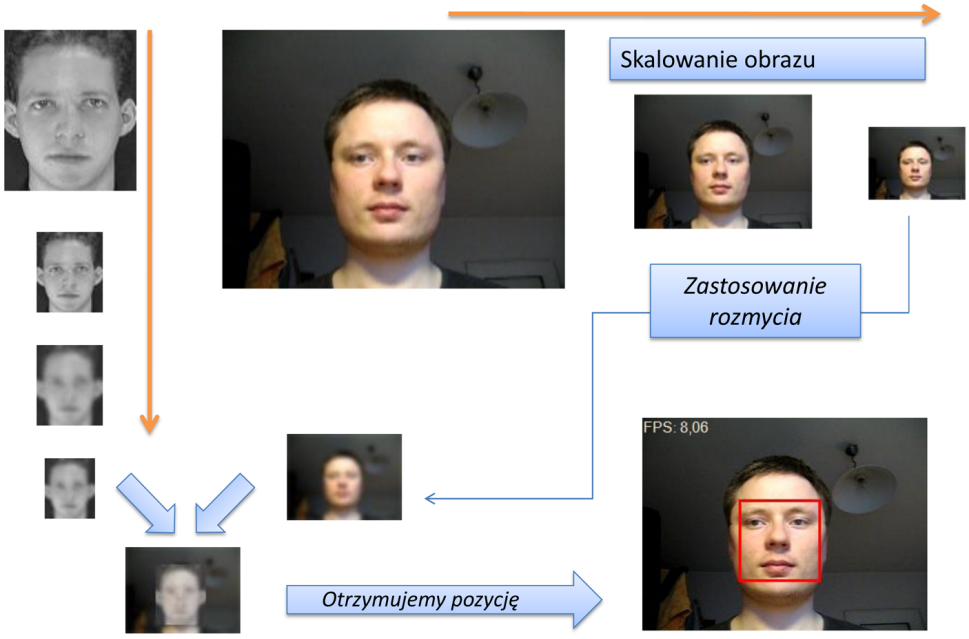
Rysunek 4.1: Zbiór szablonów twarzy.

Na rysunku 4.2 przedstawiono przykład, który ilustruje elementy wchodzące w skład dopasowania. Po lewej stronie znajduje się obraz szablonowy, który należy dopasować do obrazu wejściowego po prawej stronie. W wyniku dopasowania wyznaczone zostają współrzędne oraz wymiary prostokąta, który opisuje dopasowanie szablonu.

Na rysunku 4.3 przedstawiony został algorytm dopasowania krok po kroku.



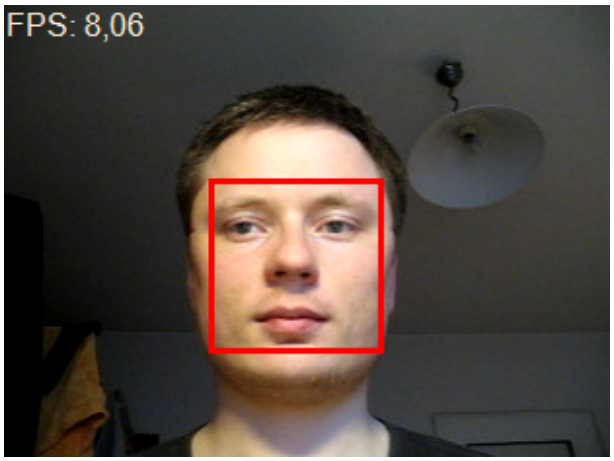
Rysunek 4.2: Poszukiwanie szablonu na obrazie wejściowym.



Rysunek 4.3: Kolejne etapy dopasowania szablonu.

Rysunek 4.4 przedstawia naniesiony prostokąt opisujący znaleziony fragment obrazu na obraz wejściowy. W tym momencie można przejść do dalszej analizy obrazu znajdującego się wewnątrz zaznaczenia. Niezależnie od pozycji twarzy na obrazie wejściowym, dalszej analizie poddawany będzie wyłącznie odnaleziony poszukiwany fragment obrazu.

Gdy już pozycja twarzy zostaje wyznaczona, ze względów wydajnościowych zostaje tworzony szablon na podstawie twarzy z obrazu wejściowego. Nie potrzebna jest już wtedy analiza i próba dopasowania obrazów szablonowych, których jest stosunkowo dużo. Wystarczy poszukiwać na obrazie znalezionej już wcześniej twarzy, co bardzo zwiększa wydajność, a zarazem pewność dopasowania jest większa, gdyż szablon powstał na podstawie obrazu oryginalnego. Jeżeli tylko stwierdzone zostaje, że pierwsze dopasowanie szablonu ze zbioru szablonów było prawidłowe, dalsze dopasowania są o wiele bardziej



Rysunek 4.4: Znaleziona twarz opisana prostokątem.

dokładne i precyzyjne.

Poprawa wydajności

Ponieważ baza szablonów zawiera stosunkowo dużą ilość zdjęć osób, które należy dopasować do obrazu wejściowego, algorytm może okazać się mało wydajny na wolniejszych komputerach. Aby poprawić wydajność poszukiwania twarzy na obrazie zastosowano algorytm wykrywający ruch elementów na nim się znajdujących. Tworzony zostaje *obraz ruchu*, który wyznaczany jest na podstawie klatki bieżącej oraz dwóch poprzednich. Obraz ten jest wynikiem operacji odejmowania obrazów, czyli w rezultacie otrzymujemy różnicę pomiędzy klatkami w sekwencji wideo.

Osoby siedzące przed kamerą internetową bądź nagrywane kamerą wideo wykonują pewne ruchy, nawet minimalne. Kamera natomiast w większości przypadków jest statyczna, położona na nieruchomym podłożu. W takich warunkach obraz wyznaczony poprzez poszukiwanie ruchomych fragmentów obrazu będzie zawierał głównie części należące do twarzy. W ten sposób wyznaczymy okno zawierające *ruchome piksele*, co pozwoli na szukanie szablonu i dopasowania wewnątrz wyznaczonego okna. W większości przypadków pozwala to na zaoszczędzenie ponad połowy czasu, jaki potrzebny jest na wykonanie algorytmu dopasowania szablonu.

Gdy dopasowanie szablonu nie nastąpi dla tak wyznaczonego okna obrazu bądź współczynnik dopasowania jest poniżej ustalonego progu, poszukiwanie powtarzane jest dla całego obrazu wejściowego. Jeżeli sytuacja powtarza się kilkakrotnie, algorytm poszukiwania ruchu zmniejszający okno poszukiwań zostaje wyłączony na stałe. Oznaczać to może występujące zakłócenia na obrazie lub ruchome elementy w tle za użytkownikiem. Przykładem może być ujęty na sekwencji wideo wiatrak kręcący się za plecami osoby filmowanej.

Kolejnym sposobem na poprawę wydajności rozpoznawania obrazów jest sposób opisany w [18],

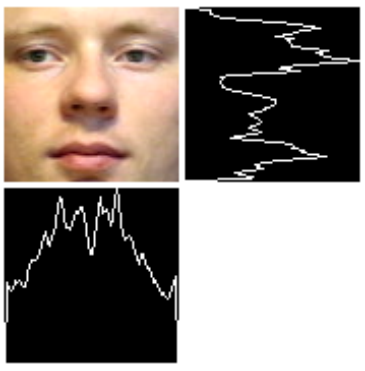
mianowicie analiza tylko wybranych klatek sekwencji. Jeżeli film wideo został nakręcony z wysoką ilością klatek na sekundę, analiza może stać się zbyt wolna, co spowoduje, że obraz wyjściowy będzie opóźniony w stosunku do obrazu wejściowego i co gorsze różnica ta będzie się powiększać w miarę dalszego przetwarzania.

Rozwiązanie zastosowane w niniejszej pracy polega na równoległym odtwarzaniu filmu wejściowego oraz jego przetwarzaniu w taki sposób, że w momencie przetwarzania, gdy procesor zajęty jest wykonywaniem algorytmu, klatki odtwarzane w tym samym czasie są gubione. Kolejną klatką obrazu, która zostanie przetworzona to ta, która pojawi się na wejściu w momencie, gdy algorytm zakończy pracę z poprzednią.

4.1.3. Wstępna analiza obszaru twarzy

Wstępna analiza obszaru twarzy służy klasyfikacji jego obszarów – podziałowi na regiony, w których na dalszych etapach dokonywane będzie poszukiwanie oczu, ust oraz brwi.

Do wyznaczenia charakterystycznych obszarów na obrazie twarzy zastosowano metodę gradientów. Na rysunku 4.5 pokazane zostały gradient poziomy oraz pionowy. Można zauważyć, że na poziomie ust oraz oczu gradient poziomy jest wyraźnie większy. Analizując gradienty pionowe można zauważyć, że na poziomie oczu mają większe wartości. Korzystając z cech charakterystycznych gradientów wstępnie wyznaczono poziom, na jakim znajdują się usta oraz oczy. Ta informacja zostanie wykorzystana w kolejnym etapie, w którym wyliczone będą punkty kontrolne. Aby zminimalizować czas potrzebny na przetwarzanie obrazu, dalsze operacje będą wykonywane na jak najmniejszych fragmentach obrazu wejściowego.



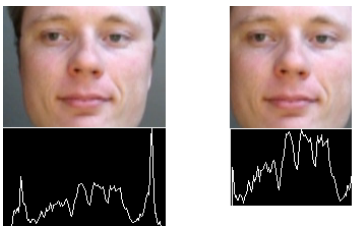
Rysunek 4.5: Wyznaczenie gradientu poziomego oraz pionowego dla obrazu twarzy.

