

Sprawozdanie z przedmiotu

Komputerowe Przetwarzanie Obrazów

Krupnik Mateusz 285608

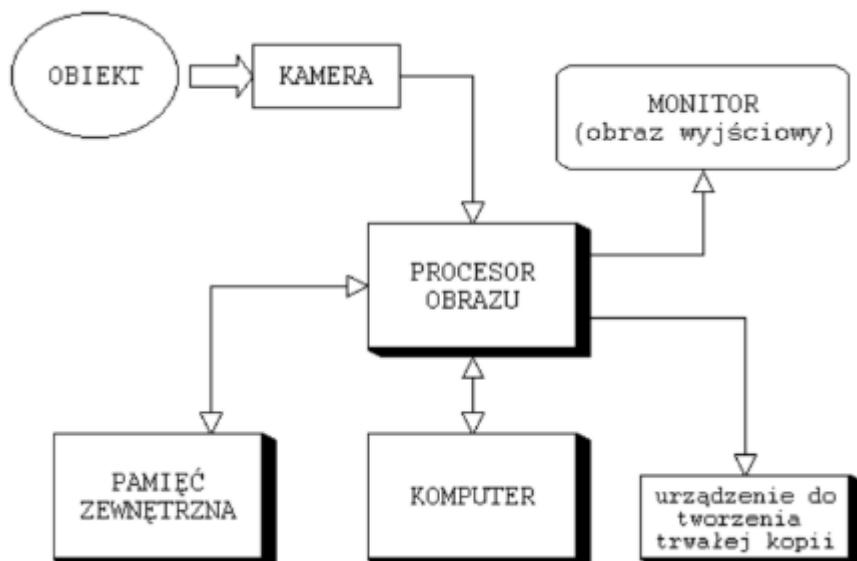
1. Wprowadzenie do zagadnień przetwarzania obrazów.

a. Obraz

Obraz w przypadku komputerowego przetwarzania jest sygnałem dwu lub trzy wymiarowym. Jest to cyfrowa reprezentacja postrzegania, patrzenia człowieka. Podobnie jak w przypadku widzenia naturalnego jak i sztucznego proces przetwarzania lub analizy obrazów przebiega według szeregu operacji jak:

- recepcja (akwizycja),
- przetwarzanie,
- analiza,
- rozpoznanie lub interpretacja.

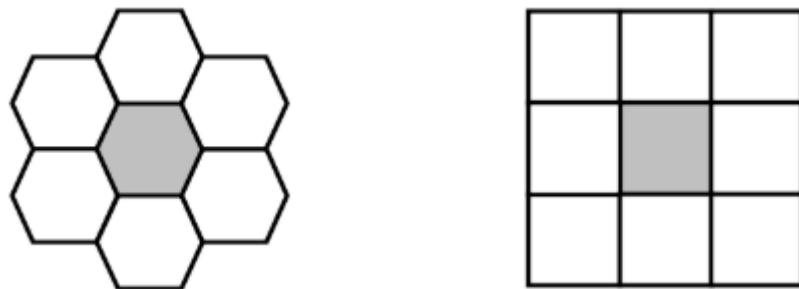
Tor cyfrowego przetwarzania obrazów składa się z kamery rejestrującej natężenie światła padającego na matryce, procesora graficznego, pamięci zewnętrznej do zapisu danych oraz monitora lub urządzenia do tworzenia trwałych kopii obrazu. Moduł wejściowy zamienia sygnał analogowy na cyfrowy, w celu dalszego przetwarzania na komputerze z procesorem obliczeniowym. Dokonując operacji przetwarzania na obrazach w wyniku otrzymujemy również obraz. Analiza obrazu dostarcza danych jakościowych i ilościowych pewnych cech obrazów.



Rysunek 1.1. System cyfrowego przetwarzania obrazów.

b. Dyskretna postać obrazów

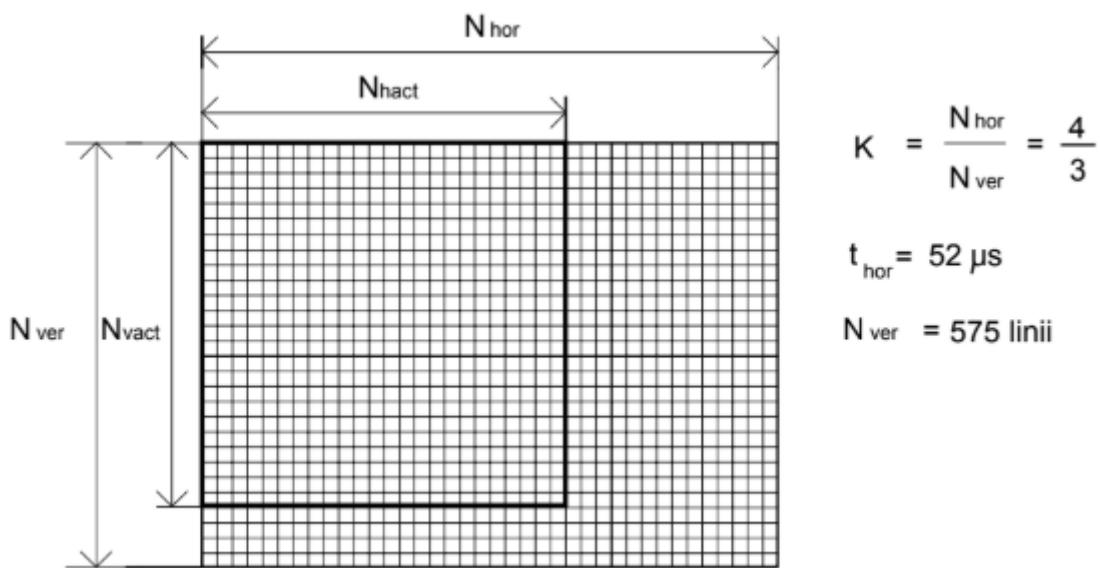
Obrazu mogą być zapisane w postaci funkcji dwóch lub trzech zmiennych, jednak najczęściej operuje się na obrazach dwuwymiarowych. W postaci cyfrowej obrazy dwuwymiarowe można określić według różnych siatek, jednak najczęściej występują siatki heksagonalne i kwadratowej. Najpopularniejsza i najbardziej intuicyjna w zastosowaniu jest siatka kwadratowa.



Rysunek 1.2. Przykłady siatek, heksagonalnej i prostokątnej.

Siatka określana jest też mianem rastra, a jej rozmiary wpływają na częstotliwość próbkowania sygnału wizyjnego i można ją określić za pomocą wzoru poniżej, parametry jak na rysunku poniżej.

$$f = \frac{K \cdot N_{ver}}{t_{hor}} \text{ [MHz]}$$



Rysunek 1.3. Siatka typowego obrazu.

Wybór siatki wpływa na rozdzielczość, a więc zdolność do rozpoznawania szczegółów, jednak powoduje ona wzrost objętości i czasu przetwarzania. Z wielkością siatki kojarzymy rozdzielcość przestrzenną.

Każdy z elementów takiego rastra zawiera informacje o kolorze, zapisywanej w postaci cyfrowej o określonej liczbie stanów. Ta liczba nazywana jest ilością kolorów i może być rozumiana jako liczba bitów przeznaczonych na zapamiętanie stanu jednego elementu (bpp – bits per pixel). Poziomy te określają rozdzielcość poziomów szarości. Przykładowe formaty zapisu kolorów:

- Binarny – czarno-biały, 1 bit,
- Monochromatyczny -skala szarości, 8 bitów – 256 poziomów,
- Kolorowy (24 lub 32 bpp) – standardowo 8 bitów na nasycenie każdej z braw podstawowych RGB.

To co rozumiemy przez barwę możemy zdefiniować przez:

- Odcień (kolor, ton, Hue) – różnica jakościowa barwy określana przez dominującą długość fali,
- Nasycenie (Saturation) – odstępstwo barwy od bieli,
- Jasność (Value) – określa czy barwa jest bliższa bieli czy czerni.

Powыższe opisy pozwalają zdefiniować modele kolorów:

- RGB – model popularny w grafice komputerowej, model addytywny,
- HSV – model bardziej intuicyjny dla człowieka,
- CMYK – stosowany w poligrafii, model substraktywny z dodatkiem czarnego.

Podczas dyskretyzacji dokonywany jest podział przestrzenny, a następnie dla każdego elementu dokonywana jest dyskretyzacja poziomów jasności. Stąd przyjęta liczba bitów zamienia analogową wartość natężenia światła na odpowiadający jej poziom.

Rozdzielcość poziomów jasności można zmniejszyć według poniższej formuły. L' oznacza nowe poziomu jasności, L poziomy przed zmianą, B i B' – liczbę bitów przed i po operacji.

$$L'(m,n) = \left\lceil \text{round} \left(\frac{L(m,n) - \left(\frac{\delta}{2} - 1 \right)}{\delta} \right) \right\rceil \cdot \delta + \left(\frac{\delta}{2} - 1 \right), \quad \text{gdzie } \delta = \frac{2^B}{2^{B'}}$$

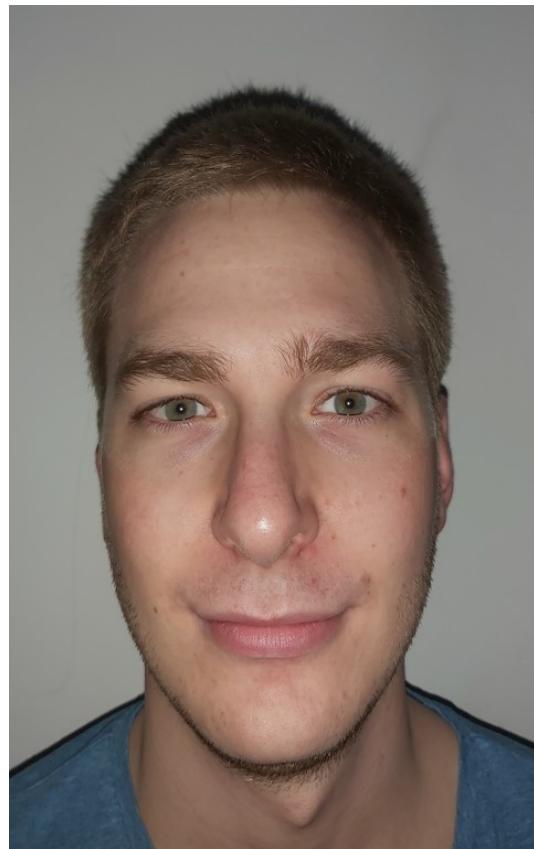
2. Akwizycja obrazu do przetwarzania w ramach sprawozdania.

Obraz wykorzystany do przetwarzania podczas sprawozdania został uzyskany przy pomocy telefonu komórkowego. Zdjęcie zapisane przez telefon w rozszerzeniu .jpg zostało następnie za pomocą skryptu „Przeskalowanie_i_zapis.m” przycięte do odpowiedniego kadru i zapisane w rozszerzeniach .bmp, .tiff i .jpg po uprzednim przeskalowaniu do rozmiarów 768x480.

Takie zmniejszenie rozmiarów zdjęcia pozwoli na efektywne jego przetwarzanie. W skrypcie za pomocą polecenia `iminfo(„obraz”)` wypisywane są dostępne informacje o zdjęciu. Poniżej przedstawione zostaną informacje dla zdjęcia oryginalnego i po obróbce.



Rysunek 2.1. Zdjęcie pozyskane za pomocą telefonu.



Rysunek 2.2. Zdjęcie po obróbce

Widoczne w tabeli parametry pokazują jak znaczco zmniejsza się rozmiar obrazu w zależności od siatki. Początkowo zdjęcie posiadało rozmiar 4032x3024 a docelowo uzyskało 480x768 pikseli. Oba obrazy posiadały tą samą głębie kolorów, czyli standardowe 16,7mln dla 24 bitów. W tabelach widoczne są dodatkowe parametry jak maski kolorów, urządzenie dokonujące akwizycji, orientacja zdjęcia, rozdzielcość i jednostka czy data utworzenia.

Tab. 2.1. Informacje o analizowanym obrazie.

Zdjęcie oryginalne	Zdjęcie po obróbce
Filename: 'D:\Studia\KPO\Laboratoria\zdj_org3.jpg' FileModDate: '03-May-2020 18:15:09' FileSize: 2295896 Format: 'jpg' FormatVersion: " Width: 4032 Height: 3024 BitDepth: 24 ColorType: 'truecolor' FormatSignature: " NumberOfSamples: 3 CodingMethod: 'Huffman' CodingProcess: 'Sequential' Comment: {} Make: 'samsung' Model: 'SM-A705FN' Orientation: 6	Filename: 'D:\Studia\KPO\Laboratoria\zdj_3.bmp' FileModDate: '03-May-2020 18:19:27' FileSize: 1105974 Format: 'bmp' FormatVersion: 'Version 3 (Microsoft Windows 3.x)' Width: 480 Height: 768 BitDepth: 24 ColorType: 'truecolor' FormatSignature: 'BM' NumColormapEntries: 0 Colormap: [] RedMask: [] GreenMask: [] BlueMask: [] ImageDataOffset: 54

XResolution: 72 YResolution: 72 ResolutionUnit: 'Inch' Software: 'A705FNXXS5ATD1' DateTime: '2020:05:03 18:12:37' YCbCrPositioning: 'Centered' DigitalCamera: [1x1 struct] GPSInfo: [1x1 struct] ExifThumbnail: [1x1 struct]	BitmapHeaderSize: 40 NumPlanes: 1 CompressionType: 'none' BitmapSize: 1105920 HorzResolution: 0 VertResolution: 0 NumColorsUsed: 0 NumImportantColors: 0
---	---

Wszystkie poniższe operacje przetwarzania będą prowadzone na obrazie z rysunku 2.2. Program realizujący te operacje został napisany w programie Matlab z wykorzystaniem dodatku Image Processing Toolbox. Skrypt ma postać jednego pliku z odpowiednimi sekcjami. Uruchomienie jednej sekcji odbywa się za pomocą skrótu klawiszowego Ctrl+Shift+Enter. Można uruchomić także program wykonując wszystkie sekcje za pomocą klawiszu F5.

