ci2.3.429Filtracja krawędzi o małym gradiencie jasnościsubsection.2.3.4

Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki
kierunek: informatyka

Mateusz Trzeciak

Określenie wieku twarzy na podstawie tekstury

praca dyplomowa magisterska

promotor: dr hab. inż. Karolina Nurzyńska

Gliwice, październik 2019

załącznik nr 2 do zarz. nr 97/08/09

Oświadczenie

Wyrażam zgodę / Nie wyrażam zgody* na udostępnienie mojej pracy dyplomowej / rozprawy doktorskiej*.

Gliwice, dnia 25 października 2019

.....
(podpis)

.....
(poświadczenie wiarygodności
podpisu przez Dziekanat)

* podkreślić właściwe

Oświadczenie promotora

Oświadczam, że praca „Określenie wieku twarzy na podstawie tekstury" spełnia wymagania formalne pracy dyplomowej magisterskiej.

Gliwice, dnia 25 października 2019

.....

(podpis promotora)

Spis treści

1 Wstęp **1**

2 Metoda bazowa - wrinkle feature **3**

2.1 Metoda wykrywania twarzy 5

2.1.1 Konwersja do skali szarości oraz przestrzenie barw . . . 10

2.1.2 Algorytm Haar Cascade 13

2.2 Wyznaczanie stref 20

2.3 Wykrywanie zmarszczek - detektor Canny 25

2.3.1 Redukcja szumów na obrazie filtrem Gaussa 27

2.3.2 Szukanie gradientów jasności 28

2.3.3 Zastosowanie techniki zmniejszania grubości krawędzi . 28

2.3.4 Filtracja krawędzi o małym gradiencie jasności 29

2.3.5 Filtracja poprzez histerezę 29

2.4 Wyliczanie wrinkle feature 30

2.5 Algorytm trenowania 31

2.6 Grupowanie danych - Fuzzy C-Means oraz wyznaczanie wieku 32

3 Modyfikacje metody bazowej **37**

3.1 Odjęcie wybranej strefy 37

3.1.1 Zmiana algorytmu względem metody bazowej 38

3.2 Zastosowanie metody HOG 38

3.2.1 HOG - Zastosowanie w projekcie 44

3.3 Metoda HOG oraz algorytm KNN 45

3.3.1 Zastosowanie w projekcie 45

3.3.2 Grupowanie KNN 46

4 Badania **51**
4.1 Środowisko badań 51
4.2 Rozmiar obrazu 52
4.3 Wykrywanie zmarszczek przez detektor Canny oraz Histogram of Oriented Gradients 55
4.4 Statystyki z działania programu 58
4.5 Skuteczność poszczególnych metod szacowania wieku 59
4.5.1 Metoda bazowa 60
4.5.2 Odjęcie strefy 2 61
4.5.3 Metoda HOG 61
4.5.4 Metoda HOG + KNN 61
4.6 Podsumowanie 61
4.7 Wnioski 61

5 Podsumowanie **63**

Rozdział 1

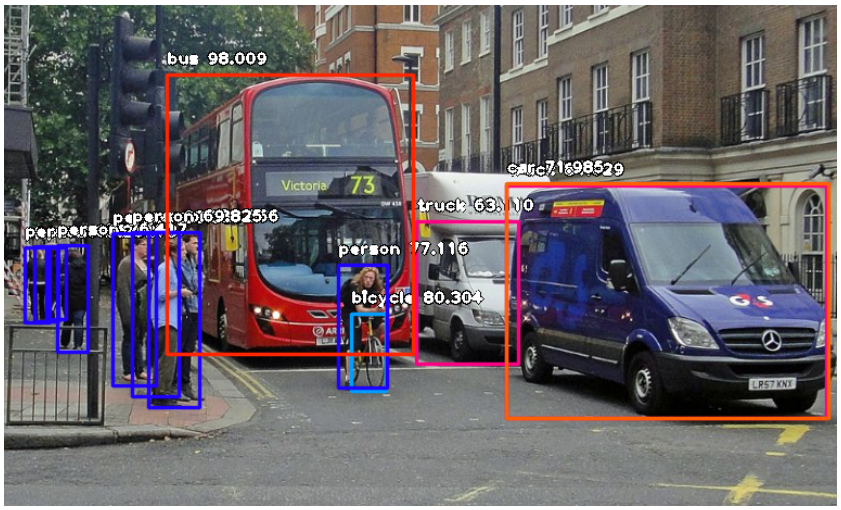
Wstęp

Wiek jest cechą, którą niełatwo człowiekowi odczytać z czyjejś twarzy. Dla komputera rozpoznawanie wieku jest trudniejsze niż dla człowieka. Dlatego do wyznaczania wieku z pomocą programu komputerowego należy podchodzić z dystansem. Mimo trudności programiści i naukowcy udoskonalają algorytmy, tak aby ocena wieku danej osoby była coraz dokładniejsza.

Istnieje wiele sposobów wyznaczania wieku. Większość metod skupia się na analizie tekstury twarzy. Idąc dalej - z obrazu danej osoby lub jego części, np tułowia, musi zostać wykryta twarz. Wykrycie twarzy na teksturze jest możliwe dzięki algorytmom rozpoznawaniu obrazu. Rozpoznawanie obrazu jest stosowane w wizji komputerowej i polega na wyodrębnieniu z obrazu jakichś szczegółów. Mogą to być osoby, pojazdy, przedmioty itp. (Rys. 1.1)

Można znaleźć wiele witryn internetowych, które udostępniają interfejsy programistyczne umożliwiające zaimplementowanie rozpoznawania wieku z obrazu. Istnieją algorytmy przetwarzania obrazu, które oprócz wieku wyznaczają z pewnym prawdopodobieństwem płeć danej osoby. Oprócz płci mogą one także wyznaczyć mine oraz czy dana osoba nosi okulary.

Z weryfikacją wieku danej osoby można się spotkać przed wejściem do niektórych miejsc, tj. klub nocny. Większość osób musi okazać ważny dowód osobisty, co generuje duże kolejki do wejścia. Aplikacje analizujące wiek na podstawie obrazu twarzy z kamery przed wejściem do takich miejsc znacząco usprawniłyby weryfikację wieku. Rozpoznawanie wieku może być wykorzy-



Rysunek 1.1: Przykład rozpoznawania obiektów na zdjęciu ulicy. [22]

stywane przy analizie średniego wieku ludzi w jakimś miejscu np. podczas demonstracji.

Wiele gier posiada treści nieodpowiednie dla młodszych użytkowników. Możliwe jest stosowanie technologii wykrywania wieku użytkownika przed udostępnieniem mu treści, która wymaga odpowiedniego wieku.

Można znaleźć o wiele więcej potencjalnych zastosowań przetwarzania obrazu oraz rozpoznawania wieku na podstawie tekstury (obrazu) twarzy. Z biegiem lat z pewnością będzie można zauważyć dalszy rozwój tej dziedziny, która opiera się w głównej mierze na sztucznej inteligencji [28]. TODO cel i zakres pracy

Rozdział 2

Metoda bazowa - wrinkle feature

Istnieje wiele metod wyznaczania wieku z obrazu twarzy. W literaturze spotkano rozwiązania, w których wyznaczany jest konkretny wiek osoby przez algorytm lub przedział wiekowy. Jedna z pierwszych metod szacowania wieku opierała się na wyznaczaniu proporcji twarzy, a następnie na detekcji i interpretacji zmarszczek. Była ona w stanie ze stu procentową poprawnością wyznaczyć czy dana osoba jest osobą dorosłą lub dzieckiem [31].

W kolejnych latach algorytmy i techniki szacowania wieku były udoskonalane. Badano wpływ starzenia się osób na wygląd skóry. Oprócz naturalnych zmian skóry pod wpływem starzenia się skóry należało uwzględnić także inne czynniki. Takimi czynnikami są min. płeć, poziom stresu, ekspozycja na działanie środowiska zewnętrznego. Powyższe metody zastosowano w pracy „Toward automatic simulation of aging effects on face images" autorstwa A. Lanitis, Ch. J. Taylor oraz T. F. Cootes [1]. Należy dodać, że w powyższej pracy stosowano trenowanie zbioru zdjęć. Trenowanie polega na wykryciu relacji pewnych cech twarzy do wieku osób.

W kolejnych latach pojawiło się podejście porównywania cech twarzy tej samej osoby w różnym wieku. Różnice w powyższych cechach posłużyły do zbudowania statystyki zmian cech twarzy wraz ze starzeniem się. Powyższe podejście zostało zaprezentowane w pracy „Face verification across age

progression" autorstwa N. Ramanathan oraz R. Chellappa [20].

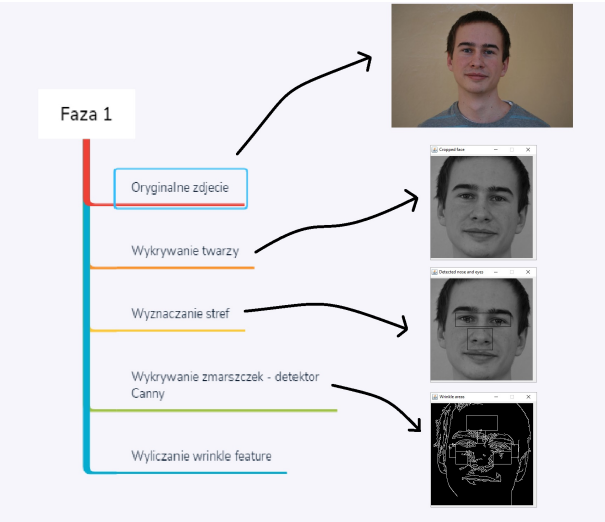
Rozwinięciem tego pomysłu była praca „Automatic age estimation based on facial aging patterns" autorstwa X. Geng, Z. Zhou i K. Smith-Miles [30]. W tej pracy porównywano sekwencje wielu zdjęć twarzy jednej osoby. Zdjęcia przestawiały twarz w różnym wieku. Powyższe badania pozwoliły na zbudowanie wzorca starzenia się twarzy.

Praca „A new algorithm for age recognition from facial images" autorstwa M.M. Dehshibi oraz A. Bastanfard [8] przy szacowaniu wieku analizuje proporcje twarzy oraz ilość zmarszczek.

Praca „Age Estimation from Face Images: Challenging Problem for Audience Measurement Systems" autorstwa Vladimira Khryashcheva, Alexandra Ganina, Olgi Stepanovej oraz Antona Lebedeva podsumowała techniki szacowania wieku [14]. Z podsumowania wynikło, że najczęściej stosuje się do wyodrębniania cech z twarzy BIF, czyli biologically inspired features. Powyższa metoda została zaprezentowana w książce „Human Age Estimation Using Bio-inspired Features" autorstwa Guodong Guo i in. Mniej popularne metody analizujące cechy twarzy to filtry Gabora oraz LBP- local binary patterns.

Metoda bazowa została opisana w artykule „Age Estimation from Face Image using Wrinkle Features" [25]. Wykrywanie wieku dzieli się na kilka faz. Na początku należy wykryć twarz. Zastosowany algorytm wykrywania został opisany w sekcji 2.1 Następnie należy wyznaczyć strefy zmarszczkowe na twarzy. W artykule [25] udowodniono, że istnieje kilka konkretnych stref, w których następuje znacząca zmiana ilości zmarszczek wraz z wiekiem. Powyższe strefy zostały wymienione w sekcji 2.2. Sekcja **??** przedstawia technikę wykrywania zmarszczek znajdujących się w strefach. Wykryte zmarszczki pozwalają na obliczenie wrinkle feature dla danej twarzy, zgodnie z opisem w sekcji 2.4. W tym miejscu kończy się faza wyznaczania wrinkle feature dla danej osoby (Rysunek 2.1). Kolejna faza jest potrzebna do znalezienia relacji pomiędzy wrinkle feature a wiekiem. Do tego celu należy zastosować algorytm trenujący, który został opisany w sekcji 2.5. Wynikiem algorytmu trenującego jest zbiór danych, który należy pogrupować, tak jak to opisano w sekcji 2.6. Ostatnią fazą algorytmu jest wykrywanie wieku na podstawie

wyników działania FCM - sekcja **??** (Rysunek 2.2).



Rysunek 2.1: Faza 1 algorytmu

Rysunek 2.2: Faza 2 algorytmu

2.1 Metoda wykrywania twarzy

W literaturze można odnaleźć wiele metod wykrywania twarzy. Istnieje kilka podejść aby skutecznie wykrywać twarz na danym obrazie [26]:

- metoda oparta na nauce
- metoda niezmienności cech
- metoda dopasowania szablonu twarzy
- metoda bazująca na wyglądzie

Metoda oparta na nauce kieruje się wiedzą na temat wyglądu twarzy, a bardziej precyzyjnie chodzi o charakterystyczne cechy, dzięki którym na zdjęciu można wyodrębnić obszar twarzy. Mowa tutaj o cechach takich jak kształt twarzy, kolor, miejsca o różnej jasności czy specyficzne krawędzie tworzone np przez usta.

W metodzie niezmienności cech wyszukuje się takie strukturalne cechy twarzy, które są widoczne w każdych warunkach oświetleniowych. Ponadto te cechy są widoczne bez względu na punkt widzenia, nawet jeśli twarz jest widoczna z profilu czy przechylona pod kątem.

Z kolei metoda dopasowania szablonu twarzy wykorzystuje kilka standardowych wzorów opisujących twarz. Na wejściu algorytmu obraz jest porównywany z tymi wzorami, a na wyjściu dostajemy informację, w jakim stopniu obraz jest dopasowany do szablonu twarzy.

Ideą metody bazującej na wyglądzie jest trenowanie dużego zbioru obrazów twarzy, tak aby wychwycić zmienność cech twarzy. Tak wytrenowany model jest później wykorzystywany do wykrywania twarzy.

Ponadto w procesie ekstrakcji twarzy z obrazu istnieje wiele problemów [26].

Jednym z problemów jest nieodpowiednia poza. Wiąże się to z różnymi ustawieniami twarzy wobec aparatu fotograficznego lub kamery. Twarz może być nachylona, przechylona lub odchylona. Inaczej mówiąc może mieć różne położenie w trzech wymiarach, a niektóre części twarzy lub jej cechy mogą zostać przysłonięte. Im mniej cech widocznych na twarzy, tym mniej danych, które algorytm może z niej wyodrębnić, a im mniej danych, którymi algorytm operuje, tym mniejsze prawdopodobieństwo prawidłowego wykrycia twarzy.

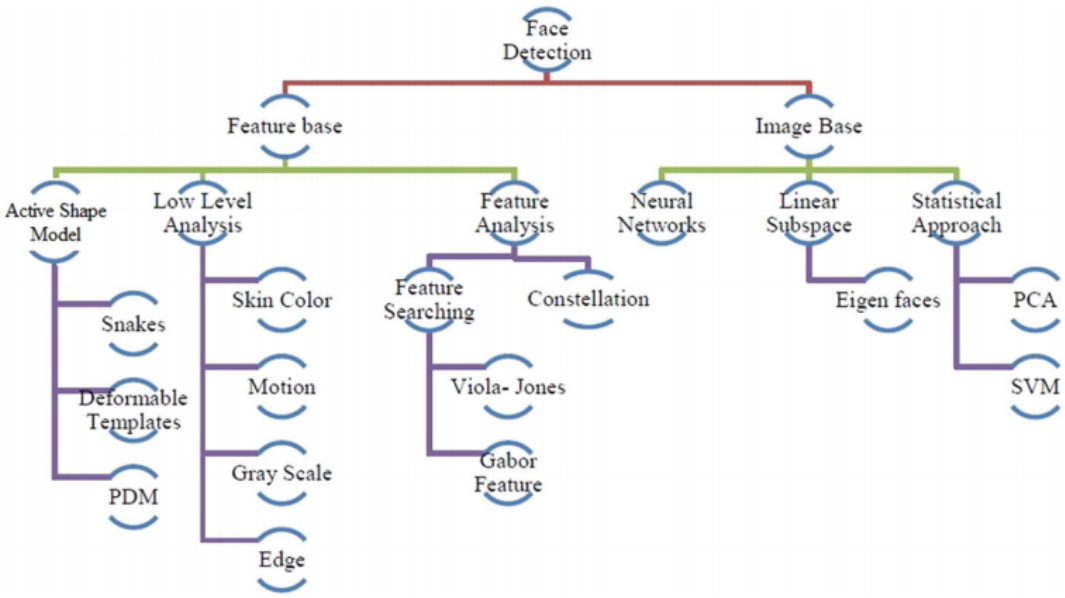
Niektóre twarze mogą zawierać pewne wyróżniające je cechy takie jak broda, blizny czy okulary. Różnorodność tych cech także wpływa na efektywność wykrywania twarzy.