4.1.4. Korekcja uzyskanej pozycji i geometrii twarzy

Analiza gradientu wyznaczonego dla obszaru twarzy jest pomocna również do oceny jego poprawności. Dla obrazu gdzie twarz jest lekko obrócona, przechylona bądź po prostu inaczej oświetlona z jednej strony, może wystąpić efekt przedstawiony na rysunku 4.6 – z brzegu widoczny jest stosunkowo duży obszar zakwalifikowany jako część twarzy. Błąd tego typu może prowadzić do nieprawidłowej detekcji elementów twarzy, między innymi oczu oraz brwi.

Rysunek 4.6: Obszar twarzy błędnie wyznaczony, wychodzący poza jej obszar.

Na rysunku 4.7 z lewej strony przedstawiono gradient pionowy dla obszaru twarzy poszerzonego o boczne marginesy. Łatwo można zauważyć, że w miejscu krawędzi, gdzie występują brzegi twarzy gradient jest znacznie większy, niż na pozostałej części obrazu. Przy analizie gradientu wyznaczonego wyłącznie dla obszaru twarzy bez dodatkowych, źle zakwalifikowanych marginesów – co przedstawiono na rysunku 4.7 po prawej stronie – widać, że wartości maksymalne gradientu znajdują się w środkowej części obrazu.



Rysunek 4.7: Gradient wyliczony dla obszaru twarzy z marginesami bocznymi (po lewej), oraz bez marginesów (po prawej).

Korekcja obszaru twarzy polega na wykryciu zwiększonych gradientów przy brzegach obrazu, eliminując w ten sposób błąd związany z wyznaczeniem większego obszaru twarzy, który zawiera fragment obrazu wykraczający poza szukany region. Algorytm działa w sposób poniższy:

1. Ustaw $x_1 = 0$ oraz $x_2 = 0$

2. Znajdź maksymalny gradient pionowy dla fragmentu obszaru od lewego brzegu do 30% szerokości obrazu.
3. Jeżeli odległość maksymalnego gradientu od lewego brzegu jest mniejsza od ϵ wykonaj: $x_1 = x_1 + \epsilon$
4. Znajdź maksymalny gradient pionowy dla fragmentu obszaru od 70% szerokości obrazu do jego prawego brzegu.
5. Jeżeli odległość maksymalnego gradientu od prawego brzegu jest mniejsza od ϵ wykonaj: $x_2 = x_2 + \epsilon$
6. Zmniejsz obszar analizy twarzy o x_1 z lewej strony oraz o x_2 z prawej strony.

Wartości zmiennych x_1 oraz x_2 zerowane są tylko raz w momencie uruchomienia analizy twarzy.

4.1.5. Detekcja oczu

Na tym etapie docelowo należy wyznaczyć dokładne położenie oczu oraz źrenic. Pierwszą czynnością, jaka zostanie wykonana to ponowne wyznaczenie gradientów. Różnica w stosunku do etapu poprzedniego polega na tym, że teraz wyznaczone zostaną wyłącznie dla obszaru zawierającego oczy. Na rysunku 4.8 zaprezentowano obliczone gradienty pionowe oraz poziome. Po wstępnej analizie wykresów można zauważyć, że podobnie jak w poprzednim przypadku dla całej twarzy, obszar oczu ma swoje charakterystyczne odzwierciedlenie na wykresach. Wyraźnie zwiększony gradient poziomy pozwala ustalić poziom, na jakim znajdują się oczy. Gradienty pionowe osiągają wartości maksymalne w punktach o współrzędnych znajdujących się w pobliżu źrenic. Niestety pozycja pozioma źrenic często jest lekko zaburzona, co może być spowodowane różnym ułożeniem twarzy, kolorem skóry, zmiennym oświetleniem. Z tego powodu należy przejść do dalszej, bardziej wnikliwej analizy, która pozwoli na bardziej skutecznie określenie pozycji źrenic.

Rysunek 4.8: Wyznaczenie gradientów dla fragmentu obrazu zawierającego oczy.

Po wyznaczeniu gradientów dla obszaru twarzy zawierającego oczy, wyodrębniamy je jako obrazy zawierające pojedyncze gałki oczne, a następnie stosujemy kolejno filtry graficzne przedstawione na rysunku 4.9. Kolejne kroki algorytmu:

- wyrównanie histogramu oraz korekcję kontrastu,
- konwersję na skalę szarości,
- binaryzację obrazu,
- odrzucenie skrajnych pikseli oraz wyliczenie środka masy.

Stosując wymienione powyżej operacje uzyskano punkt opisujący pozycję źrenicy.

Rysunek 4.9: Kolejne etapy zastosowania filtrów na fragmencie obrazu zawierającym oczy.

Zastosowanie metody Hough

Alternatywnym rozwiązaniem stosowanym do detekcji oczu w opisywanej aplikacji jest algorytm Hough wykrywający okręgi. Na podstawie wykrytego okna zawierającego twarz możliwe jest oszacowanie średnicy oka, co pozwoli dobrać odpowiedni parametr dla algorytmu. Dla obrazu, który zostaje poddany algorytmowi Hough, wstępnie dokonujemy wykrycia krawędzi metodą Canny'ego (opisana w rozdziale 2.3).

Dla dwóch powyższych rozwiązań rezultaty przedstawiają się różnie i zależą głównie od warunków otoczenia. Bardziej wnikliwa analiza i porównanie tych dwóch rozwiązań znajduje się w rozdziale 5.

4.1.6. Detekcja mrugnięcia

W dokumencie [19] opisano sposób bazujący na metodzie szablonowej. Podczas próby dopasowania szablonu oka sprawdza się dla kolejnych klatek współczynnik dopasowania (czyli wartość określającą jak dobrze dopasowanie zostało wykonane). Jeżeli wartość współczynnika nagle spada zakłada się, że wystąpiło mrugnięcie.

Ponieważ algorytm dopasowania szablonu jest procesem stosunkowo czasochłonnym, a pozycja oczu jest już na tym etapie znana, zastosowano nieco inne rozwiązanie.

Aby wykryć mrugnięcie potrzebnych jest kilka kolejnych klatek obrazu. Dla kolejnych klatek należy obliczyć wagę obrazu oka. Wprowadzono oznaczenia: N to szerokość obrazu w pikselach, M to jego wysokość, funkcja $f(x, y)$ zdefiniowana jest następująco:

$$f(x,y) = \begin{cases} 1 & \text{jeżeli punkt (x,y) jest częścią oka} \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{4.1}$$

W pierwszym kroku obliczone zostają wartości X_{sum} oraz Y_{sum}:

$$X_{sum} = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} g_x(f(x,y)) \tag{4.2}$$

$$Y_{sum} = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} g_y(f(x,y)) \tag{4.3}$$

gdzie funkcje g_x oraz g_y mają postać:

$$g_x(f(x,y)) = \begin{cases} x & \text{dla } f(x,y) = 1 \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{4.4}$$

$$g_y(f(x,y)) = \begin{cases} y & \text{dla } f(x,y) = 1 \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{4.5}$$

Następnie obliczamy wartości M_x oraz M_y, które opisują środek masy oka:

$$M_x = X_{sum}/c, \tag{4.6}$$

$$M_y = Y_{sum}/c, \tag{4.7}$$

gdzie c:

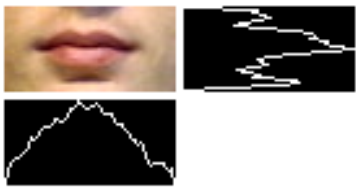
$$c = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y). \tag{4.8}$$

Łatwo wyobrazić sobie, że dla zamkniętego oka wartość c będzie dużo mniejsza, niż dla otwartego. Wartość wagi wyliczona dla kolejnych klatek obrazu jest porównywana i jeżeli różnica przekroczy ustalony próg to oznacza to, że wystąpiło mrugnięcie. Oprócz porównywania wartości c dla kolejnych klatek sprawdzane są wartości M_x oraz M_y. Podczas mrugnięcia następują znaczne ich skoki, co również wykorzystywane jest do kwalifikacji mrugnięcia.

4.1.7. Detekcja ust

Podobnie jak w przypadku analizy oczu, na wejściu przekazywany jest wyodrębniony obraz ust na podstawie obliczonych gradientów. Gdy obliczone zostają gradienty dla tego obszaru można przekonać się, że również w tym przypadku okażą się pomocne. Na rysunku 4.10 zaprezentowano wykresy gradientów poziomych oraz pionowych. Szczególnie przydatny okazuje się gradient poziomy wyznaczając bardzo dokładnie poziom ust, a właściwie połączenia się wargi dolnej z górną. Gradienty pionowe są

już mniej pomocne. Co prawda mogą posłużyć do wyznaczenia środka symetrii, ale w przypadku opisywanych w dalszej części metod nie zostanie to wykorzystane ze względu na zbyt małą dokładność odwzorowania geometrii ust.



Rysunek 4.10: Wyznaczenie gradientów dla fragmentu obrazu zawierającego usta.

Dalsza analiza opiera się na operacjach przedstawionych na rysunku 4.11, w kolejności:

- wyrównanie histogramu oraz zwiększenie kontrastu,
- konwersja na skalę szarości,
- binaryzacja oraz segmentacja.