Do podstawowych algorytmów przetwarzania obrazów zaliczamy:

- Przekształcenia geometryczne,
- Przekształcenia punktowe (bezkontekstowe)
- Przekształcenia kontekstowe (filtry),
- Przekształcenia widmowe,
- Przekształcenia morfologiczne.

Przekształcenia punktowe transformują każdy punkt obrazu w taki sam sposób, bez względu na otoczenie.

Przekształcenia kontekstowe, czyli np. filtry konwolucyjne, medianowe uzależniają wynik od otoczenia rozważanego elementu rastra, ale wykonywane są zawsze (nawet gdy wartość w danym punkcie nie ulega zmianie).

Morfologiczne przekształcenia z kolei przekształcają tylko tę część punktów, których otoczenie jest zgodne z elementem strukturalnym co pozwala na planowanie przekształceń.

Przekształcenia widmowe polegają na modyfikacji dwuwymiarowego widma Fouriera obrazu (na przykład filtracja dolnoprzepustowa).

Przekształcenia geometryczne to przesunięcia, odbicia, obroty, przycinanie, czyli przekształcenia transformujące obraz.

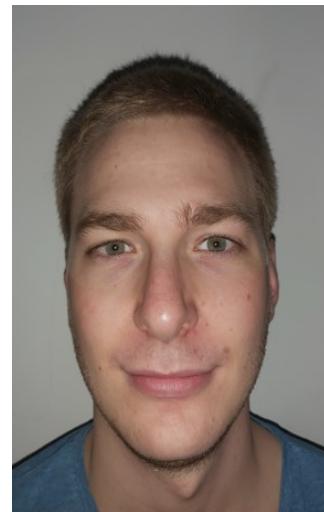
3. Przekształcenia geometryczne

Przekształcenia geometryczne służą do przygotowania obrazu dla bardziej złożonych operacji. Najczęściej wykorzystywane są do korekcji błędów geometrii.

Pierwszą operacją w sekcji przekształceń geometrycznych jest skalowanie. Wykorzystywana jest funkcja *imresize(obraz, skala)*. Oprócz skali można podać także rozmiar docelowy obrazu w postaci [wysokość, szerokość].



Rysunek 3.1. Obraz przed skalowaniem.

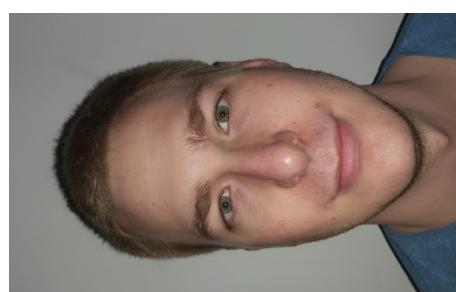


Rysunek 3.2. Obraz po skalowaniu. Skala = 0.6.

Kolejną operacją jest funkcja służąca do obracania obrazu: *imrotate(obraz, stopnie, 'metoda_interpolacji', 'przycinanie')*. Metoda interpolacji pozwala określić wielkość otoczenia służącego do obliczania wynikowych pikseli. Przyciananie odpowiada za wyjściowy rozmiar zdjęcia, można przyciąć obraz do nowych rozmiarów (po obrocie) lub zachować bazowe wymiary.

Podobną funkcją jest *imcrop(obraz, [x0, y0, szerokość, wysokość])*, która pozwala na przycinanie obrazu do obszaru określonego przez piksel początkowy (x_0, y_0) i zaczepionego w nim prostokąta o rozmiarach (szerokość, wysokość). Funkcja pozwala także wyciąć obszar obrazu wraz z mapą kolorów.

Ostatnią funkcją jest *imtransform(obraz, transformacja, 'metoda_interpolacji', ...)*. Podane parametry wejściowe są podstawowymi, natomiast w dokumentacji jest opisana szeroka liczba argumentów dodatkowych. Transformację można utworzyć za pomocą funkcji *maketfrom* oraz *cp2tform*. Funkcje te oraz ich duża liczba możliwości opisane są w dokumentacji Matlaba.



Rysunek 3.3. Obraz po obróceniu o 90 stopni.



Rysunek 3.4. Obraz po przycięciu funkcją imcrop.



Rysunek 3.5. Obraz po transformacji funkcją imtransform.

4. Przekształcenia punktowe

Przekształcenia punktowe to:

- Przekształcenia oparte na przetwarzaniu pojedynczych punktów,
- Przekształcenia oparte na arytmetycznych operacjach na pojedynczych punktach,
- Przekształcenia z użyciem LUT (Look Up Tables),
- Wyrównywanie histogramu,
- Operacje wykonywane na dwóch obrazach,
- Binaryzacja obrazu.

Takie operacje nie wprowadzają zmian geometrii obrazu. W przypadku istnienia funkcji ścisłe monotonicznej zawsze istnieje operacja odwrotna, pozwalająca odzyskać obraz pierwotny. Ważna cechą tych operacji jest fakt, iż nie wprowadzają one żadnej informacji do obrazu. Operacje te mogą pomimo prostego przekształcenia znacząco zmienić postrzeganie obrazu.

Przykładowe operacje punktowe:

- *imadd(obraz, stała_lub_obraz)* – funkcja dodająca do każdego piksela stałą wartość lub wartość piksela z obrazu o takim samym rozmiarze, co spowoduje dodanie do siebie wartości odpowiednich pikseli.
- *immultiply(obraz, stała_lub_obraz)* – podobnie jak powyżej, z tym że zamiast dodawania piksele są mnożone.
- *imabsdiff(obraz1, obraz2)* – funkcja służąca do obliczania modułu różnicy obrazu2 i obrazu1,
- *imadjust(obraz, [pocz_wej kon_wej], [pocz_wyj kon_wyj])* – funkcja służąca do zwiększenia kontrastu obrazu za pomocą rozciągnięcia histogramu z zakresu początkowego do

końcowego. Zakres przed i po operacji określa się wartościami z zakresu od 0 do 1. Dla obrazów kolorowych można podać osobne zakresy dla każdego kanału. Przekształcenie to jest nazywane modulacją (korekcją) gamma i służy do usuwanie zniekształceń wprowadzonych przez urządzenie jak np. monitor,

- *histeq(obraz)* - funkcja wyrównująca histogram tak aby ilość punktów o poszczególnych poziomach jasności w każdym z przedziałów histogramu była równa (w przybliżeniu),
- *rgb2gray(obraz)* – funkcja zamieniająca obraz kolorowy na obraz w skali szarości według wag kolorów określonych przez normę, taki obraz pozwala na łatwiejsze przetwarzanie,
- *im2bw(obraz, poziom)* lub *imbinarize(obraz, 'metoda')* / *imbinarize(obraz, poziom)* – funkcje pozwalające zamienić obraz w skali szarości (lub kolorowej, ale zostanie on zamieniony na skale szarości) na obraz binarny, dla którego piksele przyjmą wartość 0 lub 1 w zależności od funkcji binaryzacji (dolny, górny próg lub podwójne ograniczenie),



Rysunek 4.1. Obraz po dodaniu stałej wartości (50) do każdego piksela.

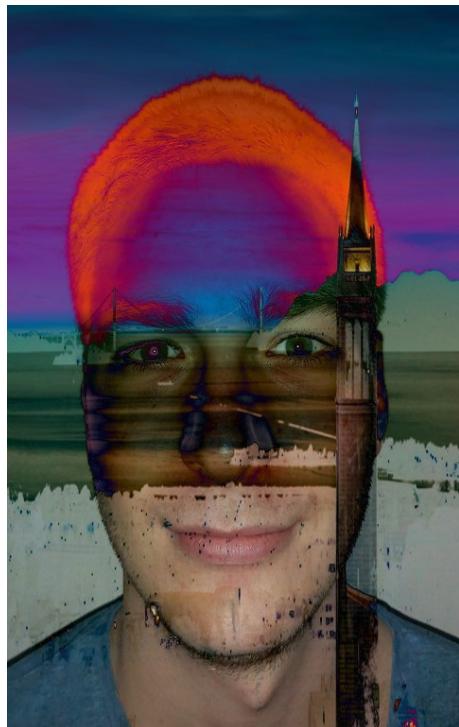


Rysunek 4.2. Obraz po odjęciu stałej wartości (-50) od każdego piksela.

Jak widać za pomocą funkcji *imadd* można rozjaśnić oraz przyciemnić obraz. Funkcja *multiply* również może rozjaśnić lub przyciemnić obraz, jednak jej wpływ jej zmian jest zależny od poziomu jasności piksela przed operacją. Funkcja *imabsdiff* przedstawiona została na rysunku 4.4. W przykładzie od przetwarzanego zdjęcia odjęto losowe zdjęcie z Internetu. W tym celu należało przeskalać je do tych samych rozmiarów a po operacji odjęcia użycia została korekcja kolorów *imadjust*. Podobnie wykonano sumę obrazów, jednak nie poddano wynikowego obrazu korekcji gamma. Na rysunku 4.6 widoczny jest obraz oryginalny po wyrównaniu histogramu.



Rysunek 4.3. Obraz po przemnożeniu przez stałą wartość (1,25).



Rysunek 4.4. Obraz wynikowy z odjęcia od obrazu losowego obrazu z Internetu.

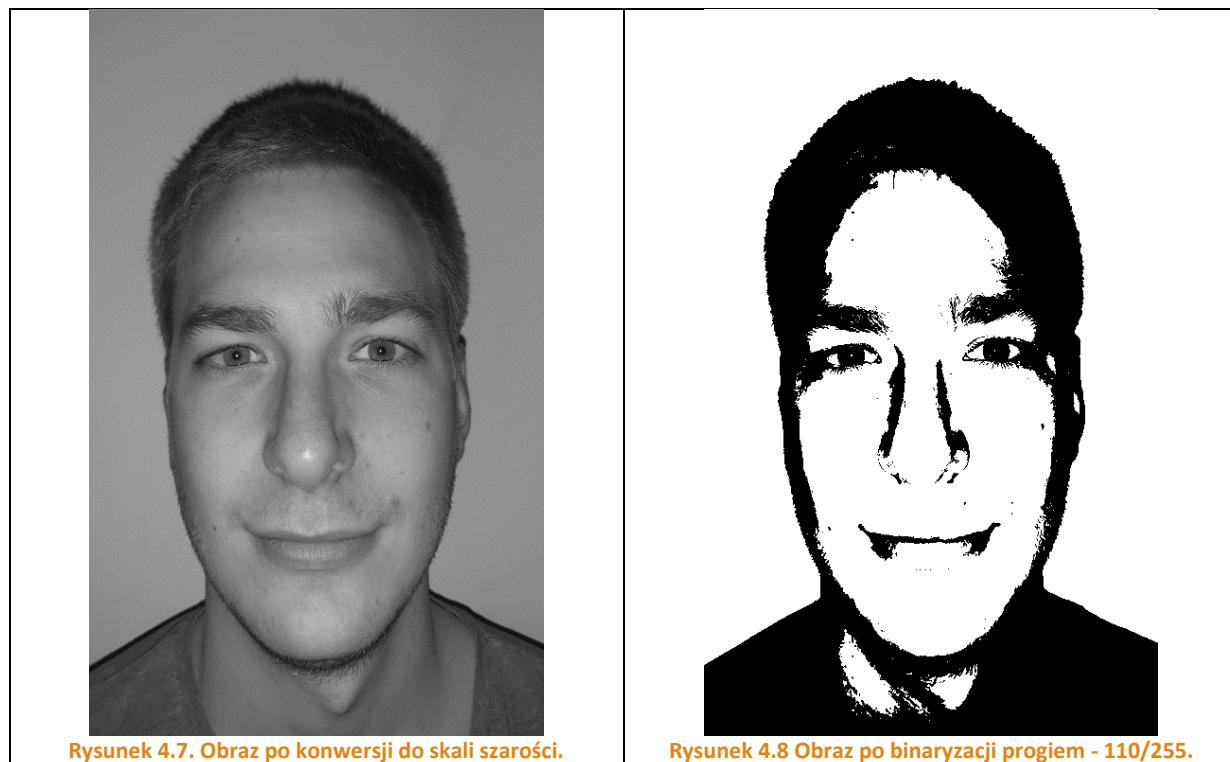


Rysunek 4.5. Operacja dodawania obrazów. Wykorzystano obraz z poprzedniego przykładu.



Rysunek 4.6. Obraz orginalny po zastosowaniu wyrównania histogramu.

Za pomocą funkcji `rgb2gray` dokonano konwersji obrazu na skalę szarości i poddano go korekcji gamma dla $\gamma=1$. Następnie dokonano przykładowej binaryzacji z progiem przyjętym na poziomie 110. Oba efekty przetwarzania widoczne są poniżej.



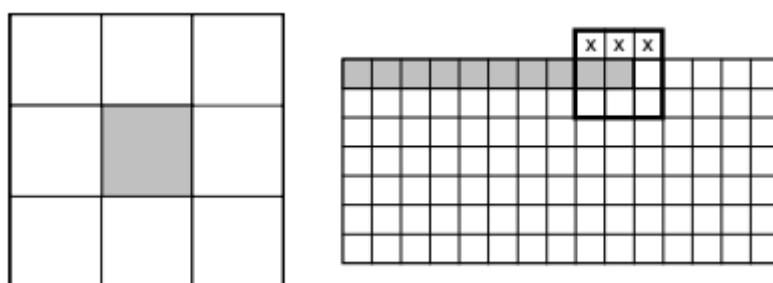
Rysunek 4.7. Obraz po konwersji do skali szarości.