Rysunek 2.3: Przykład twarzy oświetlonej silnym (twarz po lewej) oraz miękkim światłem. [10]

Ilość zmarszczeń na twarzy jest zmienna, w zależności od wyrazu mimicznego. Przy różnych minach zmienia się kształt ust, a czasem pojawiają się ostre krawędzie wynikające z pracy mięśni twarzowych. Widoczne mogą być różne pofałdowania skóry.

Zdarza się, że twarz zostaje częściowo przysłonięta przez jakiś inny obiekt. Na przykład na zdjęciu, które obejmuje grupę wielu osób, część danej twarzy może być przysłonięta przez inną twarz. Takie przysłonięcie przez inny obiekt wiąże się z utratą informacji o części twarzy, co zmniejsza prawdopodobieństwo prawidłowego jej wykrycia.

Kolejnym istotnym elementem jest oświetlenie twarzy. Gdy twarz oświetlona jest tzw. silnym światłem, występują na niej cienie i światła określane jako ostre (Sekcja 2.3). W tym przypadku ryzyko utraty szczegółów oświetlanej twarzy jest większe. Kiedy twarz jest skierowana na wprost słońca, z dużym prawdopodobieństwem można stwierdzić, że zostanie oświetlona silnym światłem. Z kolei miękkie światło jest generowane na przykład przez zachmurzone niebo. Istotne jest także źródło światła, które może być punktowe lub rozproszone. Przy punktowym źródle światła cała twarz jest okryta jednolitym cieniem, którego intensywność zależy od rodzaju światła. Natomiast przy świetle rozproszonym intensywność cieni jest mniejsza.



Rysunek 2.4: Różne techniki wykrywania twarzy. [3]

Na Rys 2.4 przedstawione są różne techniki wykrywania twarzy.

Jak widać metod wykrywania twarzy jest sporo. Omówienie każdej z nich zajęłoby dużo czasu. Poniżej zostaną przytoczone dwie metody wykrywania twarzy. Dodatkowo zostanie omówiona metoda, która posłużyła do wykrywania twarzy w niniejszej pracy.

W pracy „An efficient algorithm for human face detection and facial feature extraction under different conditions" [15] przedstawiono opisaną w skrócie poniżej technikę wykrywania twarzy. W pierwszym etapie procesu obszary, gdzie może znajdować się ludzkie oko, są wykrywane przez przeprowadzenie testów na zacienionych rejonach obrazu. Pary takich obszarów wyodrębnia się na podstawie algorytmu genetycznego, aby następnie wyznaczyć możliwy obszar twarzy. Dla każdego obszaru mierzy się wartość dopasowania na podstawie jego projekcji na wektory własne, tzw. eigenfaces. Aby wiarygodność wykrywania była wyższa, każdy możliwy obszar twarzy normalizuje się pod kątem oświetlenia. Proces ten powtarza się pewną ilość razy, a następnie do dalszej weryfikacji są wybierane możliwe obszary twarzy o wysokiej wartości dopasowania. Na tym etapie mierzy się symetrię twarzy oraz

sprawdza się, czy na każdym wybranym obszarze istnieją rysy twarzy. Rysy określa się przez ewaluację rzeźby topograficznej - wystających i wklęsłych elementów różnych regionów obszaru twarzy, poddanego uprzednio normalizacji. Algorytm jest w stanie wykryć także obszar twarzy, gdy głowa jest przechylona

W roku 1997 w pracy pt. „Vision for man-machine interaction" opisano metodę wykrywania twarzy bazującą na wykrywaniu cechy jaką jest kolor skóry [6]. Kolor skóry jest najbardziej widoczną cechą twarzy zarówno dla człowieka jak i dla maszyny. Ponadto kolor jest przetwarzany znacznie szybciej od innych cech. Przy dobrych warunkach oświetleniowych ustawienie twarzy nie ma wpływu na skuteczność wykrywalności twarzy opisywaną metodą. Każda metoda wykrywania twarzy posiada wady. Jedną z tych wad jest problem wykrywalności twarzy przy nierównomiernym oświetleniu. Problem pojawia się także, gdy na obrazie widoczny jest obszar skóry z poza twarzy np. z rąk. Warto zaznaczyć, że kolor twarzy na obrazie jest zależny od względnego kierunku oświetlenia. Obszar twarzy w omawianym algorytmie jest wykrywany poprzez normalizacje histogramu kolorów. Normalizacja jest potrzebna do redukcji wpływu luminancji na kolor.

Algorytm Haar Cascade jest najpopularniejszym algorytmem do wykrywania twarzy w bibliotece OpenCV. Właśnie ta biblioteka była jednym z niezbędnym elementów programu przy wykrywaniu wieku z tekstury twarzy. W związku z powyższym w wykrywaniu twarzy zastosowano algorytm Haar Cascade. Omawiany algorytm został zaprezentowany w książce „Rapid Object Detection using a Boosted Cascade of Simple Features" w 2001 i składa się z trzech faz [24]. W pierwszej obraz wejściowy przekształcany jest na obraz scałkowany. Następnie wykorzystywany jest algorytm do boostingu, który zmniejsza ilość klasyfikatorów tylko do tych najbardziej istotnych. W ostatniej fazie klasyfikatory łączone są w kaskady w celu przyspieszenia procesu wykrywania twarzy. Znaczna większość metod wykrywania obiektów na obrazie (w tym twarzy) wymaga wstępnego przekształcenia obrazu do skali szarości.

2.1.1 Konwersja do skali szarości oraz przestrzenie barw

Każdy piksel w trybie kolorowym ma określoną reprezentację barwy z określonego modelu. Najczęściej są to 3 lub 4 wartości [13]. Pierwszą przestrzenią barw była CIEXYZ. Została ona stworzona w 1931 przez Międzynarodowa Komisja ds. Oświetlenia (International Commission on Illumination). Przestrzeń barw CIEXYZ została specjalnie stworzona, by odtworzyć sposób postrzegania barw przez ludzkie oko. Barwa jest opisywana w trzech współrzędnych trójchromatycznych X,Y,Z. Powyższe współrzędne są zależne od składowych - sprawności wizualnych czopków. Czopki to światłoczułe receptory siatkówki ludzkiego oka [13]. Współrzędne X,Y,Z wyliczane są na podstawie trzech podstawowych barw R (czerwonej), G (zielonej) i B (niebieskiej). Współrzędne XYZ są często reprezentowane przez luminancję Y oraz współrzędne x, y chromatyczności. Wyliczanie współrzędnych x, y przedstawiono na wzorach 2.1 oraz 2.2.

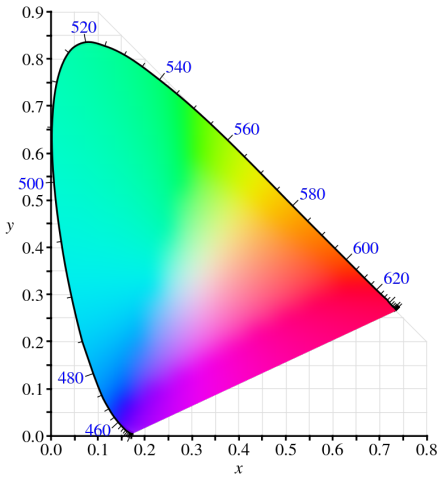
$$x = \frac{X}{X + Y + Z} \tag{2.1}$$

$$y = \frac{Y}{X + Y + Z} \tag{2.2}$$

Na Rysunku 2.5 przedstawiono diagram chromatyczności reprezentujący przestrzeń barw CIEXYZ.

Kolory mogą byc także odwzorowane przez przestrzeń barw CMYK. Skrót CMYK oznacza odpowiednio:

- Cyan - odcień niebieskiego
- Magenta - kolor karmazynowy
- Yellow - kolor żółty
- K - key colour - kolor czarny



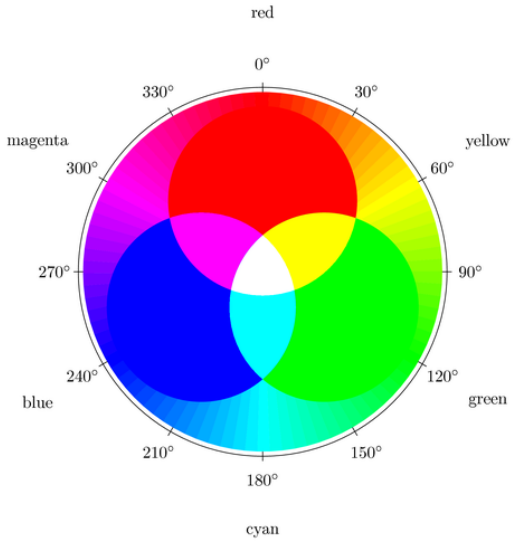
Rysunek 2.5: Przykład rozpoznawania obiektów na zdjęciu ulicy. [5]

Barwa wynikowa powstaje poprzez mieszanie trzech kolorów - niebieskiego, karmazynowego oraz żółtego. Mieszanie zachodzi według zasady syntezy subtraktywnej [13]. Synteza subtraktywna polega na mieszaniu kolorów przez odejmowanie promieniowań widzialnych różnych długości. Przykładem syntezy subtraktywnej jest np. mieszanie farb o różnych kolorach.

Najczęściej barwy są reprezentowane przez przestrzeń barw RGB [13]. Przestrzeń kolorów RGB składa się z trzech kanałów [12]:

- R - czerwonego (z angielskiego Red)
- G - zielonego (z angielskiego Green)
- B - niebieskiego (z angielskiego Blue)

Barwy mieszane są poprzez syntezę addytywną . W przeciwieństwie do syntezy subtraktywnej barwa wynikowa powstaje poprzez sumowanie wiązek światła widzialnego o różnych długościach [13]. Każdy piksel opisany za pomocą przestrzenie barw RGB ma trzy 8-bitowe wartości reprezentujący każdy kanał. Spotykane są 12- lub 16-bitowe reprezentacje kanałów, jednak 8-bitowa jest najpopularniejsza. Dla 8-bitowych kanałów wartość „0” danego kanału oznacza brak jasności, natomiast „255” maksymalną jasność. Poprzez



Rysunek 2.6: Mieszanie kanałów RGB [19].

Rysunek 2.7: Kolor R=153 G=217 B=234.

mieszanie jasności tych trzech kanałów można uzyskać szerokie spektrum barw (Rysunek 2.6).

Przykładowo kolor o reprezentacji R=153 G=217 B=234 przedstawiono na Rysunku 2.7

Kolor (Rysunek 2.7) może być też reprezentowany w kodzie szesnastkowym #99D9EA. Każda wartość heksadecymalna odpowiada kolejno kanałowi R, G, B.

Obraz może też być przedstawiony stosując odcienie jednej barwy. Taka obraz nazywa się obrazem monochromatycznym. Najczęściej stosowaną barwą w takich obrazach jest szarość [13].

Istnieją 3 metody konwersji obrazu z przestrzeni RGB na monochromatyczny [19].

- największej jasności

- średnia
- luminancji

Metoda największej jasności konwertuje na skalę szarości wg wzoru2.3.

$$\frac{(max(R,G,B)+min(R,G,B))}{2} \tag{2.3}$$

Metoda średnia bazuje na wzorze 2.4, natomiast metodę luminancji ilustruje wzór 2.5.

$$\frac{(R+G+B)}{3} \tag{2.4}$$

$$0.21R + 0.72G + 0.07B \tag{2.5}$$

W niniejszej pracy zastosowano konwersję za pomocą metody średniej.

2.1.2 Algorytm Haar Cascade

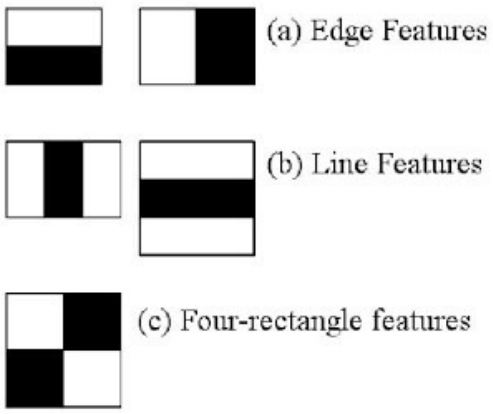
Haar Cascade jest algorytmem służącym do wykrywania obiektów na obrazach. Został stworzony przez Paula Viola oraz Michaela Jonesa w 2001 roku [24].

Opiera się na zbudowaniu kaskadowej funkcji za pomocą trenowania wielu zdjęć. Zdjęcia są dzielone na dwie kategorie - pozytywne oraz negatywne. Na zdjęciach klasyfikowanych jako pozytywne istnieje obiekt, który ma zostać wykryty, natomiast na zdjęciach negatywnych nie ma tego obiektu.

Ekstrakcja cech w algorytmie Violi i Jonesa jest realizowana przez filtry Haara. Są to prostokątne okienka nakładane na obraz, które analizują jasność pikseli (Rysunek 2.8).

Przed zastosowaniem filtru Haara obraz musi zostać przekształcony do skali szarości W niniejszej pracy należało przekształcić każdy obraz z trybu kolorowego na monochromatyczny, co opisano w sekcji 2.1.1.

Każde okienko zawiera białe oraz czarne prostokąty. Wyznaczana jest suma jasności pikseli w obu rodzajach prostokątów, a następnie dla każdego



Rysunek 2.8: Filtr Haara a) krawędziowy b) liniowy c) szachownica [2]

okna obliczana jest różnica pomiędzy białymi a czarnymi. Opisywany algorytm ma zastosowanie w wykrywaniu krawędzi. Na granicy krawędzi istnieje różnica w jasności pikseli (Rysunek 2.9).

W celu poprawy efektywności sumowania pikseli stosowane są rozwiązanie zwane w języku angielskim Summed-area table [23].Summed-area table jest również nazywana w literaturze Integral Image, czyli obrazem scałkowanym. Ideą obrazu scałkowanego jest, aby każdy obraz został przekształcony w tabelę, w której każdy element x,y tej tabeli odpowiada sumie jasności wszystkich pikseli według wzoru 2.6.

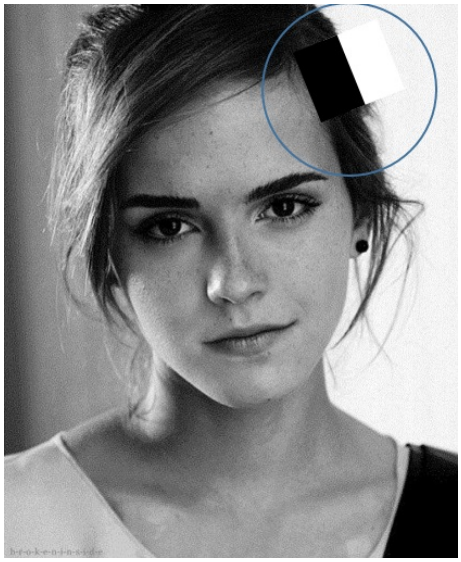
$$I(x,y) = \sum_{x' \leq x \cap y' \leq y} i(x',y') \tag{2.6}$$

gdzie I(x,y) jest wartością na pozycji x,y w tabeli(tabela obrazu scałkowanego), natomiast i(x,y) oznacza jasność piksela o współrzędnych x,y na obrazie.

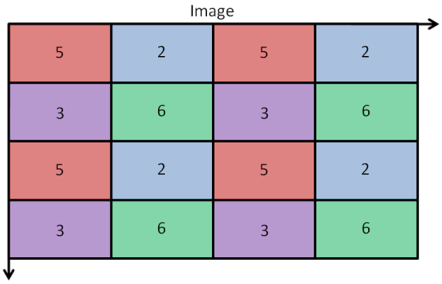
Na Rysunku 2.10 przedstawiona jest tabela prezentująca jasność pikseli przed zastosowaniem całkowania obrazu.

Po całkowaniu otrzymujemy tabelę podobną do przedstawionej na Rysunku 2.11.