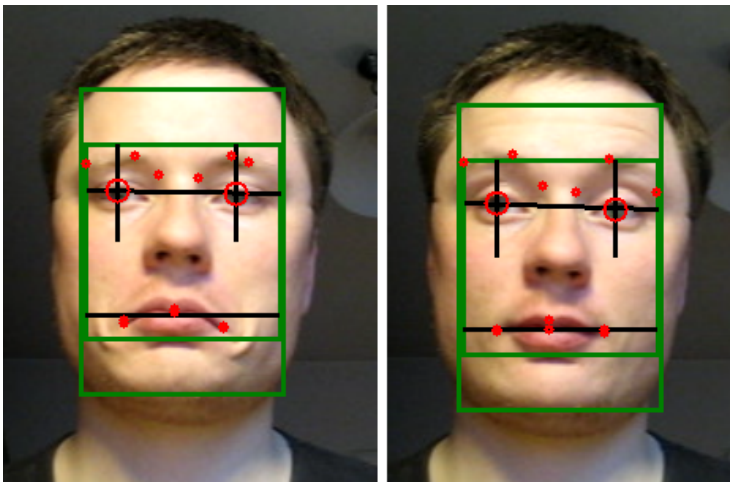
Ostatnia operacja z wymienionych – segmentacja – polega na wyodrębnieniu z obrazu binarnego jednego nieprzerwanego fragmentu, który w naszym przypadku jest przetworzonym fragmentem odpowiadającym górnej wardze. Ten właśnie kształt posłuży do wyznaczenia punktów charakterystycznych anatomii ust.

Rysunek 4.11: Kolejne etapy zastosowania filtrów na fragmencie obrazu zawierającym usta.

Punkty charakterystyczne wyznaczane są w trzech miejscach. Na brzegach po lewej oraz po prawej stronie, a także w połowie ust. Dla większej dokładności wyznaczane są dwa punkty. Jeden od górnej strony wargi, a drugi od jego dolnej strony. Na rysunku 4.12 zaznaczone są punkty wyznaczone powyżej opisanym algorytmem. Na rysunku widać, że na podstawie wyznaczonych punktów kontrolnych łatwo stwierdzić czy usta są proste, uśmiechnięte lub w jeszcze innym kształcie. Zbiór reguł opisujących wzajemne położenie punktów może wyznaczać gest, jaki dana osoba przedstawia.

4.1.8. Detekcja brwi

Detekcja brwi polega na zastosowaniu tego samego zestawu filtrów jak w przypadku ust z drobnymi modyfikacjami parametrów. Punkty kontrolne zostają wyznaczone również w trzech miejscach, z tą róż-



Rysunek 4.12: Wyznaczone punkty kontrolne ust naniesione na obraz.

nicą, że w przypadku brwi wystarczą skrajne górne punkty, ponieważ geometria brwi nie zmienia swojej szerokości w orientacji pionowej.

Rezultat detekcji punktów kontrolnych brwi można zobaczyć na rysunku 4.13. Po lewej stronie punkty naniesione są na obraz przetworzony, natomiast po prawej stronie na obraz oryginalny, wejściowy.

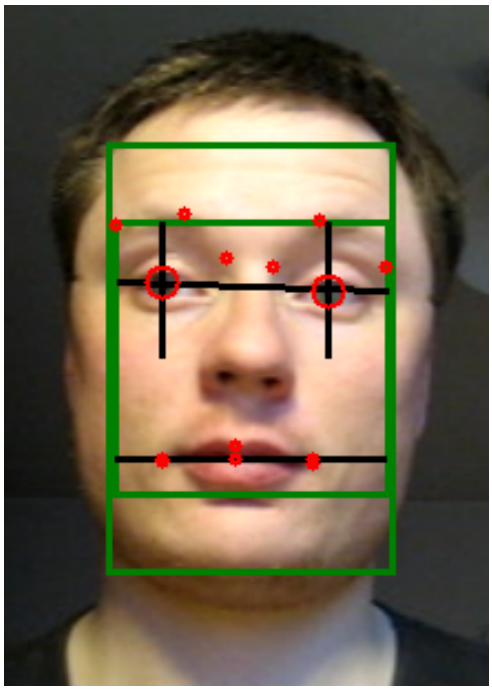
Rysunek 4.13: Wyznaczenie punktów kontrolnych brwi na obrazie przetworzonym oraz oryginalnym.

4.1.9. Rezultaty detekcji

Przechodząc przez wszystkie powyższe etapy detekcji otrzymano wszystkie istotne regiony twarzy oraz punkty charakterystyczne opisujące anatomię oczu, ust oraz brwi. Wszystkie elementy naniesione na obraz wejściowy wyglądają tak jak zostało to przedstawione na rysunku 4.14.

4.2. Modelowanie trójwymiarowe głowy

Modelowanie trójwymiarowe głowy na podstawie obrazu wejściowego to kolejny etap pracy. W rozdziale tym zostanie opisane, w jaki sposób punkty kontrolne wyznaczone wcześniej zostały przeniesione na przestrzenny model głowy. Poruszone zostaną również zagadnienia zastosowanych algorytmów do płynnej animacji twarzy oraz jak najlepszego odwzorowania głowy z oryginalnego obrazu.



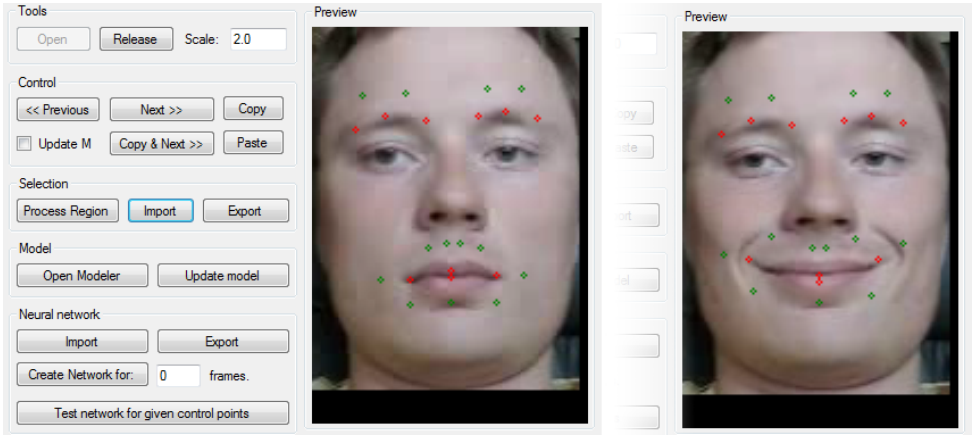
Rysunek 4.14: Obraz wejściowy z naniesionymi punktami kontrolnymi.

4.2.1. Sposób mapowania punktów

Model trójwymiarowy głowy, który odwzorowuje postać pozyskaną z obrazu wejściowego składa się z bardzo dużej ilości punktów. W stosunku do ilości punktów kontrolnych, jakie zostały wyznaczone w procesie analizy twarzy jest ich bardzo dużo. Należało więc znaleźć sposób, który pozwoliłby na podstawie punktów kontrolnych wyznaczyć pozycję (przemieszczenie) pozostałych punktów twarzy. W momencie wykonywania gestu twarzy takiego jak na przykład uśmiechu, na podstawie 4 punktów kontrolnych wyznaczonych na etapie analizy twarzy należy wyznaczyć pozycję wszystkich punktów na modelu trójwymiarowym, które powinny ulec przemieszczeniu podczas wykonywania tego gestu.

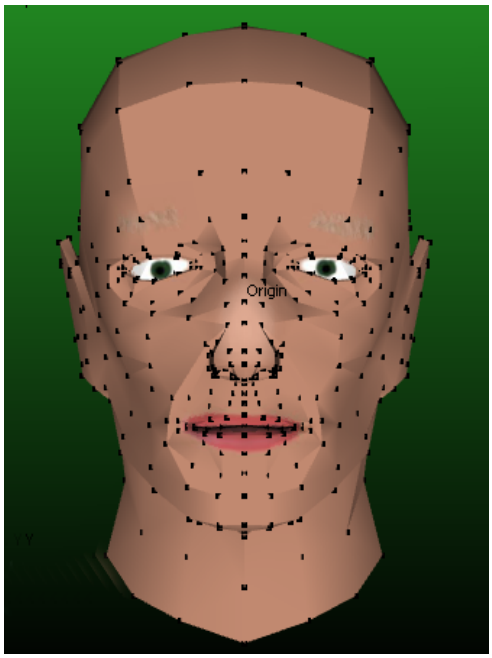
W celu rozwiązania tego problemu zastosowano sieć neuronową, która uczy się ruchu punktów będących w sąsiedztwie punktów kontrolnych. Do tego celu została stworzona aplikacja, w której mamy możliwość ręcznego definiowania pozycji punktów twarzy na podstawie punktów kontrolnych. Aplikacja pozwala stworzyć sieć neuronową a następnie nauczyć ją na podstawie ręcznie wyznaczonych punktów jak powinny się one przemieszczać podczas przemieszczania się punktów kontrolnych twarzy.

Aplikacja działa na takiej zasadzie, że pozwala otworzyć plik wideo, na którym jest nagrana twarz osoby wykonująca różnego rodzaju gesty twarzy, ruchy ust, oczu, brwi. Film wideo otwierany jest w trybie klatka po klatce tak, że możemy się dowolnie przesuwać po kolejnych klatkach filmu. W każdej klatce mamy możliwość umiejscowienia na obrazie twarzy punktów kontrolnych oraz punktów opisujących zachowanie się sąsiednich rejonów twarzy. Na rysunku 4.15 przedstawiono aplikację z klatkami



Rysunek 4.15: Aplikacja do tworzenia sieci neuronowej opisującej zachowanie się punktów twarzy na podstawie punktów kontrolnych.

obrazu z różnych fragmentów filmu. Na czerwono zaznaczono punkty kontrolne, które są uzyskane automatycznie poprzez analizę obrazu twarzy, natomiast punkty o kolorze zielonym należy umiejscowić ręcznie, zależnie od pozycji punktów czerwonych.