Rysunek 4.8 Obraz po binaryzacji progiem - 110/255.

Jak widać, operacje geometryczne oraz punktowe stanowią pewną bazę operacji przetwarzania obrazów, która pozwala na przygotowanie obrazu do dalszej obróbki.

5. Kontekstowa filtracja obrazów

Wspomniana i opisana wcześniej filtracja operuje na maskach, które mogą przyjmować różne kształty jednak najpopularniejszą jest maska kwadratowa o wielkości NxN, gdzie N to liczba elementów, z reguły N jest nieparzyste a analizowany punkt znajduje się w środku maski. Maska ta jednak nie może dotyczyć pikseli na brzegach, gdyż nie występuje tam pełne otoczenie.



Rysunek 5.1. Przykład maski i jej zachowania na brzegu.

Filtrowanie obrazów jest wykorzystywane do tłumienia szumów, wzmacniania elementów zgodnych z wzorcem, usuwaniem wad, poprawą obrazów słabej jakości, rekonstrukcją obrazów uszkodzonych. Filtracja jest prosta i intuicyjna w implementacji. Wyróżnia się filtry liniowe (liniowa kombinacja pikseli) oraz nieliniiowe. Filtry liniowe, ze względu na swoje cechy addytywności i jednorodności można opisać splotem funkcji (konwolucja). Splot funkcji obrazu z funkcją filtrującą, którą możemy dowolnie zaprojektować. Splot funkcji filtrującej z deltą Diraca (Kroneckera) jest jej odpowiedzią impulsową, która jest związana z macierzą filtru.

Do filtracji obrazów można wykorzystać funkcję *imfilter(obraz, filtr)*, która przyjmuje jako argumenty obraz oraz filtr będący macierzą NxN o znormalizowanych wartościach. W tabeli pokazany jest filtr bez normalizacji.

Przykładowo filtr dolnoprzepustowy powoduje wygładzenie zawirowań, usuwa zakłócenia (rozmycie) czy usuwanie falowania. Wadą jest rozmycie konturów i krawędzi co powoduje osłabienie możliwości wykrycia kształtu. Przykładowy filtr dolnoprzepustowy pokazany jest poniżej, wraz z wynikiem filtracji obrazu tym filtrem.

Tab. 5.1. Macierz filtra dolnoprzepustowego.

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1



Rysunek 5.2. Obraz powstały w wyniku filtracji filtrem dolnoprzepustowym przedstawionym obok.

Dodatkowo zaprezentowany jest wynik filtracji o wagach dobranych subiektywnie. Normalizacja tak tworzonego filtra polega na podzieleniu jego elementów przez sumę wszystkich elementów. Jak można zauważyć filtr zdefiniowany w tabeli 5.2. nie powoduje takiego rozmycia ze względu na dużą wagę środkowego elementu, widoczne jest jednak pewne rozmycie (okolice włosów).

Tab. 5.2. Macierz własnego filtru.

2	4	2
4	8	4
2	4	2

**Rysunek 5.3. Wynik filtracji własnym filtrem.**

Oprócz filtrów dolnoprzepustowych znane są także filtry górnoprzepustowe, czyli gradienty. Wydobywają one z obrazów szybkie zmiany (np. krawędzie). Krawędź jest to linia, która może być opisana jako skok jednostkowy, który jest całką z delty Diraca. Filtry górnoprzepustowe zawsze realizują formę różniczkowaną sygnału. Z filtrów górnoprzepustowych wyróżnia się gradient Robertsa, maski Prewitta, maski Sobela. Gradienty te są kierunkowe i mogą być obracane, jednak podkreślają one zawsze pewien kierunek. Przykłady przedstawione są poniżej.

Tab. 5.3. Gradient Robertsa w masce 3x3.

0	0	0
-1	0	0
0	1	0

**Rysunek 5.4. Wynik filtracji gradientem Robertsa.**

Tab. 5.4. Gradient Robertsa w masce 3x3, odwrócony.

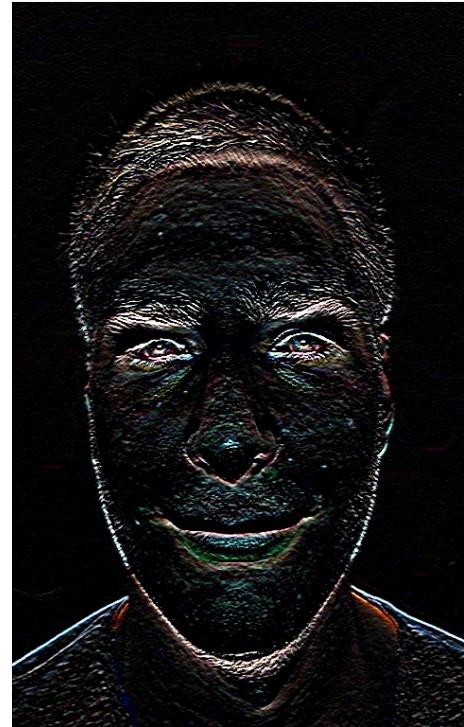
0	0	0
0	0	1
0	-1	0

**Rysunek 5.5. Wynik filtracji odwróconym gradientem Robertsa.**

Widoczna jest kierunkowość gradientu (usta).

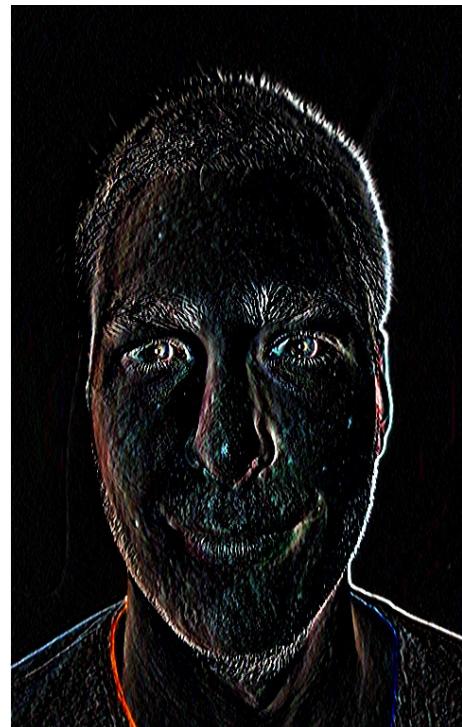
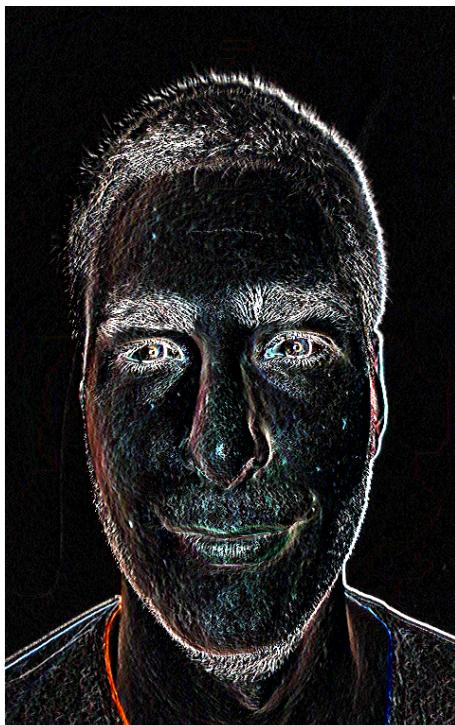
Tab. 5.5. Maska Prewitta.

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

**Rysunek 5.6 Wynik filtracji maską Prewitta.**

Tab. 5.6. Maska Prewitta, odwrócona.

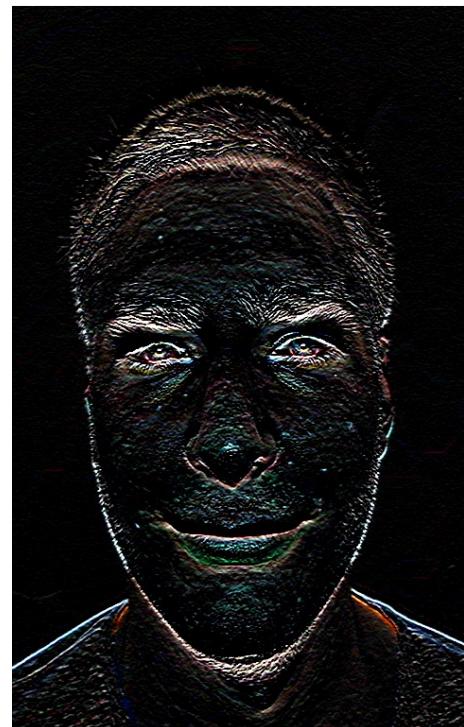
-1	0	1
-1	0	1
-1	0	1

**Rysunek 5.7. Wynik filtracji maską Prewitta, odwróconą.****Rysunek 5.8. Suma gradientów Robertsa.****Rysunek 5.9. Suma filtracji maskami Prewitta.**

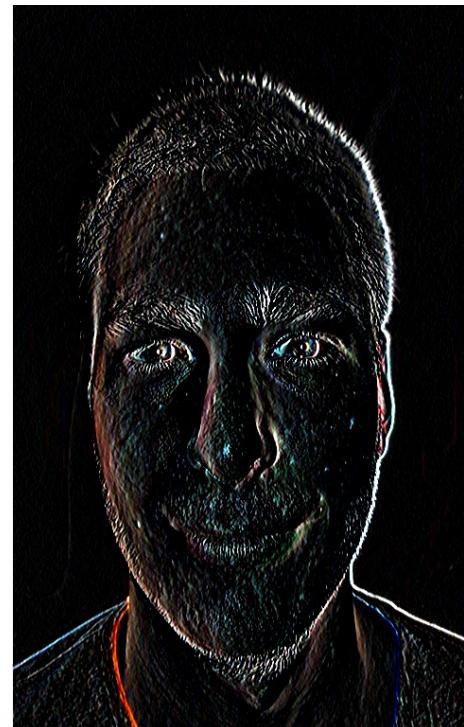
Sumowanie filtracji dla różnych kierunków pozwala uzyskać pełniejszy obraz krawędzi i konturów. Dostępne są także maski wykrywające narożniki a ich współczynniki są analogiczne do powyższych masek tylko że symetria występuje względem diagonali.

Tab. 5.7. Maska Sobela.

-1	-2	-1
0	0	0
1	2	1

**Rysunek 5.10. Wynik filtracji maską Sobela.****Tab. 5.8. Maska Sobela, odwrócona.**

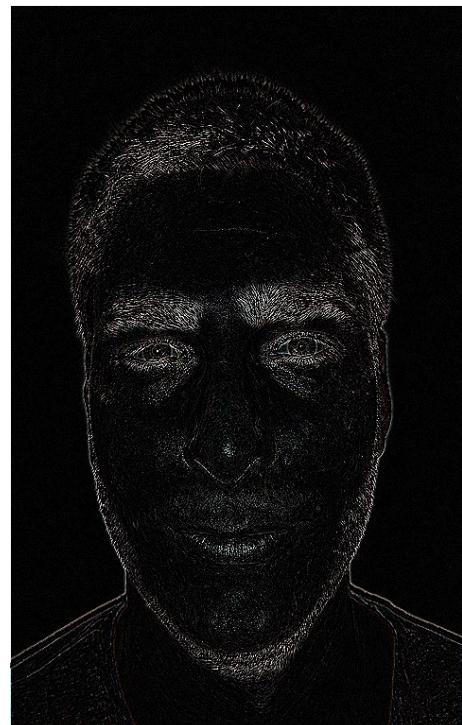
-1	0	1
-2	0	2
-1	0	1

**Rysunek 5.11. Wynik filtracji maską Sobela, odwrócona**

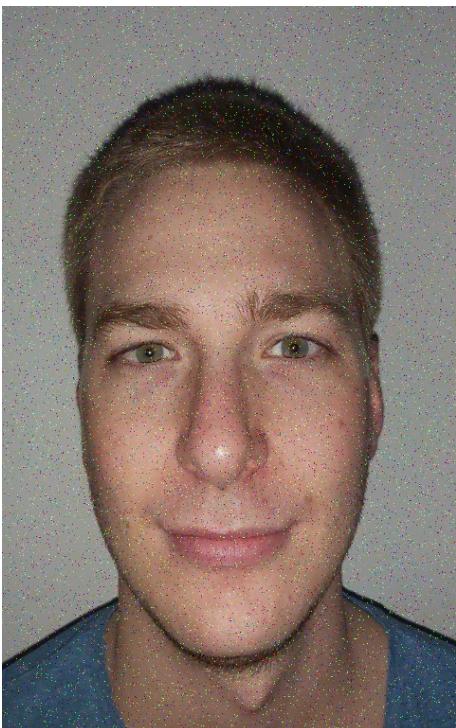
Oprócz gradientów kierunkowych istnieją także bezkierunkowe, czyli laplasjany. W wielu programach dostępne są specjalne filtry, podobnie w Matlabie funkcja *fspecial('filtr')* pozwala na stworzenie wielu filtrów o różnej budowie (prostokątne, kołowe). W dokumentacji dostępne są różne opcje.

Tab. 5.9. Maska Laplasjanu.