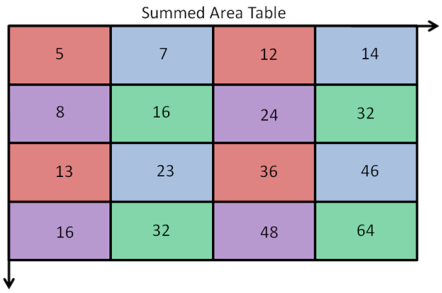
Sumowanie przykładowego okna (Rysunek 2.12) wymaga czterech opera-



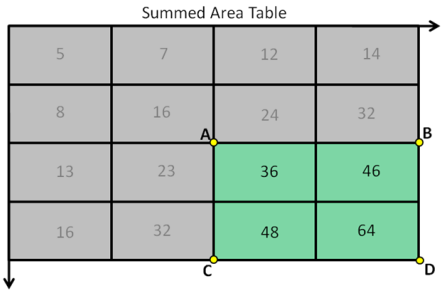
Rysunek 2.9: Filtr Haara nałożony na krawędź twarzy [2]



Rysunek 2.10: Tabela jasności poszczególnych pikseli przed zastosowaniem całkowania [4]



Rysunek 2.11: Filtr Haara nałożony na krawędź twarzy [4]



Rysunek 2.12: Sumowanie okna [4]

cji (Wzór **??**).

$$\sum_{x_0 \leq x \leq x_1 \cap y \leq y \leq y_1} i(x,y) = I(D) + I(A) - I(B) - I(C) \qquad (2.7)$$

gdzie lewa część równania oznacza sumę jasności pikseli zaznaczonego okna tj. na Rysunku 2.12, I(A) - wartość scałkowanego obrazu przy punkcie A (analogicznie I(B), I(C), I(D)) - (Rysunek 2.12).

W związku z powyższym obliczenie wartości dla krawędziowego filtru Haara wymaga obliczenia różnicy dwóch sum, zatem konieczne jest wykonanie ośmiu operacji. Reprezentacja obrazu za pomocą obrazu scałkowanego znacznie zwiększa efektywność obliczania wartości w filtrze Haara.

Liczba cech wykrywanych w danym zdjęciu za pomocą filtru Haara jest znacznie większa od liczby pikseli na obrazie [24]. Dla obrazu o wymiarach 384x288 pikseli liczba cech wynosi ponad 180000. Autorzy algorytmu stwierdzili, że dla zwiększenia jego szybkości należy wybrać małą grupę cech, które razem mogą stworzyć jeden efektywny klasyfikator obiektu. W celu wyodrębnienia tych istotnych cech zastosowano algorytm Adaboost, który został opisany poniżej.

Zbiór n obrazów do trenowania można oznaczyć tak jak we wzorze 2.8:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \qquad (2.8)$$

oznacza, że obraz jest odpowiednio negatywny lub pozytywny. Następnym krokiem jest inicjalizacja wag (Wzór 2.9).

$$w_{1,i} = \frac{1}{2m}, \frac{1}{2l} \tag{2.9}$$

odpowiednio dla $y_i = 0, 1$, gdzie m, l oznacza odpowiednio liczbę negatywnych oraz pozytywnych zdjęć. Następnie Dla t = 1, ..., T:

1. Normalizowane są wagi (Wzór 2.10):

$$w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \tag{2.10}$$

w_t jest rozkładem prawdopodobieństwa

2. Dla każdej cechy j, trenowany jest klasyfikator h_j, który używa tylko jednej cechy wyliczonej z filtru Haara. Błąd jest liczony następująco (Wzór 2.11):

$$w_t, \epsilon_j = \sum_i w_i h_j(x_i) - y_i \tag{2.11}$$

3. Wybierany jest klasyfikator h_t z najmniejszym błędem ϵ_t.

4. Następuje aktualizacja wag (Wzór 2.12):

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \tag{2.12}$$

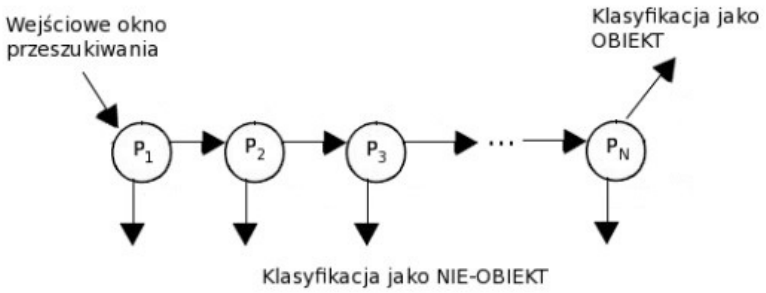
, gdzie $e_i = 0$. Jeśli x_i jest sklasyfikowane prawidłowo, wtedy $e_i = 1$, w innym wypadku $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

Silny klasyfikator $h(x)$ jest opisany równaniem:

$$h(x) = \begin{cases} 1 & \text{gdy } \sum_{t=1}^{T} \alpha_t h_t(x) >= \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{w przeciwnym wypadku} \end{cases} \tag{2.13}$$

, gdzie $\alpha_t = \lg \frac{1}{\beta_t}$

Algorytm Adaboost zmniejsza ilość cech Haara z ponad stu tysięcy do



Rysunek 2.13: Kaskada klasyfikatorów. [29]

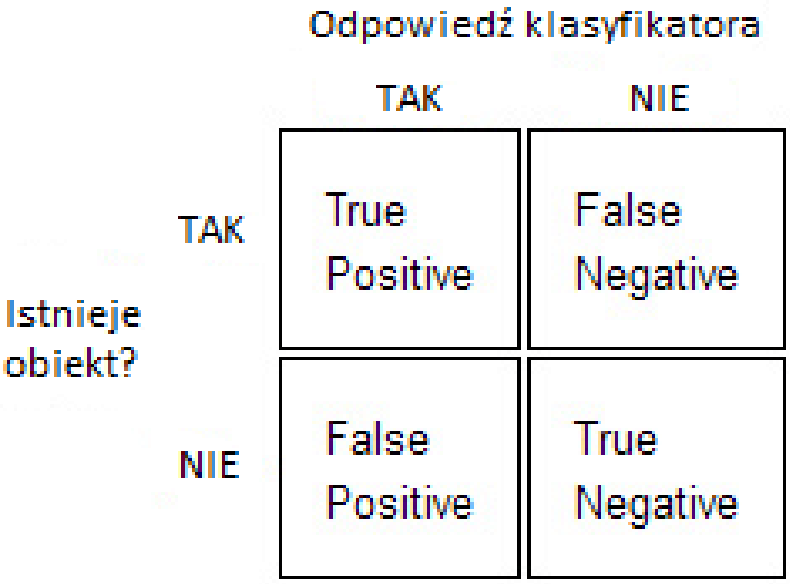
kilkuset - do tych najistotniejszych cech.

Ostatnim etapem jest wytworzenie kaskady klasyfikatorów. Zwiększa ona znacznie szybkość wykrywania pożądanego obiektu na obrazie. Ideą kaskady jest zgrupowanie klasyfikatorów, które powstały w poprzednim procesie - tzw. procesie boostingu. Klasyfikatory są grupowane w okna, które są połączone ze sobą tak jak na rysunku 2.13. Okna są oznaczone jako P1, P2, ..., Pn. Gdy dane okno wykryje obiekt, przechodzi do kolejnego okna w kaskadzie. W przeciwnym wypadku algorytm przerywa działanie i na danym obrazie nie zostaje zidentyfikowany obiekt. Okna są poustawiane tak, aby każde z nich klasyfikowało obiekt z różnym prawdopodobieństwem wykrycia oraz prawdopodobieństwem błędu. Składnikami wyżej wymienionego prawdopodobieństwa jest macierz pomyłek (Rysunek 2.14).

Z macierzy2.14 można odczytać czy klasyfikator poprawnie sklasyfikował dane testowe. W macierzy użyto 4 pojęcia:

- TP (true positive) - poprawna klasyfikacja, jako obiekt
- TN (true negative) - poprawna klasyfikacja, jako nie - obiekt
- FP (false positive) - błędna klasyfikacja jako obiekt
- FN (false negative) - błędna klasyfikacja jako nie - obiekt

Prawdopodobieństwo błędu wyliczane jest ze współczynnika FPR (false positive rate) (Wzór 2.14).



Rysunek 2.14: Macierz pomyłek.

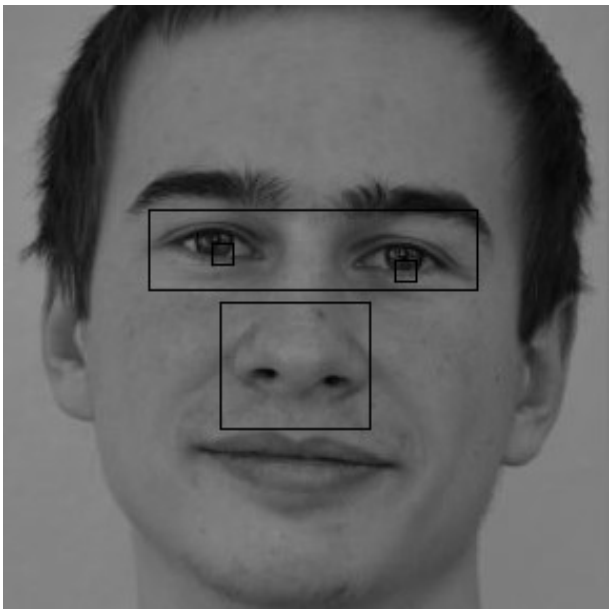
$$FPR = \frac{FP}{FP + TN} \tag{2.14}$$

Natomiast prawdopodobieństwo wykrycia obiektu wyliczane jest ze współczynnika TPR (true positive ratio) (Wzór 2.15)

$$TPR = \frac{TP}{TP + FN} \tag{2.15}$$

Pierwsze okna posiadają klasyfikatory o słabszym TPR oraz FPR niż kolejne okna. Oznacza to, że prawdopodobieństwo TPR w oknie P_{x-1} jest mniejsze od tego w P_x. Natomiast prawdopodobieństwo FPR w oknie P_{x-1} jest większe od tego w P_x. Ostatnie okna mają największy współczynnik TPR oraz najmniejszy FRP ze wszystkich. Takie ustawienie okien ma na celu wstępne przepuszczenie przez okna obrazów, które z dużym prawdopodobieństwem zawierają szukany obiekt. Natomiast ostatnie okna w kaskadzie analizują niewielką część obrazu wejściowego.

Biblioteka OpenCV zawiera wytrenowane klasyfikatory, które zostały użyte



Rysunek 2.15: Wykryty nos oraz oczy.

w tej pracy magisterskiej. Wykorzystano je do wykrycia twarzy, ust oraz oczu. Klasyfikatory mają postać plików xml, które można znaleźć na oryginalnym repozytorium projektu OpenCv.

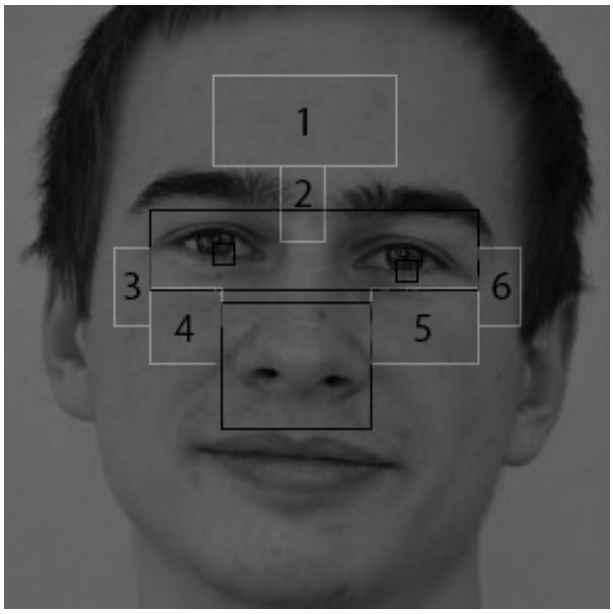
2.2 Wyznaczanie stref

Przed wyznaczeniem stref zmarszczkowych należy zidentyfikować na twarzy oczy oraz nos (Rysunek 2.15).

Należy podkreślić, że algorytm przerywa działanie jeśli nie zostanie wykryta twarz, oczy lub nos.

Gdy zostanie wykryty obszar twarzy, oczu oraz nosa wyznaczone zostaje sześć stref zmarszczkowych (Rysunek 2.16).

Strefy zmarszczkowe są na czole (Strefa „1”), w górnej części nosa (Strefa „2”), górnej części policzków (Strefa „4” i „5”) oraz w okolicach powiek (Strefa „3” i „6”). To właśnie te miejsca zostały uznane przez autorów książki „Age Estimation from Face Image using Wrinkle Features” [25] za najbardziej znaczące w detekcji wieku.



Rysunek 2.16: Strefy zmarszczkowe widoczne w białych prostokątach.

Po detekcji oczu należy zmierzyć odległość pomiędzy środkiem lewego oka (x_l, y_l), a prawego (x_p, y_p) (Wzór 2.16).

$$d = \sqrt{(x_r - x_l)^2 + (y_r - y_l)^2} \tag{2.16}$$

Odległość d służy do wyznaczanie strefy znajdującej się na czole (Rysunek 2.17).

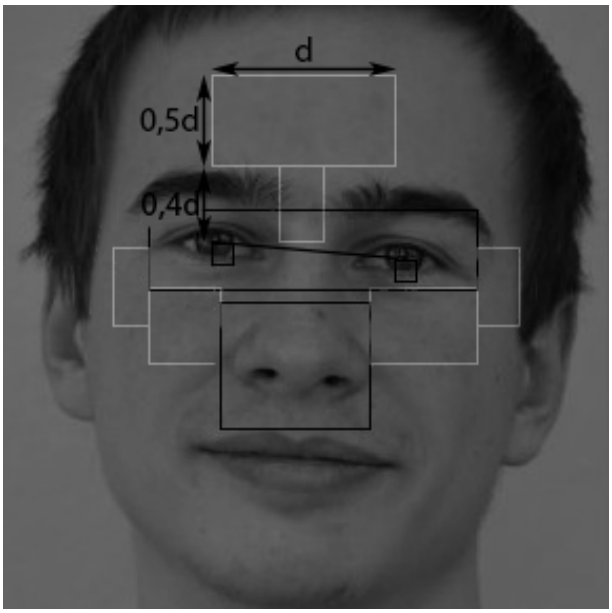
Autorzy [25] algorytmu założyli, że odległość od linii oczu do linii brwi wynosi $0,4 * d$, natomiast wymiary „strefy czoła" wynoszą $d \times 0,5d$.

Na Rysunku 2.18 przedstawione są współrzędne, które są pomocne w wyznaczaniu stref zmarszczkowych.

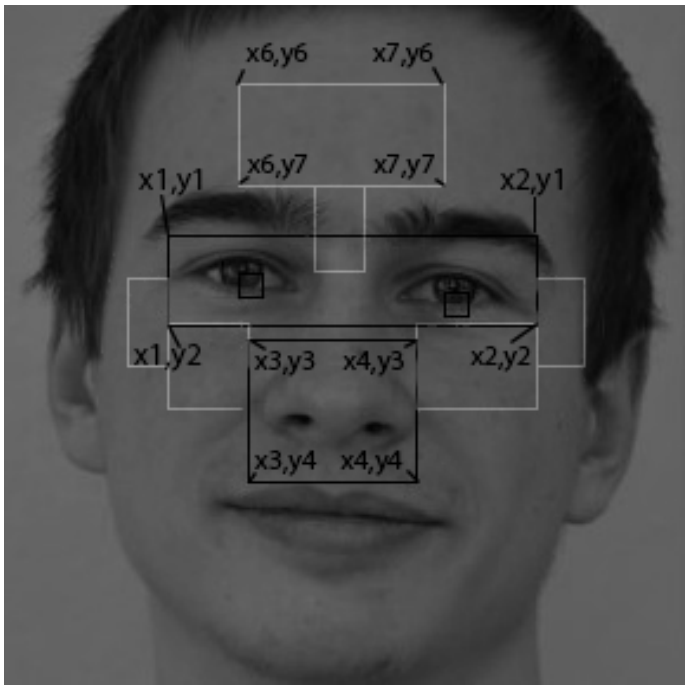
Strefa 2 została wyznaczona dzięki znajomości położenia prawego oka, dystansu między oczami oraz strefy „1". Każda strefa może być wyznaczona przez jeden punkt (Rysunek 2.19) oraz jej wymiary.