Rysunek 4.16: Model trójwymiarowy, na podstawie którego prezentujemy anatomię oraz dynamikę rozpoznanej twarzy.

Na rysunku 4.16 przedstawiono model trójwymiarowy, na podstawie którego przedstawiamy anatomię oraz dynamikę analizowanej twarzy. Czarnymi punktami zaznaczono wierzchołki modelu. Jest ich znacznie więcej niż punktów kontrolnych oraz punktów opisujących otoczenie punktów kontrolnych,

które pozyskane zostają za pomocą wcześniej przygotowanej sieci neuronowej. Aby poradzić sobie z tym problemem zastosowane zostało mapowanie punktów kontrolnych oraz tych pozyskanych z sieci na wierzchołki modelu głowy trójwymiarowego. Mapowanie wygląda następująco:

$$K_i \rightarrow \{\{P_1, W_1\}, \{P_2, W_2\}, \{P_3, W_2\}, \ldots, \{P_N, W_N\}\} \tag{4.9}$$

gdzie:

- K_i to wierzchołek modelu trójwymiarowego głowy posiadający współrzędne x oraz y,
- $P_1 \ldots P_N$ to punkty kontrolne bądź pozyskane z sieci również posiadające współrzędne x oraz y,
- W_i to waga określająca, w jaki sposób punkt P_i wpływa na dany wierzchołek K_i,
- N określa ile punktów wejściowych P ma wpływ na wierzchołek K_i.

Wyznaczenie pozycji $i - tego$ wierzchołka modelu trójwymiarowego na podstawie powyższego mapowania wygląda następująco:

$$K_i = \sum_{j=0}^{N} d_j W_j \tag{4.10}$$

gdzie:

- d_i to przesunięcie punktu K_i w stosunku do pozycji początkowej, czyli:

$$d_i = K_i - K_{pocz} \tag{4.11}$$

- $i \in \langle 1, M \rangle$,
- M to ilość mapowanych wierzchołków modelu.

Algorytm, który odwzorowuje ruchy wierzchołków modelu trójwymiarowego wygląda następująco:

1. Dla każdego mapowanego wierzchołka modelu wyznacz: $d_1, d_2, \ldots, d_M$ ze wzroru 4.11,
2. dla każdego punktu mapowania $P_1, P_2, \ldots, P_M$ wyznacz wierzchołek K za pomocą wzoru 4.10.

gdzie:

- M - ilość punktów kontrolnych oraz punktów otoczenia wyznaczonych za pomocą sieci,
- N - ilość wierzchołków, na jakie mapuje się dany punkt K.

Model głowy

Model głowy został wykonany w programie Blender3D [20]. Jest to program służący do tworzenia modeli trójwymiarowych, opartych na siatce wierzchołków. Przy pracy wspomagano się materiałami z książek [21] oraz [22], w których zawarte są cenne informacje na temat tworzenia cyfrowych postaci oraz techniki dotyczące animacji cyfrowych twarzy.

4.2.2. Dobór parametrów sieci

Sieć wykorzystywana w systemie to sieć wielowarstwowa, opisana w rozdziale 2.6.2. Sieć ta uczona jest metodą wstecznej propagacji błędu. Aby zapobiec pojawiającym się błędom uczenia, takim jak zatrzymywanie się algorytmu w minimum lokalnym funkcji błędu dla danej wagi, zastosowano rozwiązanie zaproponowane w [12]. Autor wspomnianej pozycji proponuje dodawać do wag małych losowych wartości po każdym kroku uczenia. Inną możliwością jest dodawanie przypadkowych, małych wartości do wektorów uczących.

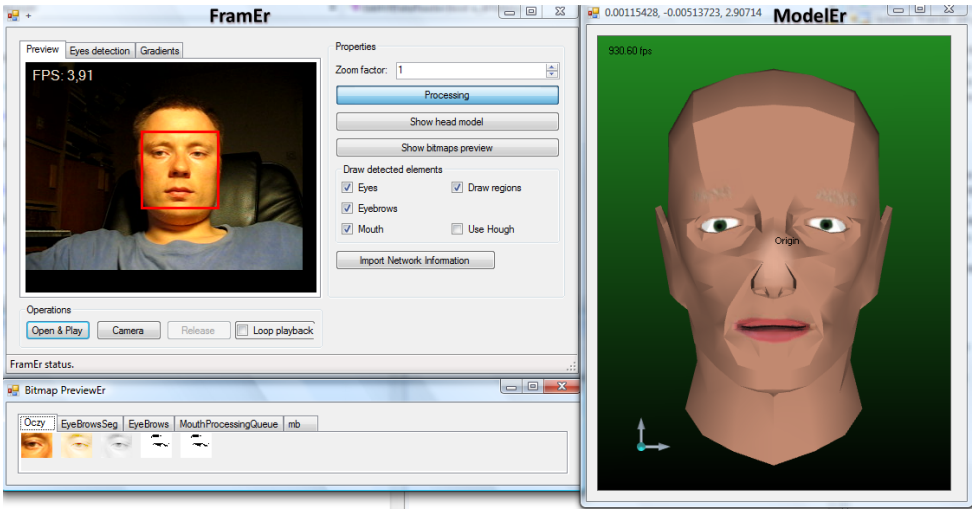
4.3. Projekt aplikacji

Algorytmy projektu zostały zaimplementowane w języku C# platformy .NET Microsoftu. Do niskopoziomowych operacji graficznych oraz przetwarzania obrazu wideo wykorzystano biblioteki zewnętrzne takie jak AForge [23] i OpenCV [24]. Aplikacja składa się z dwóch głównych, niezależnych modułów:

FramEr odpowiada za proces począwszy od dekompresji obrazu wideo, poprzez odnalezienie twarzy na obrazie, aż do wykrycia punktów kontrolnych.

ModelEr jest aplikacją, która na wejściu otrzymuje zbiór punktów kontrolnych i na tej podstawie tworzy trójwymiarowy model głowy, aktualizowany na bieżąco nowymi punktami kontrolnymi otrzymanymi od modułu *FramEr*.

Na rysunku 4.17 przedstawiono aplikację podczas działania. Po lewej stronie u góry znajduje się główne okno modułu *FramEr*. Okno zawiera kontrolki służące do sterowania aplikacją, wczytywania filmów z pliku, uruchamiania urządzenia wideo. Z poziomu *FramEra* możemy otworzyć główne okno drugiego modułu – *ModelEra*. Poniżej znajduje się okno diagnostyczne prezentujące kolejne etapy działania algorytmów, pojedyncze filtry stosowane dla elementów twarzy i inne. Po prawej stronie natomiast znajduje się główne okno modułu *ModelEr* prezentujące model trójwymiarowy głowy.



Rysunek 4.17: Aplikacja podczas detekcji twarzy.

4.3.1. FramEr

W skład tego modułu wchodzą kolejne podmoduły odpowiedzialne za operacje na niższych poziomach. Wybrane moduły wykorzystywane przez *FramEr*:

FdlFaceDetector - moduł wyszukujący twarz na zadanym obrazie korzystający z bazy twarzy, działający na zasadzie dopasowania szablonu;

CSFaceDetector - moduł oparty o algorytm CamShift do odnajdywania twarzy na obrazie;

AForgeVideoProviders - moduł służący do zarządzania urządzeniami wejściowymi, takimi jak kamery i kamerki internetowe, oraz umożliwiający odtwarzanie sekwencji wideo z plików;

HoughEyeDetector - moduł wykrywający oczy na obrazie wejściowym wykorzystujący algorytm Hough.

Aplikacja działa wielowątkowo pozwalając na wykorzystanie wszystkich możliwych zasobów komputera (w przypadku procesora wielordzeniowego zauważalny jest spory wzrost wydajności). Algorytm detekcji elementów twarzy działa równolegle poszukując niezależnych elementów twarzy. Detekcja lewego i prawego oka, ust, brwi – każdy z elementów analizowany jest w tym samym czasie.

Opisywany moduł bezpośrednio komunikuje się z modułem *ModelEr* przesyłając do niego wyznaczone punkty kontrolne. Dodatkowo w momencie znalezienia twarzy na obrazie zostaje przesłany rozmiar twarzy, tak aby moduł *ModelEr* mógł odpowiednio przeskalować model trójwymiarowy by uzyskać jak najlepsze podobieństwo do analizowanej twarzy.

4.3.2. ModelEr

ModelEr jest modułem odpowiedzialnym za tworzenie i animację modelu trójwymiarowego głowy. Moduł ten również wczytuje sieć neuronową stworzoną w aplikacji *VertexEr* opisanej poniżej. Na jej podstawie jest w stanie obliczyć pozycję punktów ruchomych twarzy opartych na ruchach punktów kontrolnych.

Moduł ten bazuje na bibliotece graficznej OpenGL [25], której używa do renderowania grafiki trójwymiarowej, nakładania tekstur oraz animacji.

4.3.3. VertexEr

VertexEr jest niezależną aplikacją służącą do stworzenia sieci neuronowej i nauczenia jej. Aplikacja wykorzystuje część modułów *FramEra*, takich jak moduł do obsługi urządzeń i plików wideo. Do operacji związanych z tworzeniem i uczeniem sieci neuronowej wykorzystano bibliotekę *NeuronDotNet* [26].