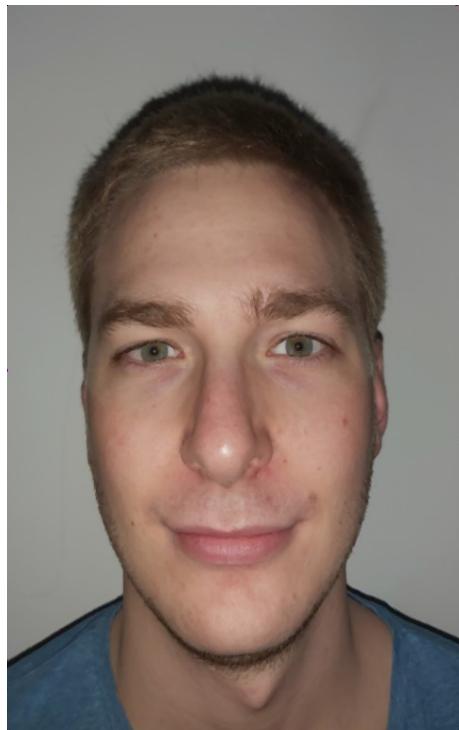
0	-1	0
-1	4	-1
0	-1	0

**Rysunek 5.12. Wynik filtracji filtrem specjalnym 'average'. Rysunek 5.13. Wynik filtracji filtrem specjalnym 'unsharp'.**

Filtryle medianowe służą do usuwania szumu (punktów różniących się od otoczenia). Przykładowo pokazana jest filtracja każdego kanału koloru zdjęcia RGB filtrem medianowym – `medfilt2(kolor_obrazu, [N N])`. Oryginalny obraz zaszumiony został za pomocą funkcji `imnoise(obraz, 'metoda', stopień_zaszumienia)`, w przykładzie zastosowano szum 'salt & pepper'.



Rysunek 5.14. Obraz zaszumiony.



Rysunek 5.15. Obraz po filtracji medianowej każdego koloru.

Filtryle nie liniowe powstają w wyniku zastosowania operacji logicznych na obrazach poddanych wcześniejszej filtracji. Przykładem mogą być zsumowane obrazów filtracji pod różnymi kierunkami. Operacje logiczne najczęściej stosowane to:

- NOT – zaprzeczenie,
- AND – iloczyn logiczny,
- OR – suma logiczna,
- SUB – różnica logiczna,
- XOR – suma rozłączna,
- NXOR – równoważność logiczna.

6. Transformacje obrazów

Obraz jako funkcja dwóch zmiennych może zostać poddany transformacją znanym funkcją jednej zmiennej. Przykładem może być transformata Fouriera czy kosinusowa. W przypadku transformacji Fouriera polega ona na poszerzeniu definicji z 1 wymiaru o kolejny. Tak więc zamiast jednego sumatora mamy dwa dla każdego z kierunków. W przypadku cyfrowych obrazów posługujemy się dyskretną transformatą, a szczególnej jej szybkiej implementacji FFT. W Matlabie dostępna jest funkcja `fft2(obraz)` pozwalająca na obliczenie dyskretnej transformaty dla 2 wymiarowych sygnałów, a funkcja `fftshift(wynik_fft2)` przestawia odpowiednio ćwiartki widma.



Rysunek 6.1. Widmo obrazu uzyskane za pomocą FFT2.

Transformacja Fouriera pozwala na filtrację obrazów, gdzie zerowanie dużej liczby amplitud widma, powoduje znaczne zmniejszenie rozmiarów przy zachowaniu rozpoznawalności obrazu.

7. Przekształcenia morfologiczne

Przekształcenia morfologiczne służą do analizy kształtów elementów obrazu, ich wzajemnego położenia. Są one podstawą budowania złożonych algorytmów analizy. Wykorzystują one pojęcie elementu strukturalnego, czyli pewnego wycinka obrazu z wyróżnionym punktem centralnym. Przykładem może być koło czy linia. Operacje te najczęściej wykonuje się na obraz binarnych.

Algorytm przekształcenia:

- Przemieszczanie elementu po całym obrazie, dla każdego punktu wykonywana jest analiza koincydencji punktów obrazu i elementu,
- W każdym punkcie obrazu następuje sprawdzenie czy rzeczywista konfiguracja pikseli obrazu w otoczeniu tego punktu jest zgoda ze wzorcem,
- W przypadku zgody wzorca i pikseli obrazu następuje wykonanie ustalonej operacji na tym punkcie.

Erozja – polega na przypisaniu analizowanemu punktowi wartość minimalną jego sąsiadów, gdzie obszar jest określony przez element strukturalny, erozja jest addytywna (złożenie erozji o mniejszych promieniach), położenie punktu centralnego nie ma wielkiego znaczenia, elementy liniowe uwypuklają linie o tym samym kierunku, usuwa drobne odizolowane obszary obrazu.

Obraz został poddany binaryzacji po zmianie skali na skale szarości. Próg binaryzacji przyjęto podobnie jak w rozdziale 4 na poziomie 110/255 a wynik jest widoczny na rysunku 4.8.

Przykładowe wyniki zastosowania erozji różnymi elementami strukturalnymi. Jak widać element liniowy i dyskowy. Widoczne jest podkreślenie kierunku elementu liniowego oraz charakter elementu dyskowego (pojedyncze czarne punkty).

Dylatacja – przekształcenie odwrotne do erozji, zdefiniować można jako negatyw obrazu erozji. Jest to filtr maksymalny, który przypisuje elementowi centralnemu wartość maksymalną z otoczenia, zdefiniowanego elementem strukturalnym. Dylatacja usuwa drobne wkleścia, jednak zmniejsza zostaje długość brzegów, a zwiększone zostają powierzchnie, prowadzi więc ona do zrastania elementów.

W przykładzie zastosowano dylatację elementem dyskowym o promieniu 2 oraz liniowym o rozmiarze 3x15. Jak można zauważyc element liniowy o takiej orientacji nie powoduje znaczących zmian względem rysunku 4.8, jednak usunięte są małe punkty widoczne na policzkach i nad ustami (rysunek 7.2).

Można zauważyć, że element dyskowy powoduje pozostawienie charakterystycznych kropek w źrenicach. Może to posłużyć obliczeniu odległości źrenic.

Funkcja `strel('arbitrary', maska)` pozwala na zdefiniowanie własnej maski elementu strukturalnego użytego do operacji morfologicznych. Na rysunku 7.5 widoczna jest operacja dylatacji elementem kwadratowym 4x4.



Rysunek 7.1. Wynik erozji elementem liniowym o rozmiarach 15x1.



Rysunek 7.2. Wynik erozji elementem dyskowym o promieniu 2.



Rysunek 7.3. Wynik dylatacji elementem 3x15.



Rysunek 7.4. Wynik dylatacji elementem dyskowym o promieniu 2.



Rysunek 7.5. Dylatacja elementem kwadratowym o boku 4.

Jak można zauważyć erozja zmniejsza pole powierzchni a dylatacja zwiększa. Ich połączenie pozwala eliminować tą wadę. W zależności od kolejności wyróżnia się:

- Otwarcie – erozja i dylatacja – usuwa drobne obiekty i szczegóły, nie zmienia zasadniczej części figury, może usunąć przewężenia łączące elementy,
- Zamknięcie – dylatacja i erozja – wyplenia wcięcia, drobne otwory, łączy elementy blisko leżące.

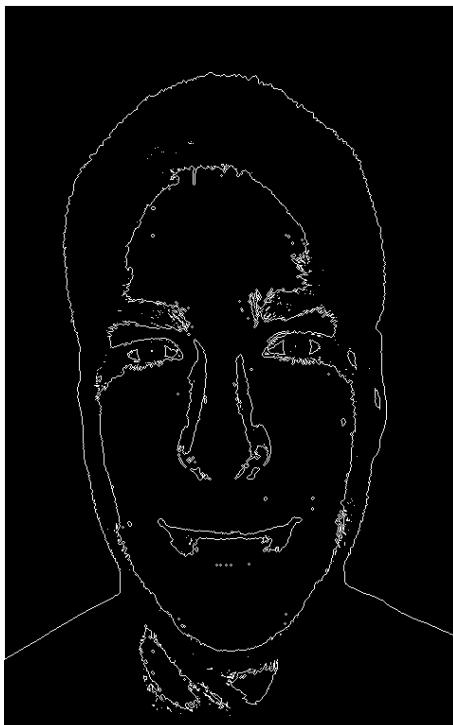


Rysunek 7.6. Zamknięcie obrazu elementem dyskowym o promieniu 2.

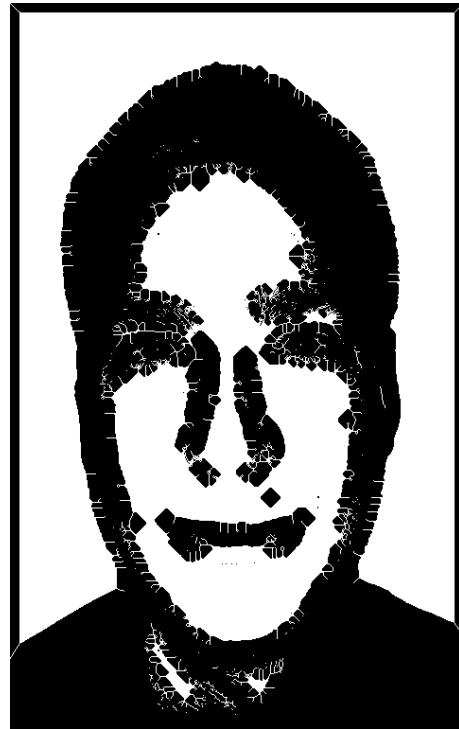


Rysunek 7.7. Otwarcie obrazu elementem dyskowym o promieniu 2.

W Image Processing Toolbox dostępna jest funkcja *bwmorph()* w której zaimplementowane zostało wiele opcji operacji morfologicznych. Na przykład opcja ‘remove’ powoduje usunięcie wnętrza figur, pozostawiając tylko kontury, z kolei ‘skel’ powoduje szkieletyzację obrazu.



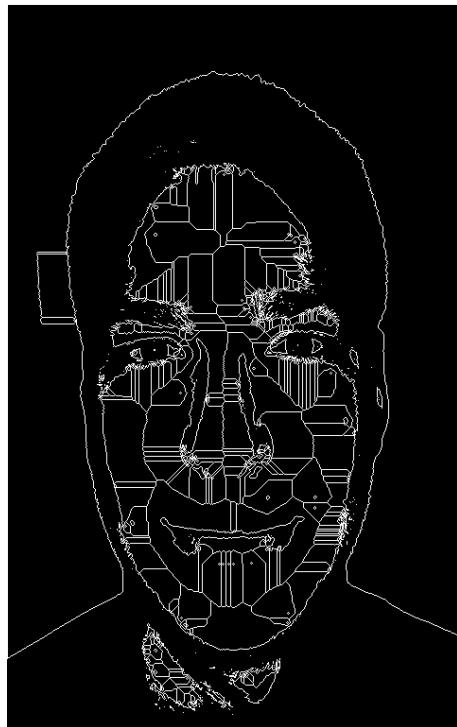
Rysunek 7.8. Obraz powstały po pozostawieniu krawędzi.



Rysunek 7.9. Obraz powstały w wyniku szkieletyzacji 10 krotniej.

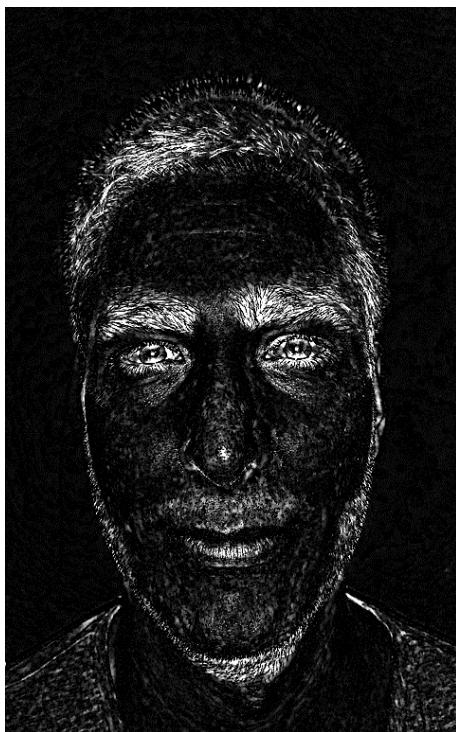
Kolejna operacją morfologiczną jest wyznaczanie centroidów. Polega to na uzyskaniu szkieletu z obciętymi gałęziami. Operacja sprowadza figury do momentu, gdy nie ma ona już otworów lub do

pętli, gdy posiada otwory. Pokazany został przykład wyznaczania centroidów jednak zdjęcie bazowe jest zbyt skomplikowane, aby uzyskać interesujące efekty.

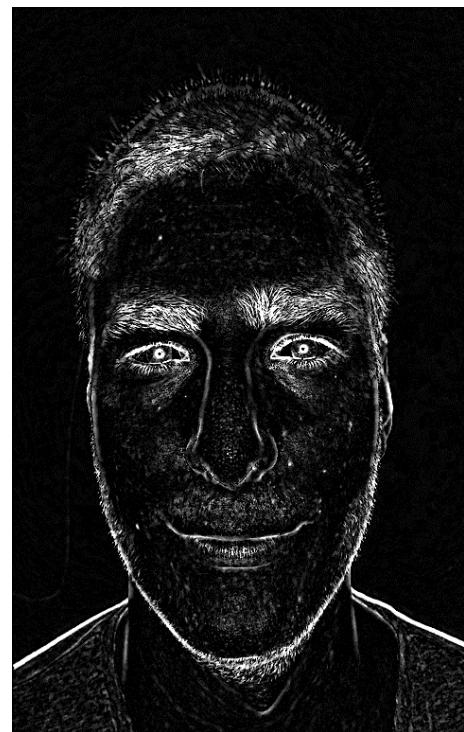


Rysunek 7.10. Wynik wyznaczania centroidów.

Przeprowadzając operacje na obrazie w skali szarości – imbothat i imtophat można wykryć na obrazie minima i maksima. Operacje te polegają na wykonaniu zamknięcia i otarcia a następnie odjęcia wyniku od bazowego obrazu. Do obu przykładów zastosowano element dyskowy o promieniu 4.



Rysunek 7.11. Wynik operacji imbothat.



Rysunek 7.12. Wynik operacji imtophat.