Współrzędna x punktu A została wyznaczona ze wzoru 2.17

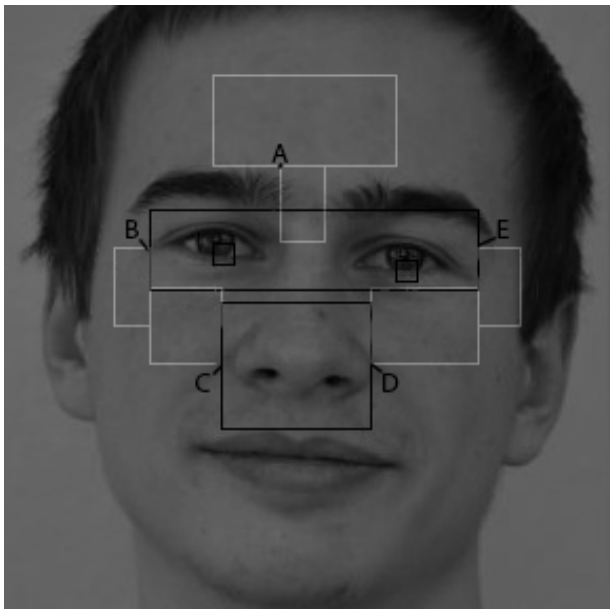
$$x_A = x_6 + 0,375 \times d \tag{2.17}$$



Rysunek 2.17: Wyznaczenie strefy znajdującej się na czole.



Rysunek 2.18: Pomocnicze współrzędne do wyliczania stref zmarszczkowych.



Rysunek 2.19: Punkty wyznaczające położenie stref.

, gdzie x_6 to współrzędna z Rysunku 2.18. Natomiast d jest odległością pomiędzy oczami (Wzór 2.16).

Współrzędna y punktu A - Wzór 2.18:

$$y_A = y_7 \tag{2.18}$$

Współrzędna y_7 analogicznie jak x_6 znajduje się na Rysunku 2.18.

Zakładając, że prawe oko ma współrzędne x_o oraz y_o wysokość strefy 2 wyznaczana jest w poniższy sposób (Wzór 2.19).

$$h_A = y_o - y_7 \tag{2.19}$$

Natomiast szerokość strefy 2 oblicza się następująco: (Wzór 2.20):

$$w_A = 0,25 * d \tag{2.20}$$

Wartość 0,25 ze wzoru 2.20 została dobrana empirycznie.

Współrzędne punktu C (strefa 4) zostają wyznaczone ze Wzoru 2.21 oraz 2.22.

$$x_C = x_3 \tag{2.21}$$

$$y_C = \frac{y_4 - y_3}{2} \tag{2.22}$$

Wysokość oraz szerokość strefy 4 (Wzór 2.23 oraz 2.24)

$$h_C = y_C - y_3 \tag{2.23}$$

$$w_C = x_3 - x_1 \tag{2.24}$$

Strefa 5 zawiera punkt D - Wzór 2.25 oraz 2.26.

$$x_D = x_4 \tag{2.25}$$

$$y_D = y_C \tag{2.26}$$

Dla strefy 5 wyznacza sie wysokość (Wzór 2.27) oraz szerokość na podstawie wzoru (Wzór 2.28).

$$h_D = h_C \tag{2.27}$$

$$w_D = x_2 - x_4 \tag{2.28}$$

Strefa 3 zawiera punkt B wyznaczany ze Wzorów 2.29 i 2.30

$$x_B = x_1 \tag{2.29}$$

$$y_B = \frac{y_2 - y_1}{2} \tag{2.30}$$

Wysokość (Wzór 2.31) oraz szerokość (Wzór 2.32) strefy 3.

$$h_B = y_C - y_B \tag{2.31}$$

$$w_B = w_C * 0,4 \tag{2.32}$$

Na samym końcu zostaje wyznaczona strefa 6. Wyznaczona zostaje współrzędna E - Wzór 2.33 i 2.34.

$$x_E = x_2 \tag{2.33}$$

$$y_E = y_B \tag{2.34}$$

Szerokość oraz wysokość strefy 6.

$$h_E = h_B \tag{2.35}$$

$$w_E = w_D * 0,4 \tag{2.36}$$

Należy zauważyć, że szerokość stref 3 i 6 jest pomnożona przez 0,4 szerokości odpowiednio stref 4 i 5. Wartość „0,4" została dobrana empirycznie.

2.3 Wykrywanie zmarszczek - detektor Canny

Następnym krokiem algorytmu po wyznaczeniu stref zmarszczkowych jest wyodrębnienie zmarszczek. W celu identyfikacji zmarszczek zastosowano detektor Canny. Detekcja krawędzi pozwala na wyodrębnienie wielu użytecz-

nych cech z obrazu, ponadto znacznie zmniejsza ilość informacji do przetworzenia przez algorytm, który wyodrębnia cechy z obrazu. Istnieje wiele detektorów krawędzi, jednak detektor Canny jest jednym z najbardziej dokładnych i niezawodnych. Metoda detekcji została opracowana przez Johna F. Canny w 1986 roku. Oprócz samej implementacji algorytmu jego twórca zaprezentował teorię obliczeniową, która wyjaśnia działanie tej metody. Ponadto Canny zauważył, że wymagania dotyczące implementacji detekcji krawędzi są podobne do siebie w wielu systemach wizyjnych. W związku z powyższym jego algorytm może zostać skutecznie zastosowany w różnych systemach wizyjnych. Poniżej przedstawiono najważniejsze zasady, które są niezbędne do dobrej detekcji krawędzi w opisywanym algorytmie:

- Detekcja krawędzi z niskim prawdopodobieństwem błędu: Oznacza to, że algorytm powinien wykryć jak najwięcej krawędzi, które rzeczywiście istnieją. Natomiast ilość krawędzi wykrytych błędnie powinna być jak najmniejsza.
- Precyzja detekcji: Algorytm powinien precyzyjnie zlokalizować krawędź.
- Brak redundantnych detekcji: Krawędź powinna być zlokalizowana jednokrotnie, a szum na obrazie nie powinien generować dodatkowych krawędzi.

W algorytmie detekcji Canny używa rachunku wariacyjnego. Rachunek wariacyjny to dziedzina analizy matematycznej, która analizuje przestrzenie funkcyjne i znajduje w nich ekstrema funkcjonałów. Funkcjonały natomiast przekształcają przestrzeń wektorową na liczby rzeczywiste. Zatem rachunek wariacyjny ma za zadanie pomóc w znalezieniu charakterystycznej funkcji, dla której funkcjonał przyjmuje wartość ekstremalną. Algorytm detektora jest podzielony na kilka kroków [16]:

- Redukcja szumów na obrazie filtrem Gaussa.
- Szukanie gradientów jasności.
- Zastosowanie techniki zmniejszania grubości krawędzi.

Rysunek 2.20: a) Obraz z nieznaczną ilością szumów b) Obraz ze znaczną ilością szumów. [27]

- Likwidacja krawędzi o małym gradiencie jasności.

- Filtracja poprzez histerezę.

2.3.1 Redukcja szumów na obrazie filtrem Gaussa

Szum na obrazie to piksele o losowym kolorze, jasności oraz umiejscowieniu (współrzędnych na obrazie). Takie piksele są nadmiarowymi informacjami o obrazie oraz stanowią efekt uboczny przetwarzania obrazu przez matrycę cyfrową (Rysunek 2.20).

Efekt szumu jest spotykany również w fotografii analogowej. W fotografii cyfrowej szum wzrasta na skutek zwiększania czułości matrycy lub przez wzrost jej temperatury. Redukcję szumów można uzyskać stosując filtr Gaussa. Splot filtru Gaussa z obrazem daje w wyniku wygładzony obraz ze zmniejszoną ilością szumów. Filtr Gaussa może mieć różne wymiary, natomiast Najczęściej stosowanym jest 5x5. Rozmiar filtra ma wpływ na wydajność wygładzania obrazu, a co za tym idzie na czułość wykrywania szumu [16].

2.3.2 Szukanie gradientów jasności

Istotnym parametrem w wykrywaniu krawędzi jest gradient jasności. Określa on jak bardzo zmienia się jasność danego piksela względem sąsiadujących pikseli. Gradient jasności otrzymany zostaje ze wzoru 2.37:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.37}$$

gdzie Gx jest zmianą jasności danego piksela w kierunku poziomym, a Gy w kierunku pionowym. Kierunek gradientu opisuje wzór 2.38

$$\Theta = atan2(G_y, G_x) \tag{2.38}$$

Krawędzie na obrazie charakteryzują się pewną zmianą jasności na danym obszarze obrazu. Krawędź na obrazie może być położona pod różnym kątem. Opisywany detektor używa czterech filtrów w celu wykrycia pionowych, poziomych oraz ukośnych krawędzi.

Kierunek gradientu jest zaokrąglony do jednego z filtrów kierunku: poziomego ($\Theta = 0°$), pionowego($\Theta = 90°$) lub ukośnego ($\Theta = 45°$ lub $\Theta = 135°$ stopni). Każdy piksel obrazu otrzymuje wartość gradientu (Wzór 2.37) oraz kierunek - zgodny z powyższym filtrem kierunku. W wyniku tego działania zostaje otrzymany obraz gradientowy [16].

2.3.3 Zastosowanie techniki zmniejszania grubości krawędzi

Zmniejszanie grubości krawędzi służy do filtracji krawędzi powstałych po poprzednim kroku. Krawędzie wykryte za pomocą gradientu są rozmyte (nieostre). Dzięki technice zmniejszania grubości krawędzi można odszukać na małym obszarze gradienty o największej wartości. Wszystkie inne, o mniejszej wartości, są ignorowane. Dzięki temu pozostawia się gradienty o największej wartości, które identyfikują najostrzejsze krawędzie. Algorytm operuje na obrazie gradientowym powstałym jako wynik działania algorytmu opisanego w

sekcji 2.3.2 Działa on następująco:

Odczytana zostaje wartość gradientu oraz jego kierunek dla danego piksela. Następnie wartość gradientu porównuje się z wartościami gradientu dla dwóch sąsiednich pikseli. Pierwszy sąsiedni piksel jest wyznaczony przed aktualny kierunek gradientu, a drugi przez kierunek przeciwny. Jeśli wartość danego piksela jest największa spośród pikseli wzdłuż wyżej wymienionych linii, to dany piksel należy do najostrzejszej krawędzi [16].

2.3.4 Filtracja krawędzi o małym gradiencie jasności

Zastosowanie techniki zmniejszania grubości krawędzi pozostawia na obrazie wiele krawędzi, które są wygenerowane przez szum i zmiany kolorów. Kolejnym krokiem jest filtracja powyższych krawędzi, gdyż są one nadmiarowe. W tym celu ustawiany jest pewien próg, zwany progiem małym. Oprócz progu małego ustawiany jest również próg duży w celu wyodrębnienia krawędzi o wysokim gradiencie jasności. Jeśli wartość gradientu dla danej krawędzi jest mniejsza od progu małego, to zostaje ona usunięta. W przypadku, gdy wartość jest większa od progu małego, ale mniejsza od progu dużego, dana krawędź zostaje oznaczona jako słaba. Krawędź oznacza się jako mocną, gdy wartość gradientu dla niej jest większa od wartości progu dużego [16].

2.3.5 Filtracja poprzez histerezę

W wyniku działania algorytmu do tej pory zostały uzyskane słabe oraz mocne krawędzie. Mocne krawędzie zostaną oznaczone jako prawdziwe krawędzie znajdujące się na obrazie. Słabe krawędzie mogą być częścią mocnych krawędzi, ale mogą również zostać wygenerowane przez szum lub zmiany kolorów. Krawędzie tego typu nie są prawdziwymi krawędziami w obrazie, więc w ostatnim kroku algorytmu powinny zostać przefiltrowane. Słabe krawędzie, które są powiązane z mocnymi znajdują się w najbliższym sąsiedztwie mocnych krawędzi. W celu odnalezienia tych powiązań pomiędzy krawędziami zostaje zastosowana analiza spójności krawędzi. Jeśli zostaje zidentyfikowane powiązanie pomiędzy mocną, a słabą krawędzią, to słabą krawędź pozostawia

Rysunek 2.21: a) Oryginalny obraz b) Obraz z wykrytymi krawędziami.

się w obrazie. W przypadku, gdy słaba krawędź nie jest powiązana z żadną mocną krawędzią - zostaje usunięta [16].

2.4 Wyliczanie wrinkle feature

W wyniku działania detektora Cannego, który został opisany w sekcji 2.3 wygenerowany zostaje obraz binarny z wyodrębnionymi krawędziami. Obraz binarny zawiera piksele o dwóch wartościach jasności pikseli. Piksel może być albo biały albo czarny. Krawędzie można rozpoznać jako białe piksele, natomiast czarne piksele oznaczają brak krawędzi (Rysunek 2.21).

Na Rysunku 2.21 widoczne są krawędzie, które wyróżniają tło od twarzy. Ponadto widoczne są poszczególne części twarzy tj. nos, oczy, brwi. Można także zauważyć dodatkowe krawędzie w strefach zmarszczkowych (Sekcja 2.2), które identyfikują zmarszczki. Ilość białych pikseli w strefach zmarszczkowych jest wprost proporcjonalna do ilości zmarszczek na twarzy danej osoby.

Z każdej strefy zmarszczkowej obliczany jest stosunek ilości białych pikseli

do wszystkich pikseli (Wzór 2.39).

$$W_{s1} = \frac{PB_{s1}}{PW_{s1}} \tag{2.39}$$

, gdzie W_{s1} jest stosunkiem białych pikseli do wszystkich w strefie 1, PB_{s1} - suma białych pikseli w strefie 1, PW_{s1} - suma wszystkich pikseli w strefie 1. Analogicznie obliczane są stosunki pikseli dla pozostałych stref. Ostatnim etapem jest sumowanie wszystkich stosunków pikseli (Wzór 2.40).

$$WF = W_{s1} + W_{s2} + W_{s3} + W_{s4} + W_{s5} + W_{s6} \tag{2.40}$$

, gdzie WF to wrinkle feature - parametr określający ilość zmarszczek dla danej osoby.

2.5 Algorytm trenowania

W celu skonstruowania programu wyznaczającego wiek na podstawie tekstury należy wcześniej zbadać zależność ilości zmarszczek od wieku. W tym celu należy posiadać odpowiednią bazę zdjęć, na których można przeprowadzić wyżej wspomniane badania.

Jest wiele darmowych źródeł obrazów twarzy. Najczęściej używane bazy to FG-NET oraz MORTH II, które zawierają dziesiątki tysięcy zdjęć. Zdjęcia są różnej jakości i nie każda baza zdjęć nadaje się do konkretnych badań [14]. Przy realizacji tej pracy wykorzystana została baza UTKFace, ponieważ w jej przypadku ścieżka pliku każdego obrazu zawiera informacje na temat rzeczywistego wieku. W związku z powyższym z każdego zdjęcia z tej bazy otrzymywano dwie informacje - rzeczywisty wiek danej osoby oraz współczynnik zmarszczek. Następnie dla całej bazy wygenerowano zbiór danych, który w dalszej kolejności był odpowiednio analizowany w celu sprawdzenia zależności pomiędzy współczynnikiem zmarszczek, a wiekiem. Wyżej wymieniony proces nazywany jest także trenowaniem zbioru danych.

Autorzy algorytmu w celu analizy wyżej wygenerowanego zbioru danych

wykorzystali algorytm grupowania - Fuzzy C-Means. W wyniku działania tego algorytmu otrzymuje się dane, dzięki którym na podstawie współczynnika zmarszczek można oszacować wiek [25].

2.6 Grupowanie danych - Fuzzy C-Means oraz wyznaczanie wieku

Grupowanie danych, zwane również klasteryzacją, jest szeroko stosowane w uczeniu maszynowym, rozpoznawaniu wzorców, analizie obrazu, bioinformatyce, kompresji danych czy w grafice komputerowej. Polega na podzieleniu dużego zbioru danych na grupy. Powyższe grupy zawierają dane, które są podobne do siebie [9].