5. Omówienie wyników przeprowadzonych doświadczeń

5.1. Działanie systemu w różnych warunkach

Aby sprawdzić poprawność działania systemu, przeprowadzono testy algorytmów w różnych warunkach. Stworzono materiały filmowe w warunkach naturalnego dziennego oświetlenia, porównując rezultaty z filmami nagranymi przy oświetleniu sztucznym. Zbadano również wpływ dodatkowych elementów zakrywających twarz. Ostatnim przeprowadzonym testem był sprawdzian zachowania się detektora twarzy dla różnych osób.

5.1.1. Naturalne oświetlenie dzienne

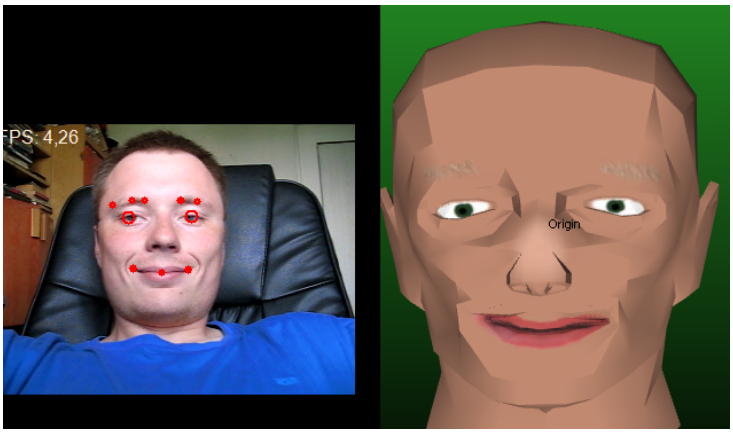
Z przeprowadzonych doświadczeń wynika, że algorytmy spisują się najlepiej dla sekwencji filmowych nagrywanych w ciągu dnia przy naturalnym oświetleniu. Obrazy są mniej zaszumione, szczegóły obrazu są bardziej wyeksponowane, a także nagłe ruchy występujące na filmie nie rozmazują obrazu. Wszystko to wpływa na działanie algorytmu w pozytywny sposób. Detekcja twarzy jest niezaburzona i punkty kontrolne odnajdywane są z największą dokładnością.

Na rysunku 5.1 przedstawiono zrzut ekranu z działania aplikacji. Obraz wideo jest bardzo jasny, dobrze naświetlony. Na rysunku przedstawiono twarz z filmu wejściowego z zaznaczonymi punktami kontrolnymi (w kolorze czerwonym), natomiast obok po prawej stronie zobaczyć można stworzony model trójwymiarowy.

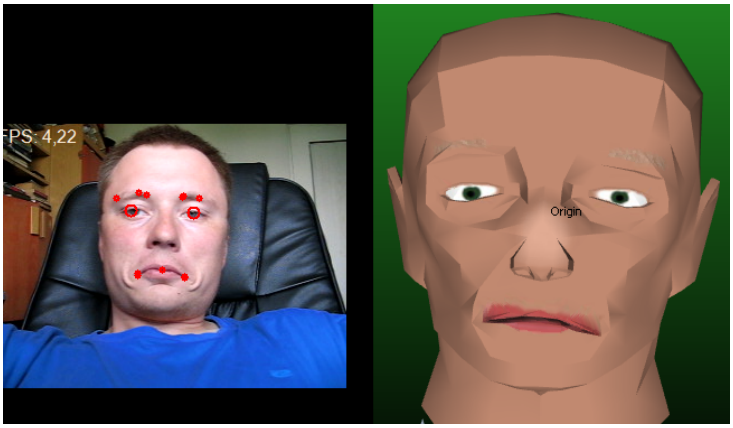
Na rysunku 5.2 przedstawiono zrzut ekranu z aplikacji przetwarzającej ten sam film co powyżej, z tą różnicą, że przedstawiono na nim osobę z inna mimiką twarzy, próbującą wykonać gest śmutny". Na modelu trójwymiarowym po prawej stronie można zauważyć, że usta są uformowane w podobnym kształcie. Dodatkowo warto zwrócić uwagę na ułożenie głowy – na modelu trójwymiarowym odtworzono lekkie pochylenie głowy w prawą stronę.

Podsumowanie

Dla naturalnych warunków oświetleniowych oraz dla nieprzysłoniętej niczym twarzy błędy w detekcji właściwie nie występowały w procesie testowania algorytmu. Można więc przyjąć, że opisane



Rysunek 5.1: Rezultat działania aplikacji w naturalnych warunkach oświetlenia.



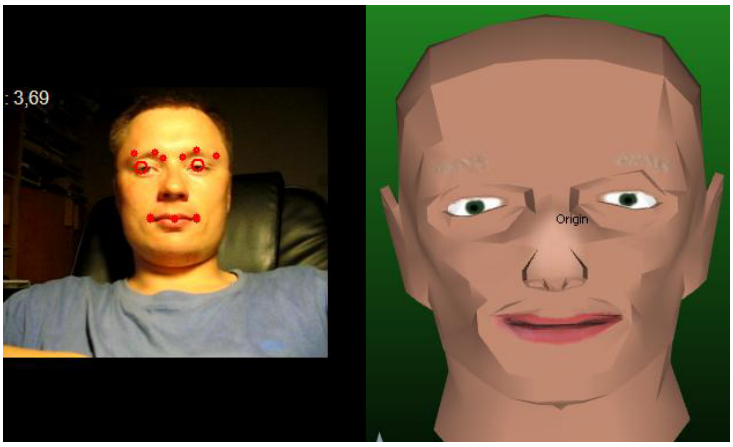
Rysunek 5.2: Rezultat działania aplikacji w naturalnych warunkach oświetlenia. Wykonano gest twarzy *smutny*.

warunki, w jakich został nagrany film wideo dają najlepsze rezultaty dla opisanego w pracy algorytmu detekcji twarzy oraz wykrywania punktów kontrolnych.

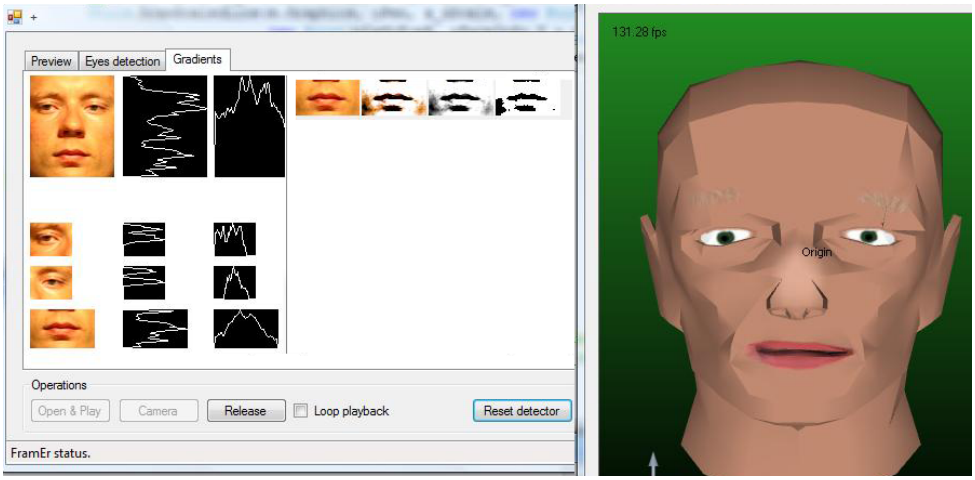
5.1.2. Oświetlenie sztuczne

Badając zachowanie się algorytmu w trudniejszych warunkach oświetleniowych, to znaczy przy sztucznym, ciemniejszym świetle, uzyskano nieco gorsze rezultaty, niż w podrozdziale poprzednim (5.1.1). Filmy stworzono w godzinach wieczornych oświetlając pomieszczenie lampką biurową.

Na rysunku 5.3 przedstawiono zrzut ekranu z działania aplikacji analizującej film nakręcony w warunkach sztucznego oświetlenia. Zauważalnie gorsza jakość filmu, mniej czytelny obraz i mniejsza ilość detali ma znaczący wpływ na działanie algorytmów.



Rysunek 5.3: Przykład działania aplikacji w warunkach sztucznego oświetlenia.



Rysunek 5.4: Działanie algorytmów w warunkach sztucznego oświetlenia. Wykresy gradientów elementów twarzy oraz kolejne etapy przetwarzania elementów twarzy.

5.1.3. Ruchoma kamera

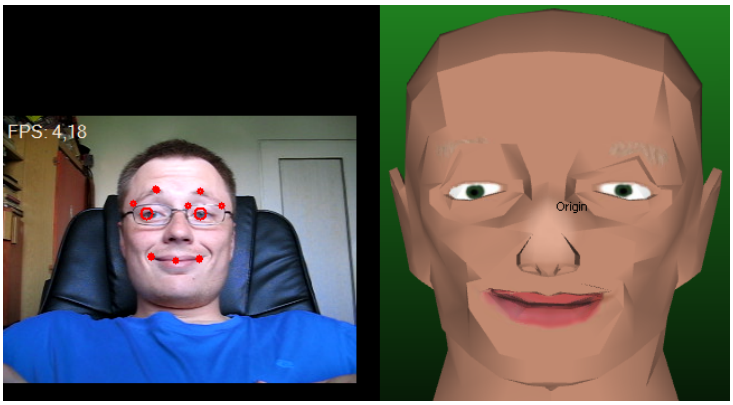
Ruch kamery oraz samej twarzy powoduje błędy w działaniu algorytmów, ponieważ obraz tak uzyskany jest zazwyczaj rozmyty i nie oddaje wiernie rzeczywistości. Jeżeli można wykorzystać kamerę lepszej jakości, rejestrującej obraz z dużą ilością klatek na sekundę, błędy będą zdecydowanie mniej odczuwalne. Testy przeprowadzane były na kamerce internetowej oraz na średniej klasy aparacie fotograficznym, tym samym jakość obrazu nie była zadowalająca. Na obrazku 5.5 przedstawiono zrzut ekranu przedstawiający działanie aplikacji przetwarzającej film nagrywany ruchomą kamerą. Zauważyć można liczne błędy w detekcji punktów kontrolnych. Po lewej stronie (*rysunek a*) widzimy zrzut ekranu wykonany w momencie, gdy na filmie kamera wykonuje dość dynamiczny ruch. Punkty kontrolne są bardzo zniekształcone, co w zasadzie uniemożliwia prawidłowe rozpoznanie cech twarzy.