Operacje morfologiczne pozwalają na uzyskanie bardzo ciekawych wyników, wymagają one jednak dobrze przygotowanego zdjęcia binarnego. Przykładowo dla zdjęcia mikrostruktury można wyznaczyć udziały procentowe składników. Przykładowo liczba białych punktów do liczby pikseli określa udział osnowy. Następnie określając za pomocą operacji binaryzacji z 2 progami fazę brązową i wykorzystując operację zalewania otworów funkcją imfill('holes') można określić udział tej fazy, a faza szara pozostaje wynikiem dopełnienia udziałów do 100%.



Rysunek 7.13. Zdjęcie mikrostruktury.

Rysunek 7.14. Obraz po binaryzacji, filtracji medianowej.
Białe tło to osnowa.

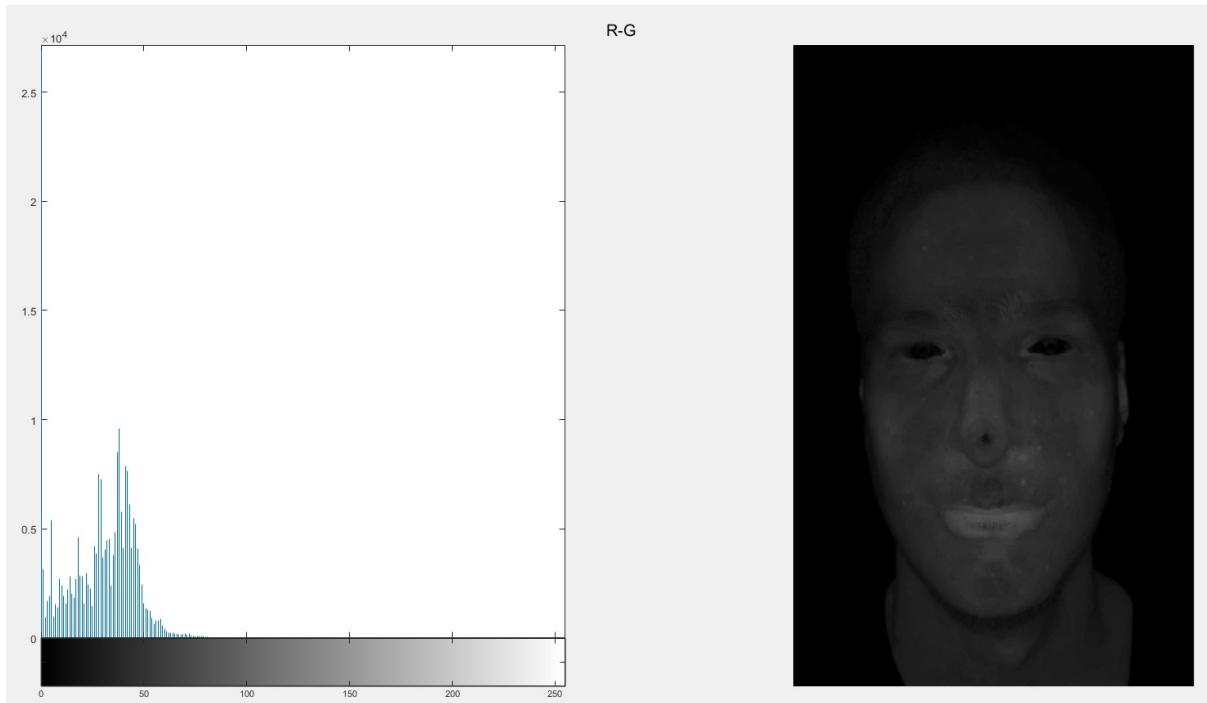
Rysunek 7.15. Udział fazy brązowej (biały kolor) po binaryzacji z 2 progami, filtracji medianowej i zamknięciu otworów.

8. Wyznaczanie cech twarzy na podstawie obrazu

W tym rozdziale opisany zostanie przykład wyznaczania cech twarzy (np. wymiarów geometrycznych) na podstawie zdjęcia i jego przetwarzania. Wykorzystane zostanie zdjęcie z poprzednich rozdziałów. W ramach ćwiczenia przeprowadzono wyznaczanie cech twarzy za pomocą dostępnych metod jak moduł *vision* a dokładniej *vision.CascadeObjectDetector*, w którym dostępne są funkcje wyznaczające obiekty na podstawie algorytmu *Viola-Jones*. Wspomniany moduł posłużył do napisania programu wyznaczającego na żywo odległość oczu w pikselach za pomocą kamery („*Wykrywanie_oczu_kamera.m*”) oraz na podstawie zdjęcia twarzy („*Wykrywanie_oczu.m*”). Kody zostaną załączone na końcu rozdziału.

Drugie podejście do wykrywanie cech twarzy polegało na wykorzystaniu binaryzacji, operacji morfologicznych, podziału obrazu na odpowiednie obszary poszukiwań. Napisany został skrypt, który wykrywa kolejno źrenice oczy, środek chrząstki nosa, usta, skrajnie policzków. Następnie na podstawie wymiaru skalującego (odległość źrenic) obliczane są wymiary pozostałych cech i porównywanie ze zmierzoną rzeczywistą wartością.

Wykrycie usta polega na zastosowaniu różnicy kanałów koloru czerwonego i zielonego. Dla takiego podejścia usta (czerwony kolor) zyskują znaczny odstęp od koloru skóry (kanał zielony występuje w kolorze skóry, ale nie w ustach). Widoczny histogram pokazuje rozkład poziomów jasności.

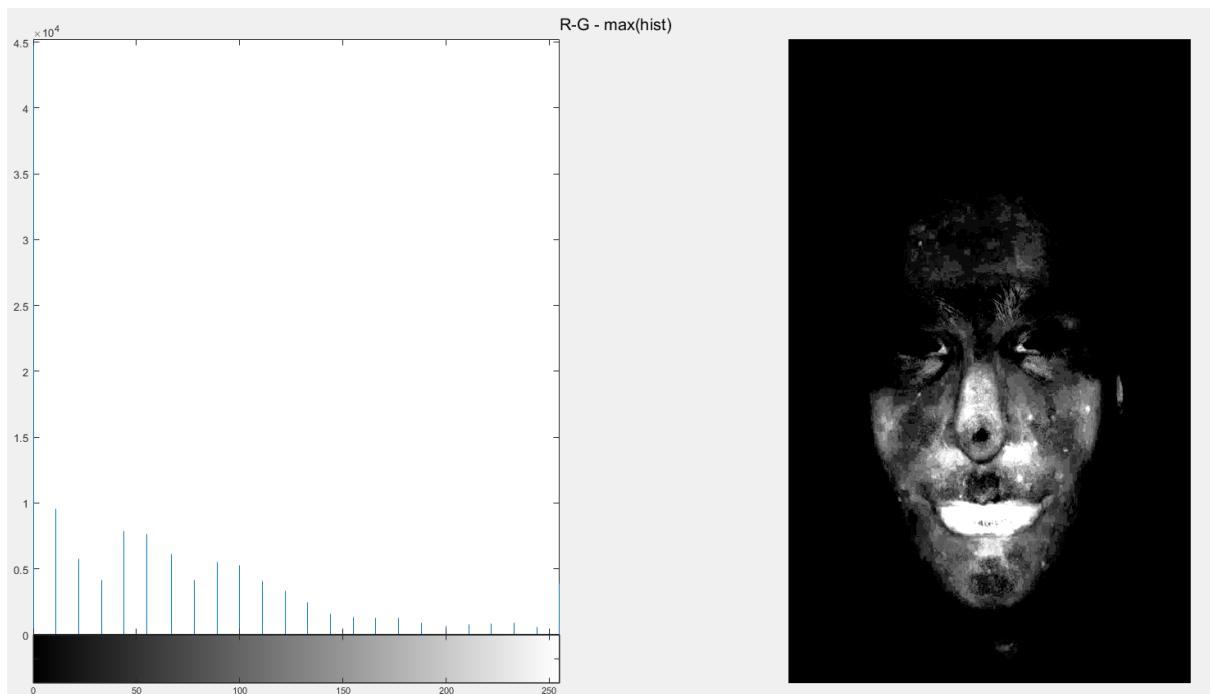


Rysunek 8.1. Histogram i obraz dla kanały R-G.

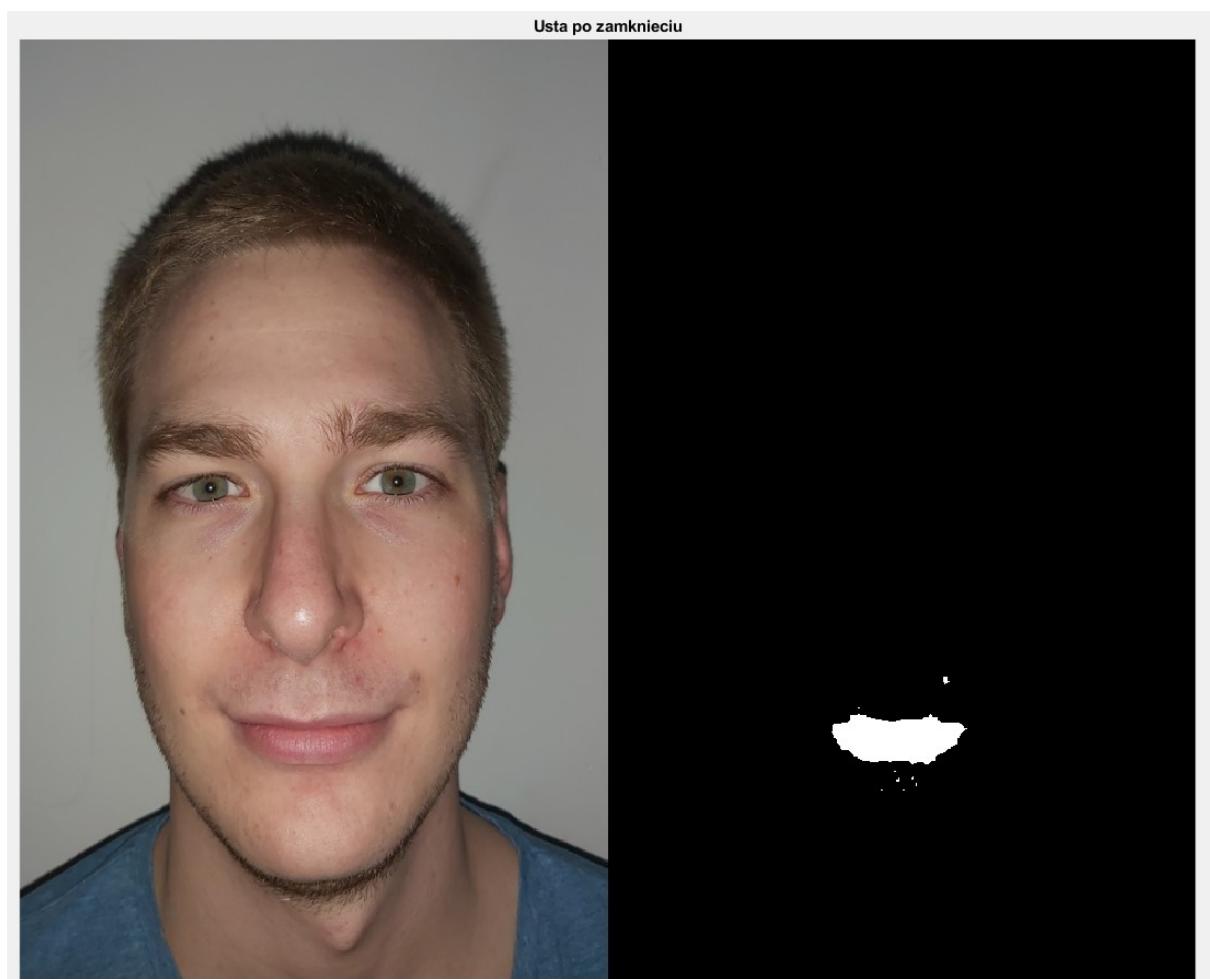
Jak widać histogram pokazuje ze poziomy nie są dobrze wykorzystane. Poza kolorem czarnym drugim najczęściej występującym poziomem jest średni poziom jasności skóry. Odejmując od obrazu wartość poziomu średniego otrzymujemy lepszy odstęp poziomu jasności ust od tła. Następnie do pełnego wykorzystania poziomów jasności zastosowano funkcję *imadjust*. Wynik przedstawiony jest na rysunki 8.2.

Kolejnym etapem jest pozostawianie około 10% najjaśniejszych pikseli, jednak zastosowano podejście binaryzacji z progiem 244/255 aby pominąć wyznaczanie progu na podstawie poziomów z histogramu. Dla obrazu binarnego wyzerowane zostały piksele poza regionem dolnym zdjęcia. Dokonano operacji morfologicznych w celu wyczyszczenia pojedynczych pikseli o odmiennej jasności od otoczenia, wypełnienie otworów. Następnie dla rosnącego elementu dyskowego dokonano zamknięcia obrazu, aby wydobyć obszar ust z wypełnieniem. Ostatecznie dokonano otwarcia elementem dyskowym, aby pozbyć się regionów odłączonych od ust.