Klasteryzacja nie jest algorytmem, lecz zadaniem, które może zostać wykonane na wiele sposobów. Jest również iteracyjnym procesem odkrywania danych w celu odnalezieniu relacji pomiędzy nimi.

Odpowiedni algorytm grupowania i ustawienia parametrów (w tym parametrów takie jak funkcja odległości lub liczba oczekiwanych grup) zależą od zestawu danych i sposobu wykorzystania wyników. Dane nieraz muszą zostać przefiltrowane lub potrzebna jest zmiana parametrów grupowania w celu osiągnięcia zamierzonego efektu [9].

Algorytmy różnią się pod względem implementacji grupowania czy wydajności. Najczęściej grupa jest definiowana przez jak najmniejszą odległość pomiędzy jej członkami. Zarazem odległość pomiędzy członkami grupy jest głównym parametrem klasteryzacji hierarchicznej. Istnieje również wiele innych modeli grupowania. Jednym z nich jest model centroidowy, który zakłada, że każda grupa jest zdefiniowana przez jeden wektor, posiadający wartość średnią. Istnieją też modele, które opierają się na sieciach neuronowych. Ich podstawą jest założenie, że dane grupują się za pomocą nienadzorowanych sieci neuronowych. Spotykany jest także model oparty o rozkłady statystyczne.

Klasteryzacja może być twarda lub miękka. Twarda klasteryzacja oznacza, że każdy element danych może należeć tylko do jednej grupy. Natomiast

w przypadku miękkiej klasteryzacji każdy element danych może w pewnym stopniu należeć do każdej z grup. Autorzy algorytmu szacowania wieku opisywanego w niniejszej pracy zastosowali grupowanie metodą Fuzzy C-means. Należy ona do grupy modeli centroidowych [9]. W Fuzzy C-means zastosowano klasteryzacje miękką.

Algorytm Fuzzy C-means został stworzony w 1973 przez J. C. Dunn. Została opisana ona w książce „A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters".

Klasteryzacja Fuzzy C-means dzieli zbiór X (Wzór 2.41):

$$X := \{x_1, x_2, x_3, \ldots, x_n\} \subset \mathbb{R} \tag{2.41}$$

W parametrze algorytmu zostaje ustalona liczba grup c. Każda grupa posiada charakterystyczną wartość nazywana centroidem $p_j \subset \mathbb{R}$. Jak zostało wspomniane na początku tego rozdziału, w algorytmie C-means zastosowana jest miękka klasteryzacja. Każdy element danych ma przypisywany wektor U_i przynależności do grupy (Wzór 2.42).

$$U_i = (u_{i1}, u_{i2}, u_{i3}, \ldots, u_{ic}) \tag{2.42}$$

, gdzie u_{i1} oznacza stopień przynależności elementu x_i do grupy 1. Analogicznie u_{i2} oznacza stopień przynależności elementu x_i do grupy 2.

W pierwszym kroku zbiór centroidów P jest inicjalizowany losowo (Wzór 2.43).

$$P^0 = (p_1^0, p_2^0, p_3^0, \ldots, p_c^0) \tag{2.43}$$

W k-tym kroku algorytmu wyliczona jest macierz funkcji przynależności

$U^k = u_{ij}^k$ (Wzór 2.44).

$$u_{ij}^k = \frac{1}{\sum_{k=1}^{c} \left(\frac{\left\| x_i - p_j^k \right\|}{\left\| x_i - p_k^k \right\|} \right)^{\frac{2}{m-1}}} \tag{2.44}$$

Wyrażenie $\|x_i - p_j\|$ oznacza odległość Euklidesową pomiędzy elementem x_i a centroidem p_j. Parametr „m" jest nazywany współczynnikiem rozmycia. Współczynnik ten może być w zakresie od 1 do nieskończoności, jednak w większości przypadków jest równy 2.

W kroku $(k+1)^{th}$ centroid $p_j^{(k+1)}$ jest uaktualniany według Wzoru 2.45.

$$p_j^{(k+1)} = \frac{\sum_{i=1}^{n} u_{ij}^{(k+1)m} x_j}{\sum_{i=1}^{n} u_{ij}^{(k+1)m}} \tag{2.45}$$

Algorytm w kolejnych iteracjach minimalizuje kryterium $J(U, P)$ podane Wzorem 2.46:

$$J(U, P) = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^m \left\| x_i - p_j \right\|^2 \tag{2.46}$$

Iteracje kończą się, gdy zostanie osiągnięty warunek opisany Wzorem 2.47

$$\left\| J^{(k+1)}(U, P) - J^{(k)}(U, P)) \right\| < \epsilon \tag{2.47}$$

, gdzie $\epsilon > 0$. Parametr ϵ jest parametrem ustawianym przed uruchomieniem klasteryzacji. W niektórych implementacjach istnieje możliwość zakończenia działania algorytmu po określonej liczbę iteracji, jeśli wcześniej nie zostanie osiągnięte kryterium ze Wzoru 2.47.

Po zakończeniu działania wyżej opisanego algorytmu zostaje wyliczony wektor przynależności U (Wzór 2.42). Zakładając, że suma wektora przyna-

leżności U_i, dla elementu x_i jest opisana Wzorem 2.48

$$\sum_{j=0}^{j=c} u_{ij} = 1 \tag{2.48}$$

, gdzie c oznacza liczbę klastrów. Element x_i będzie należał do klastru k, jeśli u_{ik} ma największą wartość w wektorze U_i.

W tym akapicie zostanie przedstawiony algorytm trenowania za pomocą Fuzzy C-means. Z każdego zdjęcia „i" otrzymywane są dane dotyczące rzeczywistego wieku A_i oraz parametr wrinkle feature F_i. W algorytmie Fuzzy C-means są grupowane dane zawierające wartości wrinkle feature F_i. Jak wiadomo z powyższego opisu algorytmu Fuzzy C-means, każdy element (w tym przypadku F_i) ma określony stopień przynależności do każdego klastra. Dany element F_i będzie należał do klastra o największym stopniu przynależności. Każde zdjęcie jest grupowane według wyżej opisanego algorytmu.

Każdy klaster ma określony centroid j, który posiada parametr P_{fj} oraz dodatkowy parametr - średnią wartość wieku P_{aj}. P_{fj} oznacza wartość wrinkle feature dla danego centroida. P_{aj} jest wyznaczany ze Wzoru 2.49.

$$P_{aj} = \frac{\sum A_c}{N_j} \tag{2.49}$$

, gdzie A_c to rzeczywisty wiek osoby z obrazu c, który należy do klastru j. Natomiast N_j to liczba zdjęć zaklasyfikowanych do klastru j.

W powyższym akapicie został opisany algorytm trenowania, który generuje n centroidów, które reprezentują każdą grupę. Natomiast osobnym etapem jest szacowanie wieku. Należy podkreślić, że zdjęcia do testowania algorytmu szacowania wieku pochodzą z innej bazy obrazów niż obrazy wykorzystane w treningu. Ma to na celu eliminację błędów w testowaniu algorytmu. W pierwszej fazie działanie algorytmu jest dokładnie takie same jak w sekcjach od 2.1 do 2.4: Algorytm szacowania wieku w pierwszej kolejności dokonuje detekcji twarzy, oczu, ust na danym obrazie O_i. Jeśli powyższe elementy zostały zidentyfikowane, wyznaczane są strefy zmarszczkowe, a na-

stępnie zostaje wyliczony parametr wrinkle feature F_i.

Dla parametru F_i zostaje wyznaczony wektor przynależności do grupy według Wzoru 2.42. Przy założeniu dla wektoru przynależności zgodnym ze Wzorem 2.48 szacowany wiek PA wynosi (Wzór 2.50).

$$PA = \sum_{j=1}^{c} (u_{ij} * P_{aj}) \tag{2.50}$$

, gdzie u_{ij} oznacza przynależność i-tego zdjęcia do j-tego klastra, c oznacza liczbę klastrów. Natomiast P_{aj} jest średnim wiekiem dla j-tego centroida.

Rozdział 3

Modyfikacje metody bazowej

W pracy została odtworzona oryginalna praca autorów, która będzie dalej nazywana metodą bazową. W celu poprawienia wyników metody bazowej wprowadzono kilka modyfikacji, a w kolejnym kroku wyniki zostały porównane z pozostałymi metodami.

Parametrem określającym jakość metody bazowej oraz metod zmodyfikowanych jest średni błąd bezwzględny, który został określony we Wzorze 3.1.

$$MAE = \frac{\sum_{i=1}^{n} |R_i - P_i|}{n} \tag{3.1}$$

, gdzie R_i oznacza wartość rzeczywistą wieku dla i-tego zdjęcia, n oznacza liczbę zdjęć testowych. Natomiast P_i oznacza wartość szacowaną wieku.

3.1 Odjęcie wybranej strefy

Pierwsza modyfikacja polegała na odjęciu jednej wybranej strefy. Detektor Canny oprócz zmarszczek wykrywa także włosy. Podczas przeglądania wielu zdjęć zauważono, że w obrębie strefy zmarszczek nr 2 (Obraz 2.16) wykrywane są części brwi oprócz samych zmarszczek. Generuje to błąd, dlatego wyżej wymienionej strefy nie uwzględniono w obliczeniach. W Sekcji 3.1.1 porównano różnice w algorytmie przed opisaną wyżej modyfikacją i po niej.

3.1.1 Zmiana algorytmu względem metody bazowej

Przed modyfikacją metody bazowej parametr wrinkle feature był obliczany według Wzoru 2.40, który jest umieszczony w Sekcji 2.4.

Po modyfikacji parametr wrinkle feature oblicza się według Wzoru 3.2.

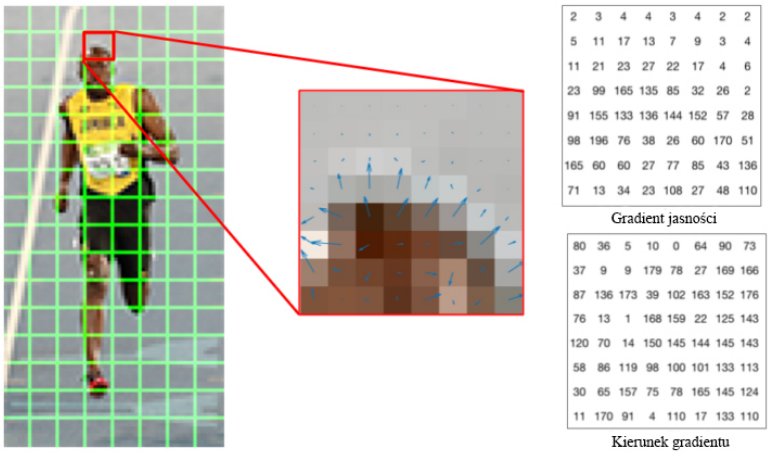
$$WF = W_{s1} + W_{s3} + W_{s4} + W_{s5} + W_{s6} \tag{3.2}$$

Kolejną modyfikacją było zastosowanie metody HOG - Histograms of Oriented Gradients, która została opisana w Sekcji 3.2.

3.2 Zastosowanie metody HOG

Metoda HOG (Histogram of Oriented Gradients) jest deskryptorem cech z obrazu. Deskryptor cech wyodrębnia z obrazu pewne cechy, do których należą między innymi kształty, kolory czy tekstury. W metodzie bazowej rolę deskryptora tego rodzaju spełnia detektor krawędzi Canny (Sekcja 2.3), który wykrywa krawędzie, identyfikowane jako zmarszczki. Z kolei Histogram of Oriented Gradients jest techniką, która liczy wystąpienia kierunków gradientów jasności. Liczenie jest dokonywane tylko w pewnych obszarach obrazów. Metoda HOG została wymyślona i opisana w 1982 przez Roberta K. McConella. Robert K. McConell pracował w ówczesnych czasach dla firmy Wayland Research Inc., a opis jego metody został zawarty w patencie „Method of and apparatus for pattern recognition" [18]. Jednak jego metoda została po raz pierwszy zaimplementowana dopiero w 2005 roku przez Navneeta Dalala i Billa Triggsa [7] z francuskiego instytutu zajmującego się badaniami w zakresie informatyki i automatyki (INRIA - Institut national de recherche en informatique et en automatique). Metoda McConella została wtedy zastosowana do detekcji pieszych na zdjęciach. Później metoda została rozszerzona o wykrywanie ludzi w filmach wideo, ponadto później stosowano ją także do wykrywania zwierząt oraz pojazdów na zdjęciach.

Metoda Histogram of Oriented Gradients opiera się na założeniu, że wygląd i kształt obiektów na obrazie może zostać opisany przez gradient jasności



Rysunek 3.1: Gradient jasności identyfikujący granicę pomiędzy głową a tłem. [17]

oraz jego kierunek. Jest to dobrze widoczne na Rysunku 3.1

Na granicy pomiędzy tłem, a głową sportowca widoczne są duże wartości gradientu jasności. W ten sposób zostaje zidentyfikowana krawędź głowy.

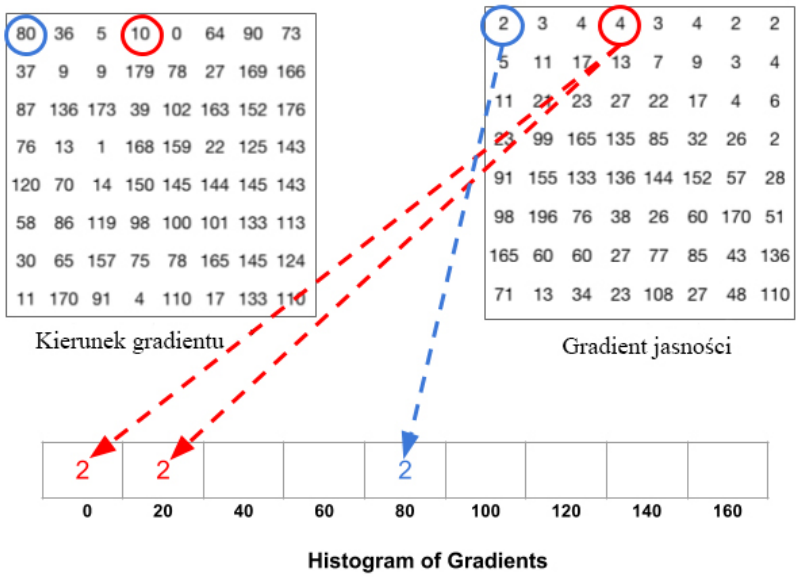
Działanie algorytmu można pokrótce opisać w następujący sposób. Obraz jest dzielony na małe obszary zwane komórkami, które zwykle mają wymiar kilka na kilka pikseli. Dla każdego piksela w komórce tworzony jest histogram kierunków gradientu. Następnie histogramy są łączone w jeden wspólny deskryptor. Dla zwiększenia dokładności lokalne histogramy normalizowane są pod względem kontrastu. Realizowane jest to przez pomiar jasności obszaru zwanego blokiem, który składa się z kilku komórek. Normalizacja zapewnia zmniejszenia niepożądanego efektu generowanego przez różnice w oświetleniu różnych obszarów zdjęcia. Wspomniana sytuacja występuje na przykład, gdy zdjęcie wykonywane jest w nieprawidłowych warunkach oświetleniowych (Rysunek 2.3).

W poniższym akapicie zostanie przedstawiony dokładny opis metody Histogram of Oriented Gradients. W pierwszej fazie obraz powinien zostać przeskalowany, tak aby jego wymiary były podzielne przez rozmiar pojedynczej komórki. (Rysunek 3.2).