Rysunek 5.5: Wpływ ruchu kamery na jakość detekcji punktów kontrolnych.

W momencie gdy kamera przestaje się poruszać i obraz stabilizuje się, algorytmy odzyskują stabilność i punkty kontrolne powracają na swoje miejsca, a dalsza detekcja jest kontynuowana poprawnie. Na rysunku 5.5 po prawej stronie (*rysunek b*) ruch kamery stabilizuje się – punkty kontrolne wracają do prawidłowych miejsc.

5.1.4. Wpływ dodatkowych czynników zewnętrznych

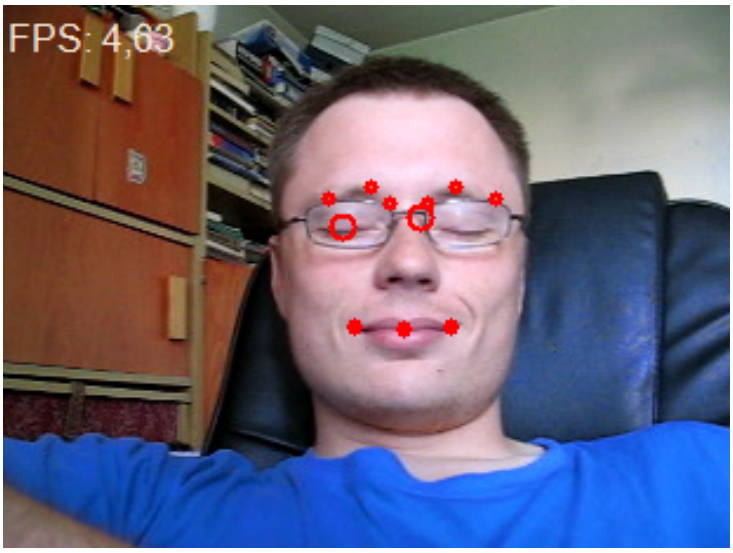
W sekcji tej przedstawiono wpływ elementów zakrywających twarz na poprawność detekcji twarzy przez algorytm. Badań dokonano stosując okulary przysłaniające część twarzy i wpływające na znaczną zmianę obrazu twarzy.



Rysunek 5.6: Detekcja twarzy osoby w okularach.

Na rysunku 5.7 przedstawiono zrzut ekranu z aplikacji podczas analizy twarzy osoby w okularach, ale tym razem z zamkniętymi oczami. Można zauważyć, że po pierwsze nie nastąpiła detekcja mrugnięcia. Drugim błędnym zachowaniem algorytmu jest przesunięcie punktu kontrolnego prawego oka na podstawę nosa. Na błędne wyznaczenie punktu kontrolnego duży wpływ miała czarna ramka okularów, która spowodowała nieprawidłowe wyznaczenie wag źrenic oraz filtrów wykrywających krawędzie.

Przeprowadzone eksperymenty wykazały, że w przypadku detekcji twarzy osoby, która nosi okulary,



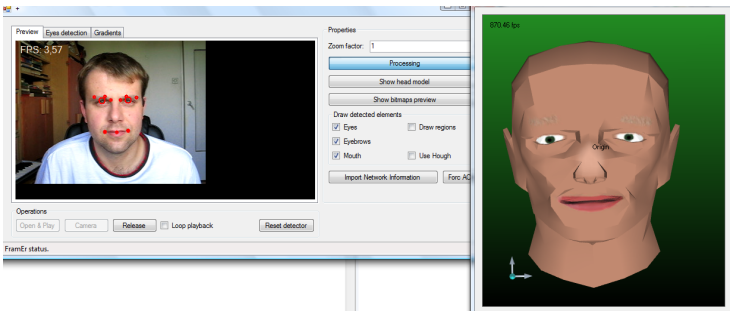
Rysunek 5.7: Zaburzone wyniki algorytmu detekcji twarzy osoby w okularach.

lepiej sprawdza się algorytm wyszukiwania oczu oparty na metodzie Hough. Ramki okularów zazwyczaj nie wpływają negatywnie na kontury oka ani źrenicy. Mają wpływ natomiast na algorytmy poszukujące krawędzi, co powoduje błędne zachowanie się algorytmów wykrywania brwi.

5.1.5. Detekcja twarzy dla różnych osób

W tym podrozdziale opisane zostały testy systemu wykonane przy udziale kilku różnych osobach. Doświadczenia przeprowadzane były w różnych warunkach oświetlenia oraz w różnych miejscach. Najciekawsze z uzyskanych rezultatów przedstawiono na rysunkach oraz opisano poniżej.

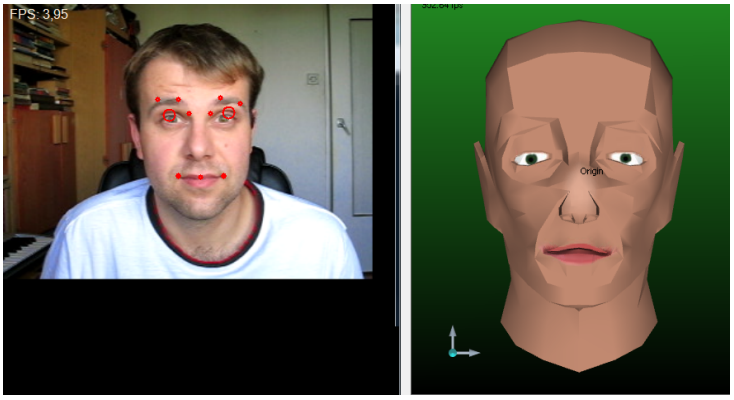
Na rysunku 5.8 przedstawiono osobę, która nie wykonuje żadnego gestu ani miny. Po prawej stronie przedstawiony jest stworzony model trójwymiarowy. Detekcja twarzy oraz punktów kontrolnych przebiegła pomyślnie, a śledzenie twarzy było prawidłowe w 90% czasu pojedynczego nagrania.



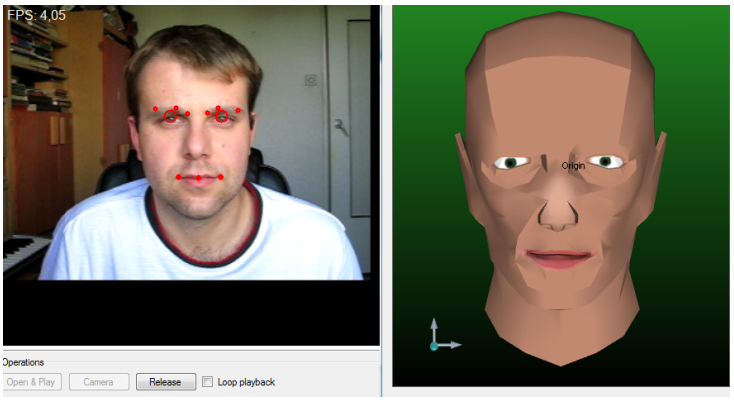
Rysunek 5.8: Testowanie systemu dla różnych osób – wyraz twarzy naturalny.

Na rysunkach 5.9 oraz 5.10 przedstawiono osobę wykonującą gesty, odpowiednio: uniesienie brwi

oraz przymrużenie oczu. Na modelu trójwymiarowym można zaobserwować jak zostały dane gesty odwzorowane. Można zauważyć, że punkty kontrolne zostały poprawnie wykryte oraz przedstawione na trójwymiarowym modelu, przybliżając układ brwi osoby analizowanej.



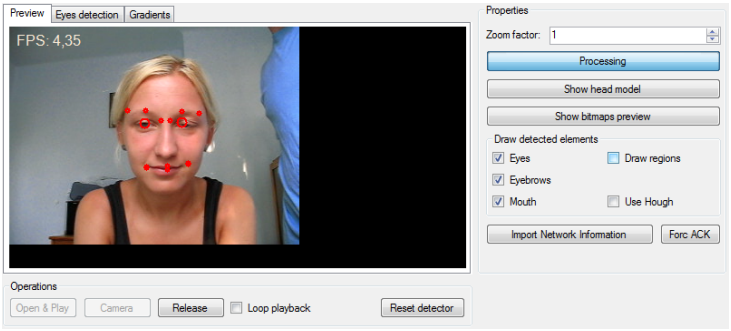
Rysunek 5.9: Testowanie systemu dla różnych osób – uniesione brwi.