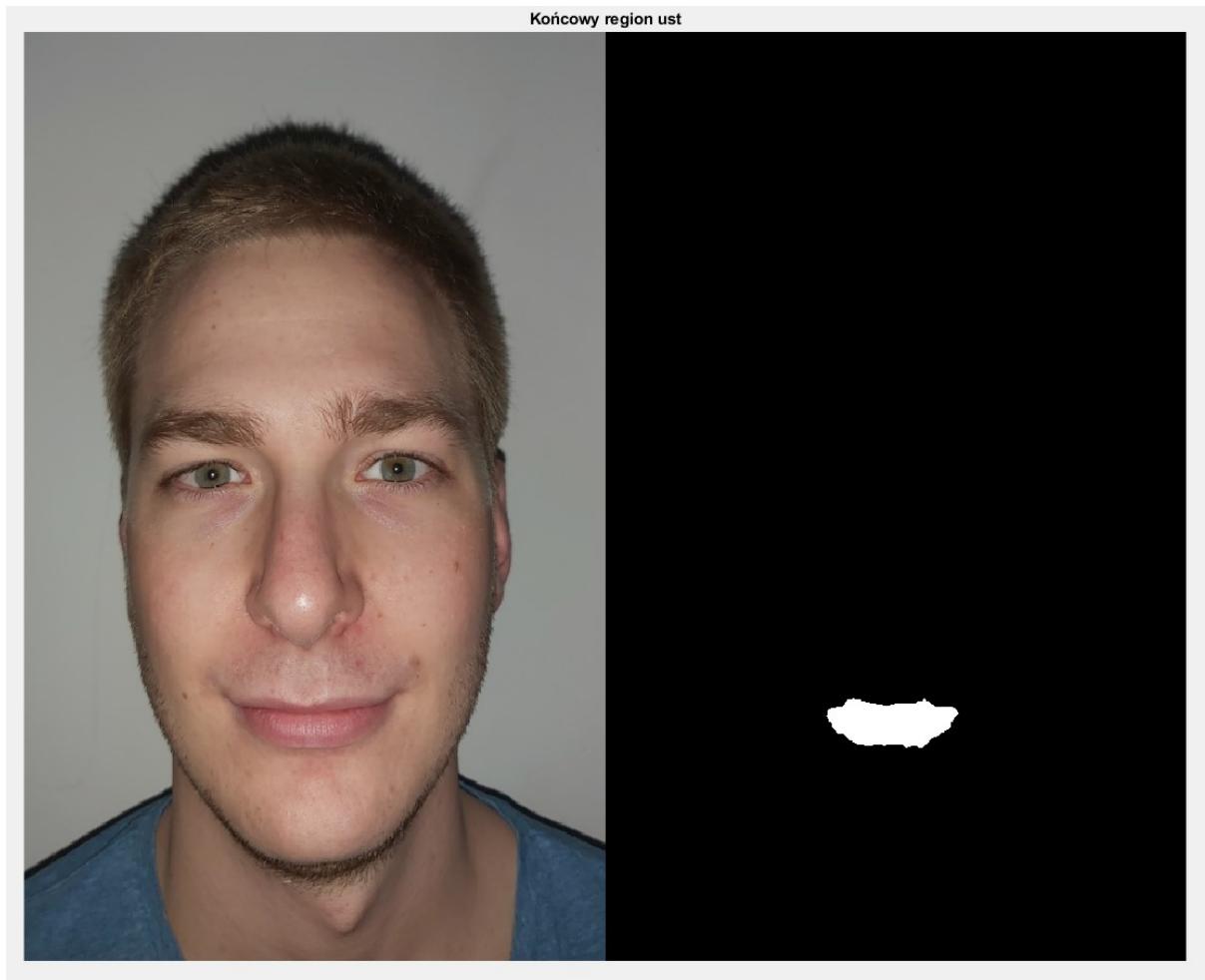
Za pomocą funkcji *regionprops* wyznaczono wszystkie parametry obszaru ust.



Rysunek 8.2. Obraz wynikowy dla operacji R-G z odjętym średnim poziomem jasności skóry.



Rysunek 8.3. Usta po operacjach zamknięcia na obrazie binarnym.



Rysunek 8.4. Końcowy obszar ust po otwarciu.

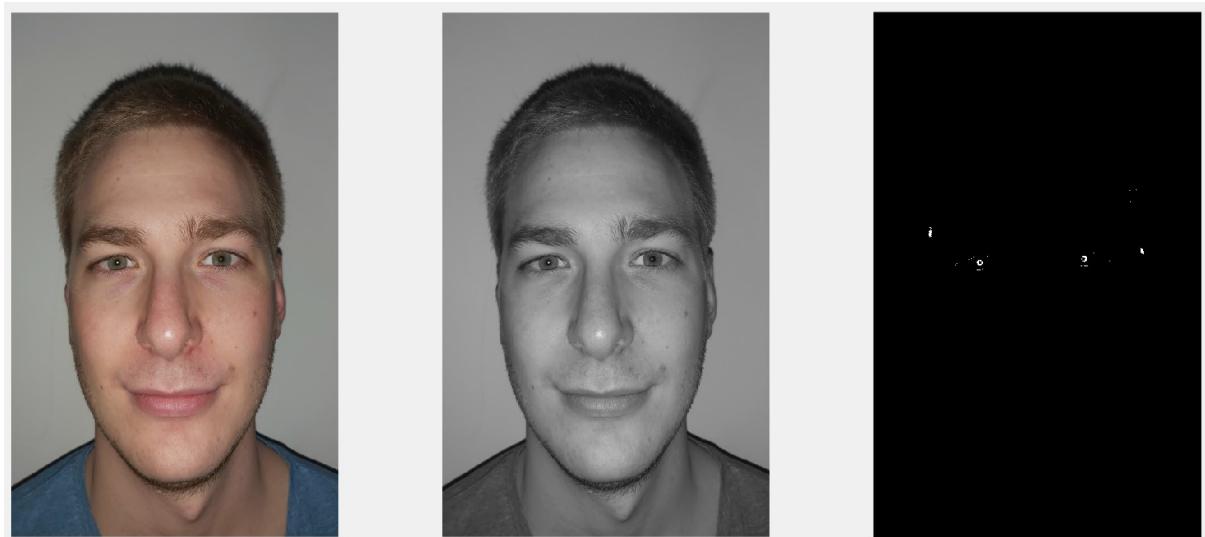
Tab. 8.1. Przykładowe podstawowe parametry obszaru ust.

Area: 3153
Centroid: [236.9026 571.4421]
BoundingBox: [183.5000 551.5000 109 41]
SubarrayIdx: {[1×41 double] [1×109 double]}
MajorAxisLength: 108.4368
MinorAxisLength: 38.4645
Eccentricity: 0.9350
Orientation: -0.4513

Kolejnym etapem jest wykrywanie oczu. Wykorzystano podejście binaryzacji obrazu w skali szarości z dolnym progiem (w matlabie to binaryzacja z górnym progiem i wykorzystanie funkcji *imcomplement*). Wyzerowane zostały regiony poza obszarem, w którym znajdują się oczy.

Następnie dokonano ponownie jak poprzednio morfologicznych operacji czyszczenia i zamykania otworów (refleks w źrenicach). W związku z dyskowym kształtem źrenic zastosowana operacja otwarcia z elementem dyskowym, aby pozbyć się elementów nie połączonych z źrenicami.

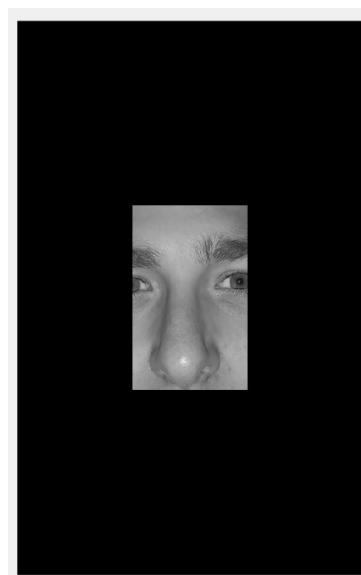
Podobnie jak wcześniej wykorzystano funkcję zwracającą parametry regionów. W tym przypadku zwrócona została struktura zawierająca 2 osobne regiony.



Rysunek 8.5. Obraz orginalny, skala szarości i binaryzacja z dolnym progiem.



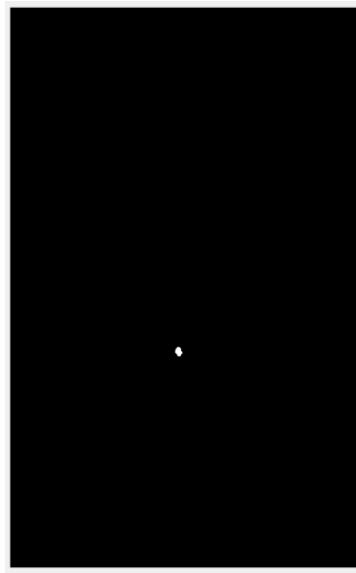
Rysunek 8.6. Regiony oczu po operacjach morfologicznych.



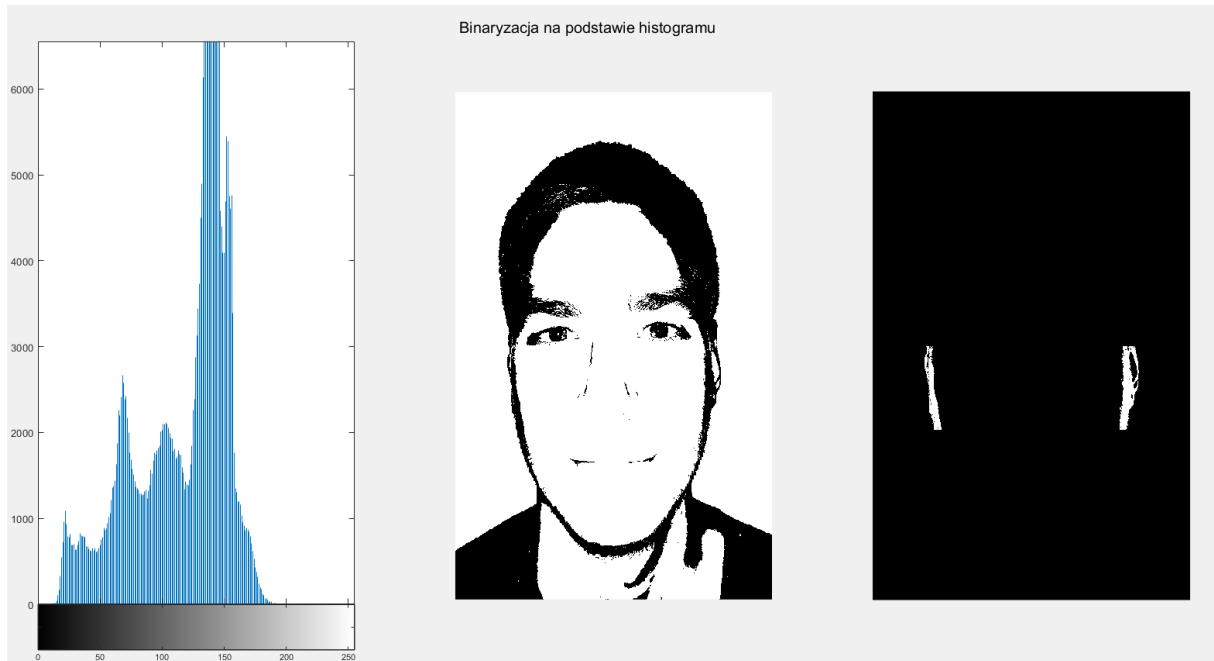
Rysunek 8.7. Obszar wyszukiwana nosa w skali szarości.

Następnie wyznaczony został obszar środka chrząstki nosa. Wykorzystany został refleks światłą na nosie. Wyzerowane zostały obszary poza centralnym obszarem zdjęcia. Dokonana została binaryzacja oraz podstawowe operacje morfologiczne do usuwania pojedynczych pikseli i zalewania otworów. Następnie dokonano operacji zamknięcia i otwarcia elementem dyskowym co pozwoliło na uzyskanie jednego regionu. Dla tak wyznaczonego obszaru wyznaczono jego parametry.

Ostatnimi obszarami twarzy są skrajnie policzków. Wykorzystano znowu podejście binaryzacji obrazu w skali szarości jednak wykorzystano fakt, że skrajnie policzków są ciemniejsze niż część twarzy prostopadła do kierunku robienia zdjęcia. Wyzerowane zostały obszary poza skrajniami środka obrazu.



Rysunek 8.8. Obszar środka chrząstki nosa.



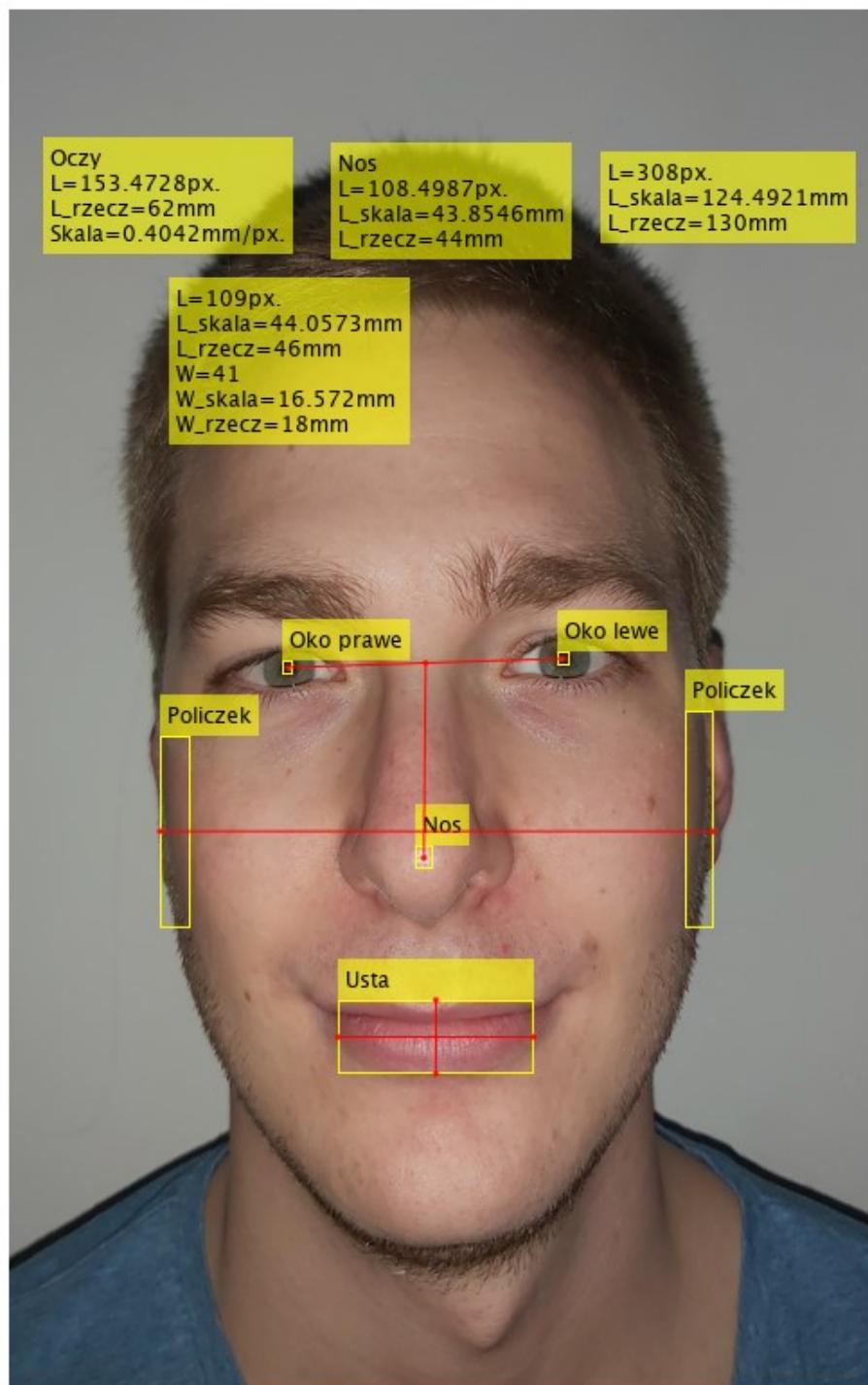
Rysunek 8.9. Wyznaczanie obszarów skrajni policzków.