Warto wspomnieć, że w wielu metodach generujących deskryptory pierwszą fazą jest korekcja luminacji. Jednak ten krok może zostać pominięty, gdyż

Rysunek 3.2: Komórki o rozmiarze 8x8 pikseli na zdjęciu. [17]

w późniejszym kroku stosowana jest normalizacja, która dokonuje korekcji luminacji cieni oraz świateł. W kolejnej fazie muszą zostać wyznaczone gradienty jasności. Wyznaczenie gradientów jasności zostało szczegółowo opisane w Sekcji 2.3.2. W kolejnej fazie zostają wyznaczone histogramy gradientów dla każdej komórki. Na Rysunku 3.1 został przedstawiony wynik działania tego kroku. Zostają wygenerowane dwie macierze. Pierwsza zawiera informację o gradiencie jasności każdego piksela komórki, natomiast druga - informację o kierunku każdego gradientu. Wartości kierunku gradientu wyrażone są w stopniach. W metodzie Histogram of Oriented Gradients zastosowano kierunki gradientu bez znaku. Oznacza to, że zakres kierunków mieści się w granicach od 0° do 180°. Kierunek gradientu ze znakiem posiadałby zakres od 0° do 180° oraz jego ujemny odpowiednik (od −179° do −1°). Użycie kierunku gradientu bez znaku jest uzasadnione tym, że wartość kierunku o takiej samej wartości, lecz różnych znakach reprezentuje gradient o tym samym kierunku, lecz różnych zwrotach. Dla detekcji kształtów krawędzi czy kształtów istotny jest kierunek, a nie zwrot. Na podstawie kierunku gradientu oraz wartości gradientu jasności zostaje wygenerowany histogram gradientów. W parame-



Rysunek 3.3: Komórki o rozmiarze 8x8 pikseli na zdjęciu. [17]

trze wejściowym algorytmu HOG ustawiana jest ilość komórek histogramu. Najczęściej stosuje się 9 komórek. Każda komórka histogramu odpowiada pewnemu zakresowi kąta kierunku gradientu. Przykładowo dla histogramu o szerokości 9 komórek każda komórka odpowiada następującym kątom: 0°, 20°, 40°, ..., 160° . Na Rysunku 3.3 przedstawiono sposób przypisywania wartości do komórek histogramu. Piksel zaznaczony na niebiesko posiada gradient o wartości 2 oraz kierunek o kącie 80° Wartość gradientu zostaje dodana do komórki histogramu reprezentującego kąt 80°. Piksel zaznaczony na czerwono posiada gradient o wartości 4 oraz kierunek o kącie 10°. Wartość gradientu równa 4 zostaje rozdzielona na dwie proporcjonalne wartości: 2 oraz 2, które są dodawane do komórek reprezentujących kąt 0° oraz 20°. Proporcje zależą od kąta gradientu i od kątów w komórkach histogramu. Tak jak w powyższym przypadku, jeśli wartość kąta wynosi 10°, wartości gradientu są przydzielane do komórek reprezentujące dwa kąty (w tym przypadku przedziały reprezentujące kąt od 0° do 20°). W związku z tym, że powyższy kąt wynosi 10°, wartości dla komórek histogramu są obliczane według Wzoru 3.3 oraz Wzoru 3.4:

$$X_a = \frac{A_x - A_a}{A_b - A_a} * G_x \tag{3.3}$$

, gdzie X_a oznacza wartość przydzielaną do komórki reprezentującej mniejszy kąt (tak jak 0° dla powyższego przykładu), A_x oznacza wartość kąta gradientu dla danego piksela (tak jak 10° dla powyższego przykładu), A_a oznacza kąt dla komórki reprezentującej mniejszy kąt (tak jak 0° dla powyższego przykładu), A_b oznacza kąt dla komórki reprezentującej większy kąt (tak jak 20° dla powyższego przykładu), G_x oznacza wartość gradientu dla danego piksela (tak jak 4 dla powyższego przykładu).

$$X_b = \frac{A_b - A_x}{A_b - A_a} * G_x \tag{3.4}$$

, gdzie X_b oznacza wartość przydzieloną dla kąta o większej wartości (tak jak 20° dla powyższego przykładu). Reszta zmiennych odpowiada zmiennym we Wzorze 3.3. W przypadku, gdy kąt kierunku gradientu jest większy od 160° to wartość danego gradientu jest przydzielana proporcjonalnie do komórki 160° oraz 0° zgodnie ze Wzorem 3.4. Jednak wartość komórki 0° zostaje zastąpiona kątem 180°, a wyliczona wartość zostaje przydzielona do komórki reprezentującej kąt 0°. Powyższa procedura jest powtarzana, a kolejne wartości gradientów są dodawane do aktualnych wartości przechowywanych w komórkach histogramu. Końcowy efekt tej fazy dostarcza histogram reprezentujący komórkę. Wartości histogramu dla danej komórki reprezentują część wektoru deskryptora. Deskryptor składa się z n histogramów, gdzie n oznacza liczbę komórek w danym obrazie.

Kolejnym krokiem jest normalizacja bloków. Blok składa się z komórek i zazwyczaj ma wymiary 2x2 komórki. Gradienty jasności są czułe na zmiany oświetlenia na obrazie. Celem tej fazy algorytmu jest normalizacja oświetlania, aby uodpornić deskryptor na zmiany oświetlania. Istnieje wiele sposobów normalizacji. Zostanie tutaj opisana normalizacja za pomocą parametru

L2-norm. Najpierw zostaje wyliczona wartość L2-norm. L2 jest wartością normalizującą wektor v, który reprezentuje wartości deskryptora dla danego bloku. Wspomniany wektor składa się z wartości zawartych w komórkach histogramu. Jak wiadomo z przedstawionego wyżej opisu, każdy histogram opisuje jedną komórkę, a blok zawiera n komórek. Przyjmując, że histogram reprezentujący komórkę posiada 9 wartości oraz rozmiar bloku wynosi 2x2 komórki, można wyznaczyć długość wektora v. Blok ma 2x2 komórki, więc zawiera w sobie 4 komórki. Każda komórka ma 9 wartości histogramu, więc wektor v będzie miał długość $9x4$, czyli 36. L2-norm dla wektora v jest obliczane według Wzoru 3.5

$$\|v\|_2 = \sqrt{\sum_{i=0}^{n} v(i)^2} \tag{3.5}$$

, gdzie $\|v\|_2$ jest wartością L2-norm, natomiast $v(i)$ jest i-tym elementem wektora v. Normalizacja jest realizowana przy zastosowaniu Wzoru

$$v = \frac{v}{\|v\|_2} \tag{3.6}$$

, gdzie $\|v\|_2$ jest wartością L2-norm dla wektora v. Działanie zgodne ze Wzorem 3.6 oznacza, że każdy element wektora v jest dzielony przez wartość $\|v\|_2$. Powyższy proces powtarza się dla wszystkich bloków. W wyniku działania algorytmu Histograms of Oriented Gradients wygenerowany zostaje deskryptor, czyli wektor reprezentujący wartości histogramów wszystkich komórek w obrazie.

Deskryptor HOG posiada kluczową zaletę w stosunku do innych metod ekstrakcji cech. Jest nią odporność metody Histogram of Oriented Gradients na działanie przekształceń geometrycznych dzięki temu, że dokonuje działań na komórkach lokalnych. Przekształcenia geometryczne to działania, które przekształcają figury z jednej przestrzeni geometrycznej w drugą. Przykładem przestrzeni geometrycznej jest przestrzeń euklidesowa czy przestrzeń

rzutowa. Ponadto działania na komórkach lokalnych pozwalają zmniejszyć efekt powstający przy ruchu pieszego. Z uwagi na powyższe zalety ta metoda nadaje się do detekcji ludzi na obrazach.

W niniejszej pracy opisywana metoda posłużyła do wykrywania krawędzi, które informują o zmarszczkach. Dokładny sposób jej zaimplementowania w pracy opisuje Sekcja 3.2.1

3.2.1 HOG - Zastosowanie w projekcie

W poprzedniej Sekcji 3.2 został dokładnie opisany algorytm HOG, w wyniku którego generowany jest deskryptor dla danego obrazu.

Omówiony wyżej deskryptor posłużył do wykrycia ilości zmarszczek w strefach zmarszczkowych. Jak wiadomo z opisu w Sekcji 3.2 metoda HOG pozwala na wykrywanie krawędzi w danym obrazie. Jeśli w danym obszarze jest dużo krawędzi, to w wektorze deskryptora będzie pewna ilość elementów tego wektora o znacznej wartości. Wyżej opisywane, charakterystyczne elementy wektora będą identyfikowały krawędzie w obrazie. Jak wiadomo z Sekcji 2.4 krawędzie mogą zostać zinterpretowane jako zmarszczki. W związku z powyższym zmodyfikowano sposób obliczenia wrinkle feature dla obrazu twarzy opierając się o deskryptor wygenerowany przez algorytm HOG.

W bibliotece OpenCv zaimplementowany jest algorytm HOG, który na podstawie całego obrazu lub jego pewnego obszaru generuje wektor reprezentujący deskryptor. Biblioteka umożliwia ustawienie różnych parametrów HOG-a. Do tych parametrów zalicza się:

- rozmiar komórki w pikselach
- rozmiar bloku w pikselach
- liczba komórek w histogramie
- rozmiar okna - wielkość analizowanego obrazu w pikselach

W celu oszacowania ilości zmarszczek dla danego zdjęcia posłużono się następującym algorytmem.

W pierwszej fazie zostały wyznaczone strefy zmarszczkowe nr 1,3,4,5,6. (Rysunek 2.16). Następnie dla każdej wyżej wymienionej strefy został wyznaczony deskryptor. Dla strefy 1 deskryptor jest opisany zmienną v_1, dla strefy 3 zmienną v_3 i tak dalej. Wrinkle feature jest obliczany ze Wzoru 3.7:

$$WF_{HOG} = \|v_1\|_1 + \|v_3\|_1 + \|v_4\|_1 + \|v_5\|_1 + \|v_6\|_1 \tag{3.7}$$

, gdzie $\|v_1\|_1$ jest parametrem L1-norm dla wektora v_1. Analogicznie $\|v_3\|_1$ jest parametrem L1-norm dla wektora v_3.

Parametr L1-norm dla danego wektora v jest wyznaczany ze Wzoru 3.8:

$$\|v\|_1 = \sum_{i=1}^{n} |v(i)| \tag{3.8}$$

, gdzie $v(i)$ jest i-tym elementem wektora.

Dla każdego zdjęcia ze zbioru podobnie jak w metodzie bazowej 2.3 wyznaczana jest para danych WF_{HOG} (Wzór 3.7) oraz wiek. Następnie wygenerowany zbiór danych jest trenowany za pomocą Fuzzy C-means (Sekcja 2.6) - tak samo jak w metodzie bazowej. Wiek szacowany jest dokładnie tak samo jak w metodzie bazowej. Kolejna modyfikacja metody bazowej została opisana w Sekcji **??**

3.3 Metoda HOG oraz algorytm KNN

3.3.1 Zastosowanie w projekcie

Kolejna modyfikacja polegała na zmianie sposobu wyznaczania wrinkle feature. W Sekcji 3.2.1 przedstawiono sposób wyznaczania wrinkle feature, który polegał na sumowaniu deskryptorów z poszczególnych stref (Wzór 3.7). W celu poprawy wyników uwzględniono deskryptory ze wszystkich stref. Dla każdego zdjęcia ze zbioru wyznaczana jest para danych - wiek oraz wektor.

$\overrightarrow{v_{KNN}}$ składający się z elementów według Wzoru 3.9

$$\overrightarrow{v_{KNN}} = (\|v_1\|_1, \|v_3\|_1, \|v_4\|_1, \|v_5\|_1, \|v_6\|_1) \tag{3.9}$$

W wyniku treningu zbioru zdjęć zostaje wygenerowany zbiór zawierający wyżej wymienioną parę danych. Następnie w celu oszacowania wieku zostaje wykorzystany algorytm KNN, który został opisany w Sekcji 3.3.2.

3.3.2 Grupowanie KNN

Algorytm KNN (ang. k-Nearest Neighbor) to algorytm regresji nieparametrycznej używany w statystyce do prognozowania wartości pewnej zmiennej losowej. Został stworzony w roku 1970. KNN tłumaczone jest na język polski, jako algorytm k- najbliższych sąsiadów. Algorytm KNN na wejściu otrzymuje zbiór danych nazywany zbiorem uczącym. Zbiór uczący zawiera dane zwane obserwacjami. Wspomniane obserwacje są parą danych (Wzór 3.10).

$$O_i = (K_i, V_i) \tag{3.10}$$

, gdzie O_i jest daną obserwacją, K_i - klasą, natomiast V_i - wektorem zmiennych objaśniających. Przykładowo taką jedną obserwację może tworzyć klasa określająca wiek danej osoby, a wektorem może być ilość zmarszczek.

Natomiast zbiór uczący będzie posiadał n powyższych obserwacji na podstawie których można wywnioskować, do jakiej klasy będzie zaliczana obserwacja testowa. Obserwacja testowa to obserwacja, która posiada wektor zmiennych objaśniających i może zostać zaliczona do danej klasy za pomocą algorytmu KNN. Wracając do powyższego przykładu, obserwacja testowa będzie zawierała tylko wektor określający ilość zmarszczek. Natomiast algorytm KNN przypisze powyższą obserwację do klasy (wieku). Przypisywanie do danej klasy jest realizowane przez ocenę podobieństwa obserwacji testowej do zbioru uczącego. Ocena podobieństwa jest zrealizowana poprzez obliczanie odległości pomiędzy wektorami zmiennych objaśniających [11].

Przykładowymi miarami odległości są:

- miara Euklidesowa
- miara Manhattan
- miara Czebyszewa
- miara Minkowskiego

Odległość pomiędzy dwoma wektorami w mierze Euklidesowej jest opisana Wzorem 3.11

$$D(\overrightarrow{n}, \overrightarrow{m}) = \sum_{i=1}^{c} (n(i) - m(i))^2 \tag{3.11}$$

, gdzie $D(\overrightarrow{n}, \overrightarrow{m})$ oznacza odległość pomiędzy dwoma wektorami w mierze Euklidesowej. Natomiast $n(i)$ oznacza i-ty element wektora $\overrightarrow{n}$, $m(i)$ - i-ty element wektora $\overrightarrow{m}$, a c jest długością wektora $\overrightarrow{n}$ oraz $\overrightarrow{m}$.

Odległość pomiędzy dwoma wektorami w mierze Manhattan jest opisana Wzorem 3.12

$$D(\overrightarrow{n}, \overrightarrow{m}) = \sum_{i=1}^{c} |n(i) - m(i)| \tag{3.12}$$

, gdzie $D(\overrightarrow{n}, \overrightarrow{m})$ oznacza odległość pomiędzy dwoma wektorami w mierze Euklidesowej. Natomiast $n(i)$ oznacza i-ty element wektora $\overrightarrow{n}$, $m(i)$ - i-ty element wektora $\overrightarrow{m}$, a c jest długością wektora $\overrightarrow{n}$ oraz $\overrightarrow{m}$.

Odległość pomiędzy dwoma wektorami w mierze Czebyszewa jest opisana Wzorem 3.12

$$D(\overrightarrow{n}, \overrightarrow{m}) = \max_{i=1:n}(|n(i) - m(i)|) \tag{3.13}$$

, gdzie $D(\overrightarrow{n}, \overrightarrow{m})$ oznacza odległość pomiędzy dwoma wektorami w mierze Euklidesowej. Natomiast $n(i)$ oznacza i-ty element wektora $\overrightarrow{n}$, $m(i)$ - i-ty element wektora $\overrightarrow{m}$, a c jest długością wektora $\overrightarrow{n}$ oraz $\overrightarrow{m}$.

Odległość pomiędzy dwoma wektorami w mierze Minkowskiego jest opi-

sana Wzorem 3.14

$$D(\overrightarrow{n}, \overrightarrow{m}) = (\sum_{i=1}^{c} |n(i) - m(i)|^p)^{\frac{1}{p}} \tag{3.14}$$

Zmienne są dokładnie takie same jak we Wzorze 3.11. , gdzie $D(\overrightarrow{n}, \overrightarrow{m})$ oznacza odległość pomiędzy dwoma wektorami w mierze Euklidesowej. Natomiast $n(i)$ oznacza i-ty element wektora $\overrightarrow{n}$, $m(i)$ - i-ty element wektora $\overrightarrow{m}$, c jest długością wektora $\overrightarrow{n}$ oraz $\overrightarrow{m}$. Parametr p nazywany jest dystansem Minkowskiego.

W celu zmniejszenia błędów klasyfikacji w algorytmie KNN może być zastosowana standaryzacja lub normalizacja danych. Zastosowanie wyżej wymienionych technik pozwala na zmniejszenie dominacji wartości wektorów, których wartość jest znacznie większa lub mniejsza od ogólnej średniej. Przykładowo, przy założeniu, że wartością objaśniającą byłaby długość wyrażona w metrach i średnia powyższej wartości w całym zbiorze uczącym wyniosłaby 1 m, to dominującą wartością w tym zbiorze byłaby np. wartość 100 metrów.