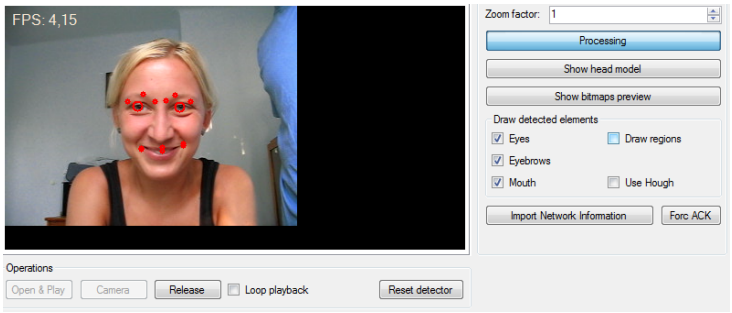
Rysunek 5.10: Testowanie systemu dla różnych osób – przymrużone oczy.

Na rysunkach 5.11 oraz 5.12 przedstawiono następną osobę na obrazie wideo nakręconym w innych warunkach otoczenia. Pierwszy z nich przedstawia naturalną minę oraz rozkład punktów kontrolnych. Drugie z przedstawionych rysunków przedstawia osobę wykonującą uśmiech i analogicznie – ułożenie punktów kontrolnych. W tym przypadku czas wykrycia twarzy znacząco się wydłużył ze względu na występujące na filmie zakłócenia. Poprawność śledzenia wynosiła około 80% – przy szybszych ruchach głowy niektóre punkty na pewien czas traciły swoje prawidłowe położenie. Błednie zlokalizowane punkty kontrolne po kilku sekundach wracały na odpowiednią pozycję.

Kolejny test przeprowadzono w nieco trudniejszych warunkach oświetleniowych – przy silnym słońcu i częściowo zasłoniętych zasłonach w pomieszczeniu, co sprawiło, że obraz stał się mało kontrastowy i wyraźny, miejscami zaciemniony, natomiast w tle widać było dużo odblasków światła. Próba

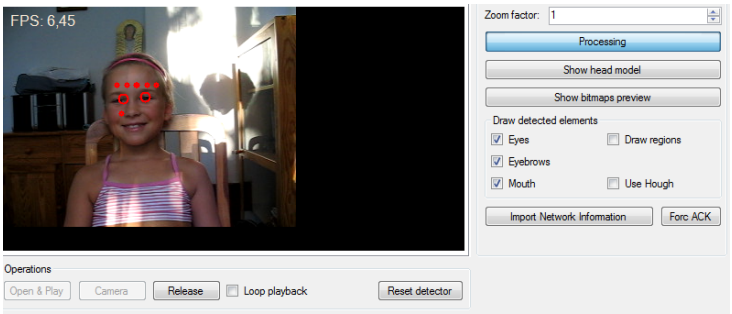


Rysunek 5.11: Testowanie systemu dla różnych osób – wyraz twarzy naturalny.



Rysunek 5.12: Testowanie systemu dla różnych osób – wyraz twarzy uśmiechnięty.

analizy w takich warunkach kolejnej osoby przedstawiona została na rysunku 5.13. Niestety system nie działał prawidłowo, wykrywając poprawnie jedynie oczy. Usta nie zostały w ogóle wykryte, natomiast punkty kontrolne brwi nie pokrywały się z brwiami na obrazie wejściowym.



Rysunek 5.13: Testowanie systemu dla różnych osób – błędne wykrycie ust oraz brwi.

Podsumowanie

Testy przeprowadzone na różnych osobach, w różnych warunkach pokazują, że algorytmy radzą sobie całkiem dobrze z różnymi cechami szczególnymi odpowiednimi dla różnych osób różnych płci. Przeszkodą w prawidłowym funkcjonowaniu mogą okazać się utrudnione warunki oświetleniowe. Jak

wykazały testy, nadmierne oświetlenie naturalne lub sztucznie wytworzone cienie mogą okazać się dużo trudniejszymi warunkami, niż sztuczne oświetlenie w środku nocy. Najmniej podatną na zakłócenia częścią algorytmu jest ta, która odpowiada za detekcje oczu. Najbardziej wrażliwą na zakłócenia i wpływ środowiska zewnętrznego jest detektor ust.

5.1.6. Analiza błędnych rozwiązań

Głównymi czynnikami powodującymi zaburzoną pracę algorytmów, co w efekcie objawiało się źle wyznaczonymi punktami kontrolnymi, są:

- słaba jakość oświetlenia,
- zakłócenia występujące na obrazie wejściowym,
- elementy przysłaniające twarz.

Jakość oświetlenia może zostać poprawiona poprzez odpowiednią, wstępną poprawę kontrastu obrazu oraz wyrównanie histogramu. Pomocny mógłby okazać się również algorytm wyostrzający obraz. Zakłócenia występujące na obrazie można próbować zniwelować stosując filtry rozmywające (dolnoprzepustowe, filtr Gaussa) bądź filtr medianowy z odpowiednio dobraną maską.

Elementy przysłaniające twarz są najtrudniejszym występującym problemem. Trudność polega na tym, że trudno jest przewidzieć, jakiego rodzaju element występuje na obrazie, oraz sposób jego identyfikacji. Próba korekcji obrazu może polegać na przybliżonym odtworzeniu obrazu znajdującym się w przysłoniętym miejscu bądź – jeżeli to możliwe – pominięcie danego fragmentu obrazu i kontynuacja dalszego przetwarzania.

Z zakłóceniami spowodowanymi przez okulary można sobie radzić na kilka sposobów. Okulary są powszechnie stosowane, więc można stworzyć osobny algorytm do ich wykrywania, a następnie odpowiedniej interpretacji i analizy takiego obrazu.

5.1.7. Podsumowanie

Jednym z wyznaczników jakości algorytmu wykrywania twarzy może być zmierzenie czasu od momentu rozpoczęcia analizy twarzy do momentu akceptacji – czyli do znalezienia twarzy na obrazie wejściowym i rozpoczęcie wyszukiwania punktów kontrolnych. W tabeli 5.1 przedstawiono wyniki dla poszczególnych testów. Każdy rodzaj testu przeprowadzany był na trzech sekwencjach wideo.

Wyniki uzyskane z przeprowadzonych pomiarów można interpretować w następujący sposób:

Tablica 5.1: Czasy wykrycia twarzy dla poszczególnych warunków otoczenia.

Rodzaj testu	**Średni czas akceptacji**
Światło naturalne	9s
Światło sztuczne	16s
Osoba w okularach, światło naturalne	14s
Ruchoma twarz, światło naturalne	28s

- Sztuczne oświetlenie wprowadza dodatkowe szumy, co powoduje wydłużenie czasu akceptacji, obszar twarzy zostaje zniekształcany poprzez zakłócenia, co powoduje, że algorytm dopasowania szablonu zwraca różne wartości współczynnika dopasowania dla kolejnych klatek.

- Osoba w okularach powoduje zniekształcenie obrazu, ale nie wpływa to w znaczący sposób na czas akceptacji. Poprawę detekcji punktów kontrolnych oczu uzyskujemy stosując metodę Hough.

- Ruchoma twarz, bądź ruch kamery w trakcie wyszukiwania twarzy na obrazie bardzo znacząco wydłuża okres akceptacji. Aby zapewnić optymalny czas wykrycia twarzy, należy starać się zapewnić jak najbardziej statyczny obraz w tej początkowej fazie działania algorytmu.

5.2. Możliwości dalszego rozwoju algorytmu i aplikacji

Prezentowane rozwiązanie i zastosowane algorytmy są bazą do dalszych prac. Możliwości rozbudowy jest bardzo wiele. Jednym z ciekawych zagadnień pozwalających poprawić skuteczność detekcji elementów twarzy jest transformata falkowa, która może być wykorzystywana do odnajdywania oraz śledzenia cech. Temat szerzej opisany jest w [8].

Model trójwymiarowy głowy oraz algorytmy animacji mogą być usprawnione, tak aby animacja była płynniejsza. Wprowadzenie dodatkowych punktów wygładzi prezentowaną twarz, która stanie się bardziej realistyczna. Bardzo wiele szczegółowych detali twarzy może zostać poprawiona lub dodana.

Aplikacja jest również dobrą bazą do stworzenia aplikacji tworzącej model trójwymiarowy postaci, która naśladuje gesty i zachowania osoby analizowana na obrazie wideo. Odpowiednia interpretacja punktów kontrolnych oraz ich przemieszczeń może pozwolić na dogłębną analizę mimiki osoby i próby naśladowania jej uczuć.