Standardowo obraz binarny poddano wypełnianiu otworów oraz czyszczeniu z pikseli niezwiązanych z obszarami policzków. Dokonana została operacja morfologiczna otwarcia elementem linowym pionowym, następnie elementem dyskowym i znowu liniowym. Dla takiego obrazu wyznaczono parametry obszarów policzków. Można by było zastosować szkieletyzację aby znaleźć dokładniejszą linię skrajni, jednak ze względu na zarost nie pokrywa się ona z faktyczną skrajnią.



Rysunek 8.10. Obraz oryginalny oraz obszary skrajni policzków.

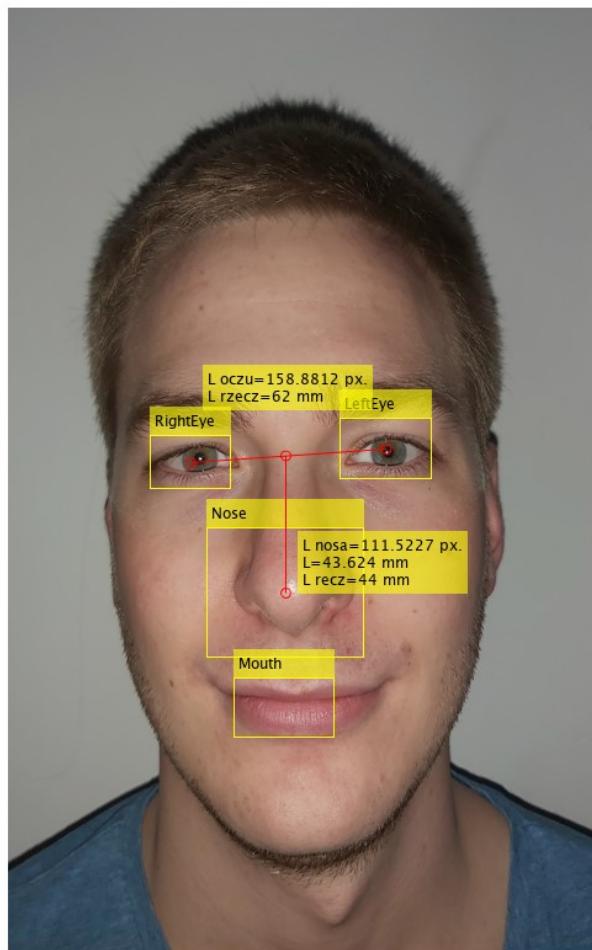
W ostatnim etapie dokonano naniesienia obszarów na zdjęcie oryginalne oraz adnotacji z wymiarami. Wymiary rzeczywiste posiadają format „ X_{rzecz} ” a wymiary obliczone na podstawie skali „ X_{skala} ”. Narysowane zostały także linie wymiarów i na ich podstawie dokonano rzeczywistych pomiarów. Dodatkowo połączone zostały wyznaczone obszary w jeden obraz binarny.



Rysunek 8.11. Obraz oryginalny z naniesionymi obszarami, wymiarami i adnotacjami.



Rysunek 8.12. Obraz połączonych obszarów określających położenie cech twarzy.



Rysunek 8.13. Wynik wykrywania cech z wykorzystaniem gotowych modułów.

W tym rozdziale pokazano, że możliwe jest w miarę dokładne określenie cech twarzy czy wymiarów na podstawie zdjęcia i jego przetwarzania. Zakładając, że zdjęcie wykonane byłoby na tle z odpowiednią podziałką możliwe jest dokładne określenie wymiarów w systemie wizyjnym. Dodatkowo wykorzystanie zaimplementowanych algorytmów znacznie przyspiesza pracę. Wykorzystując je do określania obszarów analizy a następnie stosując pokazane własne operacje morfologiczne można jeszcze dokładniej określić parametry.

Poniżej zamieszczone zostaną kody programów.

Tab. 8.2. Kod programu "Wykrywanie_cech.m".

```
%% Wczytanie zdjęcia w rozmiarze 768x480
clc; clear all; close all;
zdj = 'zdj_3.bmp';
i = imread(zdj);
info = imfinfo(zdj)
figure(1);
imagesc(i); title('Obraz orginalny');
% Zmierzone wartości
L_oczu = 62;           % [mm] odległość źrenic
L_nosa = 44;            % [mm] środek oczu - środek chrząstki nosa
L_ust = 46;              % [mm] długość ust bez kącików
L_policzki = 130;        % [mm] odległość policzków
W_ust = 18;              % [mm] wysokość ust
% Podział na nnn+1 regionów
nnn = 6;
reg_ver = [1, size(i,1)*(1:1:nnn) / (nnn)];
reg_hor = [1, size(i,2)*(1:1:nnn) / (nnn)];

%% Wykrywanie ust
% Wykrycie ust z pomocą różnicy kanałów R-G
i_m = i(:,:,1)-i(:,:,2);
% histogram, pobranie liczby pikseli i odpowiadającym im kolorow
[cn, bn] = imhist(i_m);
% określenie maksimum histogramu
[max, idx] = max(cn(2:end));
% poziom jasności o największej liczbie pixelow
bn(idx);
% Odjęcie do obrazu R-G wartości maksymalnego poziomu
i_m2 = imadd(i_m, -bn(idx));
% dostosowanie histogramu
i_m2 = imadjust(i_m2);

% Obraz po odjęciu R-G
figure(10); sgttitle('R-G');
subplot(121)
imhist(i_m)
subplot(122)
imshow(i_m)

% Po odjęciu od R-G wartości maksymalnego poziomu i poprawie histogramu
figure(11); sgttitle('R-G - max(hist)');
subplot(121)
imhist(i_m2)
subplot(122)
imshow(i_m2)

% binaryzacja z progiem 244 (najjaśniejsze pixele to usta)
```

```

i_usta = imbinarize(i_m2, 244/255);
% Obszar poza dolnym regionem = 0
i_usta([reg_ver(1):reg_ver(length(reg_ver)-2)],:) = 0;
i_usta = bwmorph(i_usta, 'clean'); % Wyczyszczenie pojedyńczych pikseli
i_usta = bwmorph(i_usta, 'fill'); % Wypełnienie dziur
figure(12)
for o=1:4
    % Zamknięcie z rosnącym elementem strukturalnym
    i_usta = imclose(i_usta, strel('disk', o));
    imshowpair(i, i_usta, 'montage');
end
% po zamknięciu
imshowpair(i, i_usta, 'montage'); title('Usta po zamknięciu')

i_usta = imopen(i_usta, strel('disk', 2)); % Otwarcie
% po zamknięciu
figure(13)
imshowpair(i, i_usta, 'montage'); title('Końcowy region ust');
% Pobranie właściwości regionu
usta_prop = regionprops(i_usta, 'all')

%% Wyszukanie oczu
i_oczyrgb = i;
i_oczygray = rgb2gray(i_oczyrgb); % Skala szarości
i_oczybw = imbinarize(i_oczygray, 33/255); % Binaryzacja (czarne żrenice)
i_oczybw = imcomplement(i_oczybw); % Odwrócenie obrazu binarnego
i_oczybw([reg_ver(1):reg_ver(3)],(:,:, :) = 0; % Zerowanie regionów
i_oczybw([reg_ver(4):reg_ver(end)],(:,:, :) = 0; % Zerowanie regionów
% Obrazy rgb, skala szarości i po binaryzacji
figure(20); subplot(131); sgttitle('Szukanie żrenic binaryzacja');
imshow(i_oczyrgb)
subplot(132); imshow(i_oczygray);
subplot(133); imshow(i_oczybw);

% Operacje morfologiczne - czyszczenie pojedyńczych pikseli, wypełnianie
% otwótorów (odbicie światła aparatu w żrenicach)
i_oczybw = bwmorph(i_oczybw, 'clean');
i_oczybw = bwmorph(i_oczybw, 'fill', 8);
i_oczybw = imfill(i_oczybw, 'holes');
% Otwarcie w celu pozbycia się elementów nie będących okrągami
i_oczybw = imopen(i_oczybw, strel('disk', 3));
figure(21)
title('Znalezione regiony żrenic')
imshow(i_oczybw);
% Wyznaczenie właściwości regionów
oczy_prop = regionprops(i_oczybw, 'all')

%% Wyszukanie nosa
i_nosrgb = i;
% Zerowanie regionów poza obszarem zainteresowania
i_nosrgb([reg_ver(1):reg_ver(3)],(:,:, :) = 0;
i_nosrgb([reg_ver(5):reg_ver(end)],(:,:, :) = 0;
i_nosrgb(:,[reg_hor(1):reg_hor(3)],:) = 0;
i_nosrgb(:,[reg_hor(5):reg_hor(end)],:) = 0;
i_nosgray = rgb2gray(i_nosrgb); % Skala szarości

% Binaryzacja (wykorzystany refleks na nosie)
i_nosbw = imbinarize(i_nosgray, 194/255);

figure(30); sgttitle('Obszar wyszukiwania nosa');

```

```

subplot(131)
imshow(i_nosrgb);
subplot(132)
imshow(i_nosgray);
subplot(133)
imhist(i_nosgray)

% Operacje morfologiczne, czyszczenie i wypełnianie
i_nosbw = bwmorph(i_nosbw, 'clean');
i_nosbw = bwmorph(i_nosbw, 'fill');
% Zamknięcie i otwarcie elementem dyskowym
i_nosbw = imclose(i_nosbw, strel('disk', 3));
i_nosbw = imopen(i_nosbw, strel('disk', 2));
figure(32); title('Refleks środka chrzastki nosa');
imshow(i_nosbw)
% Wyznaczenie właściwości regionu
nos_prop = regionprops(i_nosbw, 'all')

%% Wyszukanie skrajni policzków
i_polrgb = i;
i_polgray = rgb2gray(i_polrgb); % Skala szarości

figure(40); title('Skala szarości');
imshow(i_polgray);

figure(41); sgttitle('Binaryzacja na podstawie histogramu');
subplot(131);
imhist(i_polgray)

subplot(132);
i_polbw = imbinarize(i_polgray, 88/255);
imshow(i_polbw);

% Pełna jasność w regionach innych niż skajnych środkowych
i_polbw([reg_ver(1):reg_ver(4)],(:,:,:) = 1;
i_polbw([reg_ver(5):reg_ver(end)],(:,:,:) = 1;
i_polbw(:, [reg_hor(3):reg_hor(end-2)]) = 1;
i_polbw = imcomplement(i_polbw); % Odwrócenie obrazu binarnego

subplot(133);
imshow(i_polbw);

% Czyszczenie i wypełnianie luk
i_polbw = bwmorph(i_polbw, 'clean');
i_polbw = bwmorph(i_polbw, 'fill');
% Otwarcie elementem liniowym pionowym ( liniowość rys skrajni policzków)
a
% następnie dyskowym w celu wypełnienia i znowu elementem linowym
i_polbw = imopen(i_polbw, strel('line', 15, 90));
i_polbw = imopen(i_polbw, strel('disk', 2));
i_polbw = imopen(i_polbw, strel('line', 20, 90));

% Obszary policzków
figure(42)
imshowpair(i, i_polbw, 'montage'); title('Regniony skrajni policzków')
% Wyznaczenie właściwości
pol_prop = regionprops(i_polbw, 'all')

%% Finalne złożenie wyznaczonych cech