Standaryzacja ma na celu obliczenie nowych wartości elementów wektorów ze zbioru uczącego. Standaryzacja jest zrealizowana poprzez Wzór 3.15:

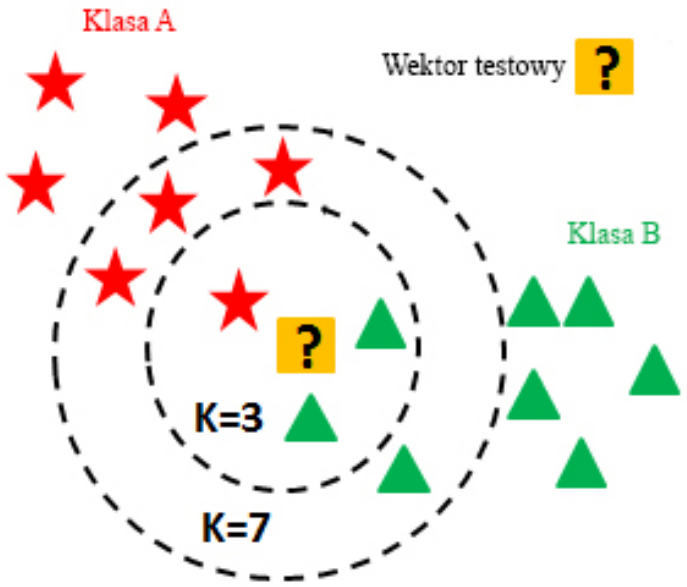
$$v_u(i) = \frac{v_u(i) - m(v_u)}{\sigma(v_u)} \tag{3.15}$$

, gdzie v_u jest wektorem zmiennej objaśniającej ze zbioru uczącego, u - indeksem wektora zmiennej objaśniającej, i jest elementem wektora v_u, $m(v_u)$ jest średnią elementów wektora v_u. Natomiast $\sigma(v_u)$ jest odchyleniem standardowym elementów wektora v_u.

Normalizacja generuje wartości elementów wektorów ze zbioru uczącego, tak aby powyższe wartości były z przedziału od 0 do 1 (Wzór 3.16).

$$v_u(i) = \frac{v_u(i) - \min(v_u)}{\max(v_u) - \min(v_u)} \tag{3.16}$$

, gdzie v_u jest wektorem zmiennej objaśniającej ze zbioru uczącego, u - indek-



Rysunek 3.4: Przykład klasyfikacji KNN [21].

sem wektora zmiennej objaśniającej, i jest elementem wektora v_u, $\max(v_u)$ oznacza maksymalną wartość spośród elementów wektora v_u. Natomiast $\min(v_u)$ minimalną wartość spośród elementów wektora v_u.

Powyżej zostały objaśnione terminy i idea algorytmu KNN. W tym akapicie zostanie przedstawiony sam algorytm. Tak jak zostało opisane wyżej - algorytm otrzymuję zbiór uczący zawierający obserwację. Następnie zbiór uczący może być standaryzowany lub normalizowany. Ponadto algorytm otrzymuje parametr k. W celu wywnioskowania, do której klasy należy obserwacja testowa zawierająca wektor V_i, szukanych jest k najbliższych wektorów (wektorów sąsiadów) ze zbioru uczącego (według kryterium odległości). Obserwacja testowa zostaje przydzielona do klasy, która najczęściej występowała wśród k sąsiednich obserwacji. Problem klasyfikacji algorytmem KNN jest pokazany na Rysunku 3.4

Na Rysunku 3.4 wyróżnione są wektory ze zbioru uczącego należące do klasy „A” oraz klasy „B”. Ponadto na środku Rysunku 3.4 znajduje się wektor, który ma zostać sklasyfikowany do jednej z wyżej wymienionych klas. Dla parametru „k” równego 3 wektor testowy zostanie przypisany do klasy

„B", gdyż z 3 najbliższych wektorów 2 należą do klasy „B". W przypadku, gdy parametr „k" będzie równy 7 wektor testowy zostanie przypisany do klasy „A". Spośród siedmiu najbliższych sąsiadów, cztery wektory należą do klasy „A", natomiast trzy - do klasy „B".

Wybór parametru k jest zależny od rodzaju danych. Im większa wartość k, tym mniejszy wpływ na proces klasyfikacji ma szum, który określa błędne dane w zbiorze uczącym. Istnieją metody dobierające optymalną wartość k dla danego zbioru uczącego. Do jednej z nich należy optymalizacja hiperparametryczna.

Na końcu Sekcji 3.3.1 został przedstawiony sposób, w jaki generowany jest zbiór uczący. Szacowanie wieku metodą KNN w pierwszym kroku polega na stworzeniu wektoru, który został opisany Wzorem 3.9. Następnym krokiem jest przydzielenie danego wektoru do klasy (wieku).

W Sekcji 4 zostaną przybliżone metody przeprowadzanie badań oraz ich wyniki.

Rozdział 4

Badania

W niniejszym rozdziale zostaną przedstawione badania metody bazowej oraz ich modyfikacji. Sekcja 4.1 prezentuje środowisko pracy, a dokładniej mówiąc wszystkie niezbędne narzędzia i akcesoria, które pozwoliły skutecznie przeprowadzić badania. W Sekcji 4.2 przedstawiono wpływ rozmiaru danego obrazu na wyniki badań. Sekcja 4.3 ukazuje efekty wykrywania zmarszczek przez detektor Canny. W kolejnej Sekcji

4.1 Środowisko badań

W celu przeprowadzenia badań do niniejszej pracy magisterskiej wymagane było stworzenie programu, który wykrywa twarz oraz wyodrębnia cechy z twarzy. Wyżej wymieniony program został napisany w języku Java. Biblioteka, która pomogła wyodrębniać cechy z obrazu, to OpenCv. Sama ta biblioteka jest stworzona w języku C++, jednak istnieją jej modyfikacje, które zostały napisane w innych językach. Najczęściej spotykane są modyfikacje stworzone w języku Java oraz Python. Ponadto w Javie generowano dane dla metody bazowej oraz wszystkich jej modyfikacji. Dodatkowo to w tym języku zrealizowano szacowanie wieku we wszystkich metodach, z pominięciem metody, która stosuje algorytm KNN (Sekcja 3.3).

Grupowanie danych algorytmem Fuzzy C-means zostało zrealizowane za pomocą programu Matlab. Również szacowanie wieku dla metody stosującej

algorytm KNN przeprowadzono za pomocą wyżej wymienionego programu.

Dane z treningu były generowane do plików CSV oraz JSON. Ponadto większość kluczowych operacji była logowana do plików. Do tego celu posłużyła popularna biblioteka Log4j. Generacja plików JSON była zrealizowana przez bibliotekę GSON z pakietu com.google.code.gson, natomiast do generacji plików CSV nie została wykorzystana zewnętrzna biblioteka.

4.2 Rozmiar obrazu

Wymiary obrazów z bazy UTKface mieszczą się w zakresie od 186 pikseli do 1300. Natomiast obszar wykrytej twarzy miał rozmiar od 165 do 450 pikseli. Sprawdzono wskaźnik zmarszczek wygenerowany za pomocą detektora Canny oraz HOG dla jednego zdjęcia ze zbioru treningowego oraz kilku rozmiarów. Dla detektora Canny zmieniano wartość powiększenia zdjęcia od 0,8 do 1,20. Wartość parametru wrinkle feature nie uległa zmianie. Natomiast ponad i poniżej powyższego zakresu powiększenia detektor nie mógł wykryć i prawidłowo zlokalizować oczu.

Dla detektora opartego na algorytmie HOG wrinkle feature był następujący:

- powiększenie 0.8: 11.944
- powiększenie 0.9: 11.813538
- powiększenie 1.0: 12.01074
- powiększenie 1.1: 11.117805
- powiększenie 1.2: 11.125154

Dla detektora opartego na algorytmie HOG wraz z klasyfikatorem KNN wrinkle feature był następujący:

- powiększenie 0.8: [2.455685, 2.4238224, 2.3258488, 2.5089853, 2.22999]
- powiększenie 0.9: [2.3424015, 2.183178, 2.402356, 2.6400175, 2.245584]

- powiększenie 1.0: [2.4636364, 2.3633847, 2.4149206, 2.171613, 2.5971856]
- powiększenie 1.1: [2.3787472, 2.184542, 2.335689, 1.9784055, 2.2404206]
- powiększenie 1.2: [2.3283577, 2.1853786, 2.3087728, 2.0892627, 2.2133815]

Na podstawie wyników można wywnioskować, że wskaźnik zmarszczek nie zwiększa się przy zmianie powiększenia zdjęcia dla metody opartej na detektorze Canny. Natomiast zmiany były zauważalne zarówno dla metody korzystającej z algorytmu HOG, jak również HOG z klasyfikatorem KNN.

Okazało się, że w pewnych warunkach detekcja metodą Hog i KNN nie oblicza współczynnika zmarszczek dla strefy 3 i 6. Wynikało to z rozmiaru obrazu oraz parametrów HOG. Okazało się również, że gdy szerokość obszaru zmarszczek jest mniejsza od szerokości pojedynczego bloku lub długość tego obszaru jest mniejsza od długości pojedynczego bloku, to długość deskryptora wynosi 0. Zgodnie z powyższym suma deskryptora dla opisywanego obszaru jest równa 0. Tracone są w ten sposób informacje o zmarszczkach.

Problem został zauważony dla wymiarów obrazu 165 x 213 pikseli i dla następujących parametrów HOG:

- blok 18 x 18 pikseli
- komorka 9 x 9 pikseli

Analogiczna sytuacja może nastąpić, gdy długość bloku jest większa od długości obszaru zmarszczek.

Można przyjąć pewne minimalne wymiary obrazu twarzy, zakładając, że maksymalna długość i szerokość bloku wyniesie 18 pikseli. W związku z powyższym szerokość i/lub długość strefy zmarszczkowej nie może być mniejsza od 18 pikseli. Jak już wcześniej zostało wspomniane, najbardziej wrażliwe na brak generacji deskryptora są strefy 3 i 6. W pierwszym kroku należało wydedukować, jaki obszar zajmuje w przybliżeniu jedna strefa 3 lub 6 w stosunku do obszaru twarzy. Na podstawie logów aplikacji wygenerowanych w trakcie detekcji stref dla zdjęć z bazy treningowej wyznaczono średni obszar jednej strefy 3 lub 6. Następnie porównano rozmiar obszaru twarzy w pikselach z rozmiarem jednej strefy 3 lub 6. Porównanie wyżej wymienionych wymiarów

przeprowadzono dla kilkudziesięciu zdjęć. Na podstawie uzyskanych danych po prostych obliczeniach stwierdzono, że jedna strefa 3 lub 6 zajmuje od 8% do 15% na szerokość obszaru twarzy oraz od 15% do 25% na długość obszaru twarzy. W związku z powyższym minimalna szerokość bloku wyniosła 8% szerokości obszaru twarzy, natomiast długość wyniosła 15%. Minimalna szerokość obszaru twarzy w pikselach względem szerokości bloku jest opisana Wzorem 4.1.

$$T_{wmin} = \frac{1}{B_{wrmin}} * B_w \tag{4.1}$$

, gdzie B_{wrmin} oznacza minimalną szerokość bloku w stosunku do szerokości obszaru twarzy (w szerokość bloku w pikselach, T_w oznacza minimalną szerokość obszaru twarzy.

Analogicznie minimalna długość obszaru twarzy względem długości bloku jest opisana Wzorem 4.2.

$$T_{hmin} = \frac{1}{B_{hrmin}} * B_h \tag{4.2}$$

, gdzie B_{hrmin} oznacza minimalną długość bloku w stosunku do długości obszaru twarzy (w %), natomiast B_h oznacza długość bloku w pikselach, T_{hmin} oznacza minimalną długość obszaru twarzy.

Wobec powyższego Wzoru 4.1 minimalna szerokość obszaru twarzy powinna być co najmniej 12,5 razy większa od szerokości bloku. Natomiast według Wzoru 4.2 długość obszaru zawierającego twarz powinna być co najmniej 6,7 razy większa od długości bloku.

W drugim kroku należało odnaleźć obszar, jaki zajmuje twarz w stosunku do wymiarów obrazu. W tym celu porównano zdjęcia z bazy treningowej pod kątem wyżej wymienionych obszarów. Podobnie jak wyżej, przeprowadzono porównanie dla kilkudziesięciu zdjęć. Okazało się, że minimalny obszar zajęty przez twarz wyniósł 75 % na szerokość całego obrazu oraz 80 % na długość. W świetle powyższych danych należy stwierdzić, że obszar twarzy jest 1,33 razy węższy oraz 1,25 razy krótszy od obszaru całego obrazu.

Zgodnie z wynikami pierwszej fazy badań, obszar twarzy powinien być

12,5 razy szerszy od szerokości bloku, a jednocześnie jest 1,33 razy węższy od samego obrazu, więc szerokość obrazu powinna być $12,5 * 1,33$, czyli w przybliżeniu 16,7 razy większa od szerokości bloku.

Obliczenia dla minimalnej długości obrazu są analogiczne do powyższych. Obszar twarzy powinien być 6,7 razy dłuższy od długości bloku, a jednocześnie jest 1,25 razy krótszy od długości obrazu. W związku z powyższym długość obrazu powinna być $6,7 * 1,25$, czyli w przybliżeniu 8,4 razy większa od długości bloku.

W celu obliczenia parametru wrinkle feature dla strefy 3 i 6 obrazy, które miały zbyt mały rozmiar zostały powiększone.

4.3 Wykrywanie zmarszczek przez detektor Canny oraz Histogram of Oriented Gradients

W tej Sekcji zostaną przedstawione typowe problemy napotkane podczas detekcji metodą Canny oraz Histogram of Oriented Gradients. W wielu zdjęciach wystąpiły nieprawidłowe detekcje zmarszczek.

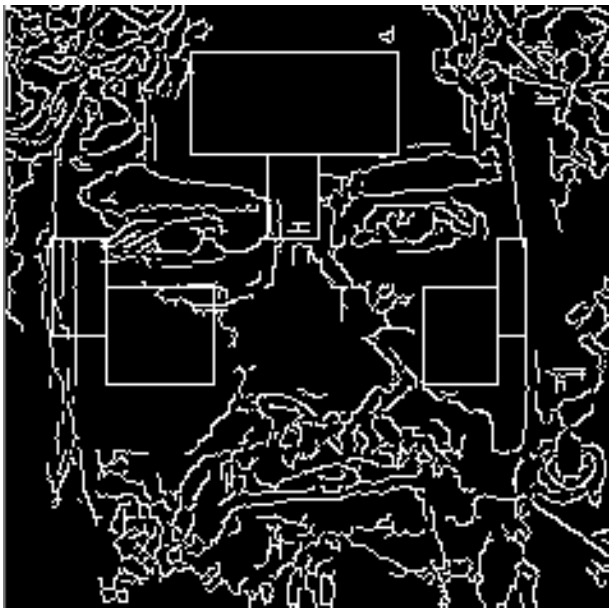
Krawędzie związane tylko ze zmarszczkami są widoczne na Obrazie 4.1 Powyższy Obraz będzie dobrym punktem odniesienia do przypadków błędnych detekcji zmarszczek.

Pierwszym przykładem było wykrycie krawędzi związanych z brwiami (Obraz 4.2) dla osoby w wieku 17 lat.

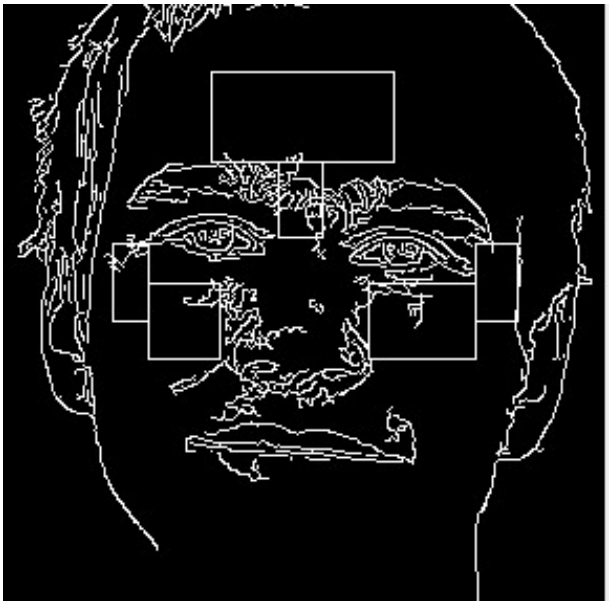
Na obrazie można zauważyć, że ilość krawędzi w strefie 2 (Obraz (2.16) jest duża i nie identyfikują one zmarszczek.