6. Podsumowanie

Skomplikowane zagadnienie, jakim jest rozpoznawanie obrazów, może zostać zaimplementowane przy użyciu stosunkowo nieskomplikowanych algorytmów, co przedstawiono w niniejszej pracy. Zadowalające rezultaty można osiągnąć bez używania skomplikowanych i złożonych obliczeń. Biorąc pod uwagę moc obliczeniową komputerów wrastającą z dnia na dzień, uzyskujemy coraz to szersze możliwości tworzenia bardzo zaawansowanych systemów detekcji oraz analizy. Zastosowane w pracy algorytmy są wystarczająco szybkie, aby system mógł działać w czasie rzeczywistym. Zaprojektowany system działa w pełni automatycznie i nie wymaga od użytkownika żadnej kalibracji ani wstępnej konfiguracji. Algorytmy są tak opracowane, aby radziły sobie z zakłóceniami występującymi stosunkowo często przy procesie nagrywania twarzy w różnych warunkach. System został przetestowany w różnych warunkach oświetlenia i o różnych porach dnia. Algorytmy spełniają swoje zadanie zarówno przy świelte dziennym jak i sztucznym. Przy różnym oświetleniu mogą pojawić się różnego rodzaju refleksje oraz cienie, co może spowodować zmianę obrazu twarzy. Jeżeli zacienienie obrazu nie jest bardzo znaczące i elementy twarzy są widoczne, system jest w stanie wykryć elementy twarzy poprawnie. Jakoś detekcji zależna jest od jakości obrazu wejściowego, od tego czy obraz jest zakłócony i w jakim stopniu. Duży wpływ na poprawność detekcji ma dynamika osoby siedzącej przed kamerą oraz samej kamery. Im więcej ruchu, tym bardziej zniekształocny obraz. Detekcja punktów kontrolnych może być mocno zaburzona poprzez ruchy kamery, jednakże w zdecydowanej większości przypadków kamera jest statyczna a ruch występuje jedynie na filmowanym obrazie, a na takiego rodzaju zakłócenia system jest o wiele bardziej odporny. Wstępnie postawione założenia pracy udało się spełnić, tworząc w pełni automatyczny system działający w czasie rzeczywistym. Rezultatem działania algorytmów zaimplementowanych w ramach niniejszej pracy jest trójwymiarowy model, który przybliża anatomię i dynamikę osoby analizowanej. Testy zaprezentowały różne aspekty odwzorowania cech anatomicznych badanych osób oraz sposób prezentacji gestów i min przez trójwymiarowy model twarzy.

Wszechobecne media, nieograniczone zasoby Internetu oraz coraz szybsze łącza stają się niezastąpionym nośnikiem już nie tylko tekstu i statycznego obrazu, ale coraz częściej obrazu wideo w wysokiej rozdzielczości. Coraz popularniejsze stają się internetowe serwisy informacyjne, które udostępniają strumień wideo przedstawiający relacje na żywo. Coraz więcej powstaje serwisów pozwalających na łatwe

gromadzenie, wymianę pików wideo, a nawet na prowadzenie dzienników poprzez nagrania multimedialne. W ten sposób zasypywani jesteśmy multimedialnymi informacjami. Sytuacja ta wpływa bardzo korzystnie na pojawianie się nowych sposobów analizy takich danych. Sprzyja to rozwojowi nie tylko nowych technik kompresji oraz kodowania wiadomości multimedialnych, ale również algorytmy detekcji i analizy są ciągle ulepszane. Powstaje masa nowych rozwiązań, w gruncie rzeczy opartych na prostych i szybkich algorytmach graficznych.

Rezultaty jakie uzyskano w niniejszej pracy mogą być wykorzystane przy opracowywaniu nowoczesnych interfejsów HCI (ang. *Human-Computer Interaction*). Wyobraźmy sobie wirtualną postać rozmawiającą z użytkownikiem, zamiast tekstowych poleceń – sposób ten z pewnością jest bliższy naturalnej konwersacji i komunikacji. Dziedzina projektowania interakcji rozwija się bardzo szybko i opera się na projektowaniu zorientowanym na użytkownika. Zastosowanie awataru jest jednym ze sposobów, by nawiązać kontakt z użytkownikiem, sprawić wrażenie, że system interesuje się klientem. Tego typu rozwiązania stosuje się również w czatbotach oraz nowoczesnych systemach CRM (ang. *Customer Relationship Management*). Te pierwsze mogą znaleźć szerokie zastosowanie w systemach ekspertowych, czyli systemach posiadających bazę wiedzy służących do diagnozowania rozmaitych problemów i usterek. System taki może być zamiennikiem biura pomocy (ang. *Help Desk*), gdzie program stara się uzyskać niezbędne informacje od użytkownika, a pośrednikiem informacji jest awatar. Dodając do tego system analizujący twarz użytkownika i wykrywający jego gesty oraz emocje, można analizować zachowania oraz reakcje osoby, przez co trafniej można sterować konwersacją i lepiej dobierać pytania. Kontekstem z jakiego czerpane są informacje to już nie tylko odpowiedzi na pytania bota[1], ale również zachowania użytkownika oraz jego reakcje na dane pytania.

Zaprezentowane tutaj rozwiązania oraz problemy, z jakimi należy się zmierzyć podczas próby analizy ludzkiej twarzy dają pogląd na to, że bardzo trudno jest stworzyć system działający bezbłędnie. Detekcja ludzkiej twarzy to temat bardzo rozległy i badany już od dawna. Techniki analizy i detekcji twarzy w dużej mierze opracowywane są przez instytuty zajmujące się bezpieczeństwem. Twarz jest w zasadzie kluczowym elementem, który może służyć do identyfikacji osób nagranych przez kamery systemu bezpieczeństwa. Tutaj także algorytmy czasu rzeczywistego mają duże znaczenie – w sytuacji, gdy kamery umieszczone są w miejscu publicznym może zaistnieć konieczność weryfikacji osób znajdujących się na obrazie zanim zdążą opuścić dane miejsce.

Z biegiem czasu, gdy komputery osiągną jeszcze wyższe moce obliczeniowe, a otaczające nas media staną się jeszcze bardziej wszechobecne, przewidywać można bardzo szybki rozwój dziedziny analizy i rozpoznawania obrazów – zarówno na komputerach, ale również na coraz bardziej rozbudowanych telefonach komórkowych. Techniki rozpoznawania obrazów i ich analizy, a także sposób prezentacji

[1] program wykonujący pewne czynności w zastępstwie człowieka

informacji w sposób trójwymiarowy znajdują coraz to nowe zastosowania.

Z pewnością nie sposób przewidzieć wszystkich możliwości dalszego rozwoju dziedziny, nie mniej jednak śledząc na bieżąco postępy, jakie są dokonywane w tej tematyce można spodziewać się bardzo zaskakujących rezultatów w niedalekiej przyszłości.

Bibliografia

[1] G. Bradski. Computer vision face tracking for use in a perceptual user interface. 1998. Santa Clara, CA, Intel Corporation, Microcomputer Research Lab.

[2] R. Tadeusiewicz and P. Korohoda. *Komputerowa analiza i przetwarzanie obrazów*. Wydawnictwo Fundacji Postępu Telekomunikacji, Krakow, 1997.

[3] R. Tadeusiewicz and M. Flasiński. *Rozpoznawanie obrazów*. PWN, Warszawa, 1991. (dostępna pod adresem: `http://winntbg.bg.agh.edu.pl/skrypty/0005/main.html`).

[4] *Google Tests Face Detection Technologies*. `http://www.everyjoe.com/articles/google-tests-face-detection-technologies-46/`.

[5] R. Ludwiczuk. Algorytm canny'ego detekcji krawędzi w procesie segmentacji obrazów medycznych. 2007. `http://kis.pwszchelm.pl/publikacje/II/Ludwiczuk.pdf`.

[6] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. 1971. `http://www.ai.sri.com/pubs/files/tn036-duda71.pdf`.

[7] R. Takei. A new grey-scale template image matching algorithm using the cross-sectional histogram correlation method. 2003. `http://www.math.ucla.edu/~rrtakei/prevRsrch/workreport03.pdf`.

[8] R. E. Van Dyck, J. F. Doherty, and Y. Wang. Moving object tracking in video. 2002. `http://w3.antd.nist.gov/pubs/aipr00.pdf`.

[9] Thanh Duc Ngo, Duy-Dinh Le, Shin ichi Satoh, and Duc Anh Duong. Robust face track finding in video using tracked points. 2008. `http://satoh-lab.ex.nii.ac.jp/users/ndthanh/pub/Ngo-RobustFaceTrackFindingUnsingTrackedPoints.pdf`.

[10] S. Osowski. *Sieci neuronowe w ujęciu algorytmicznym*. Wydawnictwa Naukowo-Techniczne, Warszawa, 1996.

[11] R. Tadeusiewicz. *Sieci neuronowe*. Akademicka Oficyna Wydawnicza, Warszawa, 1993.

[12] J. Żurada, M. Barski, and W. Jędruch. *Sztuczne sieci neuronowe. Podstawy teorii i zastosowania.* Wydawnictwo Naukowe PWN, Warszawa, 1996.

[13] I. Kemelmacher and R. Basri. Molding face shapes by example. *European Conf. on Computer Vision (ECCV-06), LNCS 3951*, I:277–288, 2006.

[14] Singular Inversions. *Facegen Modeler*. `http://www.facegen.com/modeller.htm`.

[15] Abalonellc. *FaceShop*. `http://www.abalonellc.com/`.

[16] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. 1999. `http://www.mpi-inf.mpg.de/~blanz/html/data/morphmod2.pdf`.

[17] AT&T. *The Database of Faces*. `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`.

[18] J. E Viallet and O. Bernier. Face detection for video summaries. 2002. `http://perso.rd.francetelecom.fr/bernier/publications/vialletcivr2002.pdf`.

[19] M. Chau and M. Betke. Real time eye tracking and blink detection with usb cameras. 2005. `http://www.cs.bu.edu/techreports/pdf/2005-012-blink-detection.pdf`.

[20] Blender Foundation. *Blender3D*. `http://www.blender.org/`.

[21] B. Fleming and D. Dobbs. *Animacja cyfrowych twarzy*. Helion, Gliwice, 1999.

[22] B. Fleming and R. H. Schrand. *Tworzenie cyfrowych postaci*. Helion, Gliwice, 2002.

[23] AForge.net framework, `http://www.aforgenet.com/framework/`. *AForge*.

[24] *Open Computer Vision Library*. `http://opencv.willowgarage.com/wiki/`.

[25] SGI. *OpenGL*. `http://www.opengl.org/`.

[26] Vijeth D. *NeuronDotNet*. `http://neurondotnet.freehostia.com`.