Porównano średni współczynnik zmarszczek ze strefy 2 dla osób w wieku 17 lat z ilością krawędzi wykrytych z tej samej strefy na Obrazie 4.2. Powyższe porównanie ma na celu sprawdzenie wpływu nieprawidłowo wykrytych krawędzi na współczynnik zmarszczek. Średni współczynnik zmarszczek w strefie 2 dla osób w wieku 17 lat wyniosła:

- Dla HOG-a (parametry takie i taki)



Rysunek 4.1: Przykład prawidłowej detekcji zmarszczek.



Rysunek 4.2: Przykład błędnej identyfikacji zmarszczek w strefie 2.

- Dla Canny

Natomiast dla zdjęcia zawierającego nieprawidłowo wykryte krawędzie (Obraz) ilość zmarszczek wyniosła

- Dla HOG-a (parametry takie i taki)
- Dla Canny

W związku z powyższym różnica pomiędzy średnią ilością zmarszczek w strefie 2 dla wieku 17 lat ,a ilością dla obrazu zawierającego nieprawidłowe wykryte krawędzie wynosi (Obraz **??**:

- Dla HOG-a (parametry takie i taki)
- Dla Canny

Kolejnym problemem okazała się grzywka nachodząca na obszar strefy 1. Na obrazie (ref) można zauważyć wykryte krawędzie, które identyfikują włosy. Obraz (ref) należy do x letniej osoby. Podobnie, jak powyżej, dokonano porównanie, które ma na celu sprawdzenie wpływu nieprawidłowo wykrytych krawędzi na współczynnik zmarszczek. Do powyższego celu obliczono średni współczynnik zmarszczek ze strefy 2 dla osób w wieku x lat, który wyniósł:

- Dla HOG-a (parametry takie i taki)
- Dla Canny

Natomiast dla zdjęcia zawierającego nieprawidłowo wykryte krawędzie (Obraz) ilość zmarszczek wyniosła

- Dla HOG-a (parametry takie i taki)
- Dla Canny

Różnica pomiędzy średnią ilością zmarszczek w strefie 2 dla wieku x lat ,a ilością dla obrazu zawierającego nieprawidłowe wykryte krawędzie wynosi (Obraz **??**:

- Dla HOG-a (parametry takie i taki)
- Dla Canny

4.4 Statystyki z działania programu

W tej sekcji zostaną przedstawione ogólne statystyki z działania programu. Baza UTKface zawiera x zdjęć. Detektor twarzy był w stanie wyznaczyć obszar twarzy dla x obrazów. Jednak spośród x obrazów tylko x brało udział w liczeniu zmarszczek, ponieważ jedynie w tej grupie algorytm poprawnie wykrył obszar nosa oraz oczu, które są niezbędne do dalszych obliczeń. Średnia zajętość RAM-u przez program wyniosła 790 MB. Natomiast maksymalnie program zajmował 1974 MB pamięci RAM.

Na podstawie logów aplikacji sprawdzono czasy przetwarzania poszczególnych części algorytmów.

Średni czas wykrywania twarzy trwał x sekund, maksymalny czas wykrywania wynosił x sekund, natomiast minimalny czas wyniósł x sekund.

Średni czas generacji wrinkle feature obrazu po wykryciu twarzy wyniósł x sekund dla metody bazowej. Dla metody, w której została odjęta strefa x sekund, dla metody HOG x sekund, natomiast dla metody HOG wraz z klasyfikacją KNN x sekund.

Maksymalny czas generacji wrinkle feature obrazu po wykryciu twarzy wyniósł x sekund dla metody bazowej. Dla metody, w której została odjęta strefa x sekund, dla metody HOG x sekund, natomiast dla metody HOG wraz z klasyfikacją KNN x sekund.

Minimalny czas generacji wrinkle feature obrazu po wykryciu twarzy wyniósł x sekund dla metody bazowej. Dla metody, w której została odjęta strefa x sekund, dla metody HOG x sekund, natomiast dla metody HOG wraz z klasyfikacją KNN x sekund.

W związku z powyższym najszybciej generował wrinkle feature algorytm x.

Warto porównać szybkość szacowania wieku na podstawie danego wrinkle feature.

Średni czas szacowania wieku dla danego obrazu po wykryciu twarzy wyniósł x sekund dla metody bazowej. Dla metody, w której została odjęta strefa x sekund, dla metody HOG x sekund, natomiast dla metody HOG wraz z klasyfikacją KNN x sekund.

Maksymalny czas szacowania wieku dla danego obrazu po wykryciu twarzy wyniósł x sekund dla metody bazowej. Dla metody, w której została odjęta strefa x sekund, dla metody HOG x sekund, natomiast dla metody HOG wraz z klasyfikacją KNN x sekund.

Minimalny czas szacowania wieku dla danego obrazu po wykryciu twarzy wyniósł x sekund dla metody bazowej. Dla metody, w której została odjęta strefa x sekund, dla metody HOG x sekund, natomiast dla metody HOG wraz z klasyfikacją KNN x sekund.

Najszybciej szacował wiek algorytm x. Warto podkreślić, że szacowanie wieku dla metody HOG wraz z klasyfikacją KNN realizował program realizowany przez skrypt w Matlabie. Reszta algorytmów szacowania wieku została zrealizowana w programie w języku Java. W związku z powyższym porównanie szybkości działania algorytmu zaimplementowanego w Matlabie do algorytmu zaimplementowanego w Javie może być niedokładne.

4.5 Skuteczność poszczególnych metod szacowania wieku

Do oceny skuteczności szacowania wieku poszczególnych metod potrzebna jest baza treningowa oraz testowa. Baza testowa może pochodzić od części danych treningowych. Istnieje kilka metod rozdzielenia jednej bazy na bazę treningową oraz testową. Jednym z nich jest rozdzielenie bazy w oparciu o stosunek. Oznacza, to, że uzyskany zbiór treningowy będzie wynosił x procent np. 90%, a zbiór testowy odpowiednio $100\% - x$, np 10%.

Drugi ze sposobów to schemat CV-n, zwany również walidacją krzyżową. Zakłada ona, że zbiór danych zostanie podzielony na n-równych podzbiorów. Następnie jest wykonywane n iteracji. W każdej iteracji jeden podzbiór zostaje zbiorem testowym, natomiast reszta podzbiorów tworzy zbiór treningowy. Każda iteracja daje wynik działania klasyfikatora. W wyniku „n" iteracji otrzymuje się uśredniony wynik działania klasyfikatora.

Metoda Leave-One-Out zakłada wykluczenie jednego obiektu, który reprezentuje zbiór testowy. Natomiast reszta obiektów zostaje zakwalifikowana

do zbioru treningowego. Po przetestowaniu klasyfikatora powyższym obiektem otrzymuje się wynik skuteczności klasyfikacji. Proces ten jest powtarzany dla wszystkich obiektów. Metoda Leave-One-Out działa dokładnie tak samo jak metoda CV-n dla n równego liczbie obiektów w zbiorze danych.

Przy obliczaniu skuteczności klasyfikacji porównywano parametr MAE Wzór 3.1. Porównywano 100 zdjęć dla każdej metody. Natomiast każda metoda miała zmieniane parametry algorytmów. W przypadku metody z algorytmem grupowania Fuzzy C-means zmianie ulegały parametry wyżej wymienionego algorytmu. Analogicznie zmieniano parametry dla metody HOG i klasyfikatora KNN.

4.5.1 Metoda bazowa

Pierwsza została przetestowana metoda bazowa. Przetestowano dokładność szacowania wieku zmieniając parametry algorytmu trenującego Fuzzy C-means:

rozmycia, $\epsilon, ilościgrup, liczbyiteracji$

Przykładowy wynik testowania metody został przedstawiony na (obrazie?, tabeli). Dla parametrów rozmycie = 2.0, liczba iteracji = 100

$orazliczbagrup = 10błądMAEwyniósł12,75lat.Następniezmienionorozmyciedowartości3.0$

W kolejnym kroku przetestowano wpływ parametru $\epsilon dlametodyoryginalnej.Dlaparametr$ $2.0, liczbaiteracji = 100\epsilon = 1e{-}6orazliczbagrup = 10błądMAEwyniósł11,19lat.Zmianapar$ $7spowodowała, żeparametrMAEwyniósł11,24lat.Dlawartości1e{-}8parametru\epsilonparametrM$ $7oraz1e-8.$

4.5.2 Odjęcie strefy 2

4.5.3 Metoda HOG

4.5.4 Metoda HOG + KNN

4.6 Podsumowanie

4.7 Wnioski

Rozdział 5

Podsumowanie

Bibliografia

[1] T. F. Cootes A. Lanitis, Ch. J. Taylor. *Toward automatic simulation of aging effects on face images*. IEEE Transactions of Pattern Analysis and Machine Intelligence, 2002.

[2] Abid K. Alexander Mordvintsev. Face Detection using Haar Cascades . `https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html`, 2019. [Online; Dostęp: 28.09.19].

[3] Munish Kumar Ashu Kumar, Amandeep Kaur. *Face detection techniques: a review*. Springer, 2018.

[4] BADGERATI. https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/, 2010. [Online; Dostęp: 28.09.19].

[5] Brent Bergherm. https://brentbergherm.com/cie-chromaticity-diagram/, 201. [Online; Dostęp: 10.10.19].

[6] Coutaz J Crowley JL. *Vision for man-machine interaction*. Robotics and Autonomous Systems, 1997.

[7] Navneet Dalal, Bill Triggs. *Histograms of Oriented Gradients for Human Detection*. Conference on Computer Vision Pattern Recognition, 2005.

[8] M.M. Dehshibi, A. Bastanfard. *A new algorithm for age recognition from facial images*. Signal Processing, 2010.

[9] Brian Everitt. *Cluster analysis*. Wiley, 2011.

[10] Erik Fritts. https://www.videomaker.com/article/c03/18848-tell-your-story-more-effectively-with-the-correct -application-of-hard-and-soft-light, 2017. [Online; Dostęp: 07.10.19].

[11] Alvador Garcia, Joaquin Derrac, Jose Cano, Francisco Herrera. *Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study.* IEEE, 2012.

[12] Robert Hirsch. *Exploring Colour Photography: A Complete Guide.* Laurence King Publishing, 2004.

[13] Noor A. Ibraheem, Mokhtar M. Hasan, Rafiqul Z. Khan, Pramod K. Mishr. *Understanding Color Models: A Review.* ARPN Journal of Science and Technology, 2012.

[14] Vladimir Khryashchev, Alexander Ganin, Olga Stepanova, Anton Lebedev. *Age Estimation from Face Images: Challenging Problem for Audience Measurement Systems.* Yaroslavl State University, 2014.

[15] Wan-Chi Siu Kwok-Wai Wong, Kin-Man Lam. *Age Estimation from Face Images: Challenging Problem for Audience Measurement Systems.* Centre for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hung Hom, Hong Kong, 2000.

[16] Xiaoju Ma, Bo Li, Ying Zhang, Ming Yan. *The Canny Edge Detection and Its Improvement.* Springer, 2012.

[17] Satya Mallick. https://www.learnopencv.com/histogram-of-oriented-gradients/, 2016. [Online; Dostęp: 21.10.19].

[18] Robert K. McConnell. Method of and apparatus for pattern recognition, 1982. Patent nr US4567610.

[19] Henrik Skov Midtiby. Example: Rgb color mixing, 2019. [Online; Dostęp: 28.09.19].

[20] R. Chellappa N. Ramanathan. *Face verification across age progression.* IEEE Transactions on Image Processing, 2006.

[21] Avinash Navlani. https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn, 2018. [Online; Dostęp: 22.10.19].

[22] Moses Olafenwa. Przykład rozpoznawania obrazów. `https://towardsdatascience.com/object-detection-with-10-lines-of-code-d6cb4d86f606/`, 2018. [Online; Dostęp: 30.09.18].

[23] M. Jone P. Viola. *Robust Real-Time Object Detection.* International Journal of Computer Vision, 2002.

[24] Michael Jones Paul Viola. *Rapid Object Detection using a Boosted Cascade of Simple Features.* Accepted conference on computer vision and pattern recognition, 2001.

[25] Rituparna Sahaa Ranjan Janaa, Debaleena Dattaa. *Age Estimation from Face Image using Wrinkle Features.* RCC Institute of Information Technology, 2014.

[26] Qaim Mehdi Rizvi. *A Review on Face Detection Methods.* Qassim University, 2011.

[27] Przemyslaw Spurek. http://analizaobrazu.x25.pl/articles/11, 2018. [Online; Dostęp: 19.10.19].

[28] Richard Szeliski. *Computer Vision: Algorithms and Applications.* Springer, 2011.

[29] Piotr Wilkowski. *Wykorzystanie algorytmu detekcji i lokalizacji w zadaniu chwytania.* Politechnika Warszawska, Wydział Elektroniki I Technik Informacyjnych, 2009.

[30] K. Smith-Miles X. Geng, Z. Zhou. *Automatic age estimation based on facial aging patterns.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007.

[31] N. da V. Lobo Y. H. Kwon. *Age classification from facial images. Computer Vision and Image Understanding.* 1999.

Dodatki

Dokumentacja techniczna

Spis skrótów i symboli

FCM *Fuzzy C-means* - metoda klasteryzacji miękkiej (rozmytego).

HOG *Histograms of Oriented Gradients* - metoda generacji deskryptora obrazu

KNN *k-nearest neighbors* - algorytm regresji lub klasyfikacji.

Zawartość dołączonej płyty

Do pracy dołączona jest płyta CD z następującą zawartością:

- praca w formacie `pdf`,
- źródła programu,
- zbiory danych użyte w eksperymentach.

Spis rysunków

1.1 Przykład rozpoznawania obiektów na zdjęciu ulicy. [22] 2

2.1 Faza 1 algorytmu 5
2.2 Faza 2 algorytmu 5
2.3 Przykład twarzy oświetlonej silnym (twarz po lewej) oraz miękkim światłem. [10] 7
2.4 Różne techniki wykrywania twarzy. [3] 8
2.5 Przykład rozpoznawania obiektów na zdjęciu ulicy. [5] 11
2.6 Mieszanie kanałów RGB [19]. 12
2.7 Kolor R=153 G=217 B=234. 12
2.8 Filtr Haara a) krawędziowy b) liniowy c) szachownica [2] . . . 14
2.9 Filtr Haara nałożony na krawędź twarzy [2] 15
2.10 Tabela jasności poszczególnych pikseli przed zastosowaniem całkowania [4] 15
2.11 Filtr Haara nałożony na krawędź twarzy [4] 15
2.12 Sumowanie okna [4] 16
2.13 Kaskada klasyfikatorów. [29] 18
2.14 Macierz pomyłek. 19
2.15 Wykryty nos oraz oczy. 20
2.16 Strefy zmarszczkowe widoczne w białych prostokątach. 21
2.17 Wyznaczenie strefy znajdującej się na czole. 22
2.18 Pomocnicze współrzędne do wyliczania stref zmarszczkowych. 22
2.19 Punkty wyznaczające położenie stref. 23
2.20 a) Obraz z nieznaczną ilością szumów b) Obraz ze znaczną ilością szumów. [27] 27

2.21 a) Oryginalny obraz b) Obraz z wykrytymi krawędziami. . . . 30

3.1 Gradient jasności identyfikujący granicę pomiędzy głową a tłem. [17] 39
3.2 Komórki o rozmiarze 8x8 pikseli na zdjęciu. [17] 40
3.3 Komórki o rozmiarze 8x8 pikseli na zdjęciu. [17] 41
3.4 Przykład klasyfikacji KNN [21]. 49

4.1 Przykład prawidłowej detekcji zmarszczek. 56
4.2 Przykład błędnej identyfikacji zmarszczek w strefie 2. 56

Spis tablic