```

```

i_f = i;

%%% Dodanie oznaczeń oczu i obliczenie odległości %%%
oko1_bbox = oczy_prop(1).BoundingBox;    % Prostokąt ograniczający
% Dodanie regionu z opisem
i_f= insertObjectAnnotation(i_f, 'rectangle', oko1_bbox, 'Oko prawe');
oko2_bbox = oczy_prop(2).BoundingBox;    % Prostokąt ograniczający
i_f= insertObjectAnnotation(i_f, 'rectangle', oko2_bbox, 'Oko lewe');

% Współrzędne środków żrenic x, y, indeksy: l -lewe - r - prawe
oko_r = oczy_prop(1).Centroid;    % Parametr środka cieżkości
oko_l = oczy_prop(2).Centroid;

% Wyznaczenie współrzędnych punktu pomiędzy oczyma
ys = (oko_r(2)+oko_l(2))/2;
xs = (oko_r(1)+oko_l(1))/2;

% Wymiar skalujący pozostałe wymiary z px. -> mm.
skala = L_oczu/(oko_l(1)-oko_r(1));

% Obliczenie odległości oczy, konstrukcja adnotacji i dodanie do obrazu
L_oczu_ = sqrt((oko_l(1)-oko_r(1))^2+(oko_l(2)-oko_r(2))^2);
temp_str = "Oczy" + newline + "L=" + num2str(L_oczu_) + ...
           "px." + newline + "L_rzecz=" + num2str(L_oczu) + "mm" + ...
           newline + "Skala=" + skala + "mm/px.";
i_f = insertText(i_f, [90, 105], temp_str, 'AnchorPoint', "Center");

%%% Dodanie oznaczenia ust i obliczenie wymiarów %%%
usta_bbox = usta_prop.BoundingBox;    % Prostokąt ograniczający
% Współrzędne prawego i lewego końca oraz środek wysokości
uxr = usta_bbox(1); uxl = usta_bbox(1) + usta_bbox(3);
uy = usta_bbox(2)+usta_bbox(4)/2;

% Dodanie regionu, konstrukcja adnotacji i dodanie do zdjęcia
i_f= insertObjectAnnotation(i_f, 'rectangle', usta_bbox, 'Usta');
temp_str = "L=" + num2str(usta_bbox(3)) + "px." + newline + ...
           "L_skala=" + num2str(usta_bbox(3)*skala) + "mm" + newline + ...
           "L_rzecz=" + num2str(L_ust) + "mm" + newline + "W=" + ...
           num2str(usta_bbox(4)) + newline + "W_skala=" + ...
           num2str(usta_bbox(4)*skala)+ "mm" + newline + "W_rzecz=" + ...
           num2str(W_ust) + "mm";

i_f = insertText(i_f, [90, 150], temp_str);

%%% Dodanie oznaczenia policzków i obliczenie odległości %%%
pol1_bbox = pol_prop(1).BoundingBox;    % Prostokąt ograniczający
i_f= insertObjectAnnotation(i_f, 'rectangle', pol1_bbox, 'Policzek');
pol2_bbox = pol_prop(2).BoundingBox;    % Prostokąt ograniczający
i_f= insertObjectAnnotation(i_f, 'rectangle', pol2_bbox, 'Policzek');

% Odległość skrajni policzków, współrzędne
pl = pol1_bbox(1);
pr = pol2_bbox(1)+pol2_bbox(3);
yp_sr = (pol_prop(1).Centroid(2) + pol_prop(2).Centroid(2) )/2;
L_pol = pr-pl;

```

```

temp_str = "L=" + num2str(L_pol) + "px." + newline + ...
    "L_skala=" + num2str(L_pol*skala) + "mm" + newline + ...
    "L_rzecz=" + num2str(L_policzki) + "mm";

i_f = insertText(i_f, [330, 80], temp_str);

%%% Dodanie regionu środka nosa i obliczenie odległości %%%
nos_bbox = nos_prop.BoundingBox; % Prostokąt ograniczający
% Współrzędne środka
nx = nos_prop.Centroid(1);
ny = nos_prop.Centroid(2);
i_f= insertObjectAnnotation(i_f, 'rectangle', nos_bbox, 'Nos');
% dlugosc nosa środek chrząstki do środka linii łączącej oczy
L_nosa_ = sqrt( (nx - xs)^2 + (ny - ys)^2);

temp_str = "Nos" + newline + "L=" + num2str(L_nosa_) + ...
    "px." + newline + "L_skala=" + num2str(L_nosa_*skala) + ...
    "mm" + newline + "L_rzecz=" + num2str(L_nosa) + "mm";

i_f = insertText(i_f, [180, 75], temp_str);

% Obraz z naniesionymi wszystkimi oznaczeniami i odległościami
figure(99);
imshow(i_f);
hold on;
line([oko_r(1), oko_l(1)], [oko_r(2), oko_l(2)], 'Color', 'red',
'Marker','.');
line([pl, pr], [yp_sr, yp_sr], 'Color', 'red', 'Marker','.');
line([nx, xs], [ny, ys], 'Color', 'red', 'Marker','.');
line([uxr, ux1], [uy, uy], 'Color', 'red', 'Marker','.');
line([(uxr+ux1)/2, (uxr+ux1)/2], [usta_bbox(2), ...
    usta_bbox(2)+usta_bbox(4)], 'Color', 'red', 'Marker','.');

% Binarny obraz wszystkich regionów
i_f2 = imadd(i_nosbw, i_usta);
i_f2 = im2bw(i_f2, 0.99);
i_f2 = imadd(i_f2, i_polbw);
i_f2 = im2bw(i_f2, 0.99);
i_f2 = imadd(i_f2, i_oczybw);

figure(100);
imshow(i_f2); title('Wszystkie wyznaczone regiony');

```

Tab. 8.3. Kod programu "Wykrywanie_oczu.m".

```

%% Wczytanie zdjęcia w rozmiarze 768x480
clc; clear all; close all;
zdj = 'zdj_3.bmp';
i = imread(zdj);
info = imfinfo(zdj)
figure(1);
imagesc(i); title('Obraz orginalny');
% Zmierzone wartości
L_oczu = 62;      % [mm] odległość źrenic
L_nosa = 44;       % [mm] środek oczu - środek chrząstki nosa

```

```

L_ust = 46;      % [mm] długość ust bez kącików
L_policzki = 130;    % [mm] odległość policzków
W_ust = 18;       % [mm] wysokość ust

% Inicjalizacja obiektu rozpoznawania twarzy
faceDetector = vision.CascadeObjectDetector;
bbox = faceDetector(i);           % Wykrycie twarzy, zapisanie obszaru
% Dodanie annotacji obszaru twarzy
i_f = insertObjectAnnotation(i, 'rectangle', bbox, 'Face');

% Inicjalizacja detektora oczu, zwiększenie zakresu scalania wykryć do 10
% Określenie minimalnej wielkości obszaru oczu
bothEyes =
vision.CascadeObjectDetector('ClassificationModel','EyePairBig');
bothEyes.MergeThreshold = 10; bothEyes.MinSize = [30 200];
% Inicjalizacja detektora nosa, zwiększenie zakresu scalania wykryć do 20
% Określenie minimalnej wielkości obszaru nosa
nose = vision.CascadeObjectDetector('ClassificationModel','Nose',
'UseROI', true);
nose.MergeThreshold = 20; nose.MinSize = [70 40];

% Wykrycie oczu, zapisanie obszaru, dodanie annotacji na zdj
bothEyesbbox = bothEyes(i);
i_f = insertObjectAnnotation(i_f, 'rectangle', bothEyesbbox, 'Eyes');

% Wykrycie nosa, zapisanie obszaru, dodanie annotacji na zdj
nosebbox = nose(i, bbox);
i_f = insertObjectAnnotation(i_f, 'rectangle', nosebbox, 'Nose');
% Zdjęcie z wykrytymi oczyma i nosem
figure(2);
imshow(i_f);

% Klasyfikacja wstępna pojedyńczych oczu
% Inicjalizacja detektorów, ograniczenie minimalnej wielkości, zakres
% scalania wyników
lEyeCart =
vision.CascadeObjectDetector('ClassificationModel','LeftEyeCART');
lEyeCart.MergeThreshold = 15; lEyeCart.MinSize = [30 60];
rEyeCart =
vision.CascadeObjectDetector('ClassificationModel','RightEyeCART');
rEyeCart.MergeThreshold = 15; rEyeCart.MinSize = [30 60];

% Wykrycie oczu, zapisanie obszarów, posłużą jako ograniczony obszar dla
% dokładniejszego detektora oczu
lEyeCartbbox = lEyeCart(i);
i_e = insertObjectAnnotation(i, 'rectangle', lEyeCartbbox, 'LeftEyeCart');
rEyeCartbbox = rEyeCart(i);
i_e = insertObjectAnnotation(i_e, 'rectangle', rEyeCartbbox,
'RightEyeCart');
% Zdjęcie z wykrytymi oczyma
figure(3);
imshow(i_e);

% Klasyfikacja dokładna, mniejszy klasyfikator, na podstawie regionu
% zaufania oraz scalanie wykryć
lEye = vision.CascadeObjectDetector('ClassificationModel','LeftEye',
'UseROI', true);
lEye.MergeThreshold = 10;
rEye = vision.CascadeObjectDetector('ClassificationModel','RightEye',
'UseROI', true);

```

```

rEye.MergeThreshold = 10;

% Wykrywanie oczu w obszarach wcześniejszy wyznaczonych zgrubnie
lEyebbox = lEye(i, lEyeCartbbox);
i_e2 = insertObjectAnnotation(i, 'rectangle', lEyebbox, 'LeftEye');
rEyebbox = rEye(i, rEyeCartbbox);
% Dodanie annotacji
i_e2 = insertObjectAnnotation(i_e2, 'rectangle', rEyebbox, 'RightEye');
i_e2 = insertObjectAnnotation(i_e2, 'rectangle', nosebbox, 'Nose');

% Obliczenie środka obszaru nosa
xn = nosebbox(1) + nosebbox(3)/2; yn=nosebbox(2)+nosebbox(4)/2;

% Środki oczu
x1 = lEyebbox(1) + lEyebbox(3)/2; y1 = lEyebbox(2) + lEyebbox(4)/2;
xr = rEyebbox(1) + rEyebbox(3)/2; yr = rEyebbox(2) + rEyebbox(4)/2;
ys = (yr+y1)/2; xs = (xr+x1)/2;

% Obliczenie odległości w px. z Pitagorasa
L = sqrt((x1-xr)^2+(y1-yr)^2);
L2 = sqrt((xs-xn)^2+(ys-yn)^2);
% Wyznaczenie skali na podstawie odległości oczu
skala = L_oczu / (x1-xr);
L_nosa_skala = L2*skala;

% Dodanie adnotacji
temp_str = "L_oczu=" + num2str(L) + " px." + newline + "L_rzecz=" + ...
    num2str(L_oczu) + " mm";
i_e2 = insertText(i_e2, [xs, ys*0.85], temp_str, 'AnchorPoint', "Center");

temp_str = "L_nosa=" + num2str(L2) + " px." + newline + "L=" + ...
    num2str(L_nosa_skala) + " mm" + newline + ...
    "L_recz=" + num2str(L_nosa) + " mm";
i_e2 = insertText(i_e2, [xn*1.35, yn*0.95], temp_str,
'AnchorPoint', "Center");

% Inicjalizacja detektora ust, zwiększenie zakresu scalania wykryć
mouth = vision.CascadeObjectDetector('ClassificationModel', 'Mouth',
'UseROI', true);
mouth.MergeThreshold = 50;
% Wykrycie ust, zapisanie obszaru, dodanie annotacji na zdj
mouthbox = mouth(i, bbox);
i_e2 = insertObjectAnnotation(i_e2, 'rectangle', mouthbox, 'Mouth');

% Zdjęcie z odległościami
figure();
imshow(i_e2); hold on;
line([xr, x1], [yr, y1], 'Color', 'red', 'Marker', 'o');
line([xn xn], [yn ys], 'Color', 'red', 'Marker', 'o'); hold off;

```

Tab. 8.4. Kod programu "Wykrywanie_oczu_kamera.m".

```
% Wykrywanie na żywo
%% Kamera
clc; close all; clear all;
cam = webcam(1)

%% Tworzenie detektorów cech charakterystycznych z modułu vision
% Twarz
faceDetector = vision.CascadeObjectDetector;
% Nos, i podanie zakresu łączenia wykryć, region zainteresowania
nose = vision.CascadeObjectDetector('ClassificationModel','Nose',
'UseROI', true);
nose.MergeThreshold = 50;
% Oczy 2x i obszar łączenia, region zainteresowania
bothEyes =
vision.CascadeObjectDetector('ClassificationModel','EyePairBig', 'UseROI',
true);
bothEyes.MergeThreshold = 10;
% Oczy lewe i prawe i łączenie, region zainteresowania
lEyeCart =
vision.CascadeObjectDetector('ClassificationModel','LeftEyeCART',
'UseROI', true);
lEyeCart.MergeThreshold = 15;
rEyeCart =
vision.CascadeObjectDetector('ClassificationModel','RightEyeCART',
'UseROI', true);
rEyeCart.MergeThreshold = 15;
% Oczy mniejszy detektor, łączenie
lEye = vision.CascadeObjectDetector('ClassificationModel','LeftEye',
'UseROI', true);
lEye.MergeThreshold = 10;
rEye = vision.CascadeObjectDetector('ClassificationModel','RightEye',
'UseROI', true);
rEye.MergeThreshold = 10;
%% Petla programu
% Zapis do pliku
v = VideoWriter('test.avi');
open(v);
for i=1:45
    i = snapshot(cam); i=imresize(i, [600 800]); % pobór klatki z kamery
    bbox = [];
    bbox = faceDetector(i); % wykrecie twarzy
    i_f = i;
    bothEyesbbox = [];
    nosebbox = [];
    if ~isempty(bbox)
        i_f = insertObjectAnnotation(i_f, 'rectangle', bbox, 'Face');
        bbox=bbox(1, :);
        nosebbox = nose(i, bbox);
        bothEyesbbox = bothEyes(i, bbox);
        i_f = insertObjectAnnotation(i_f, 'rectangle', bothEyesbbox,
'Eyes');
    end

    lEyebbox = [];
    if ~isempty(bothEyesbbox)
        lEyebbox = lEye(i, [bothEyesbbox(1)+bothEyesbbox(3)/2
bothEyesbbox(2) bothEyesbbox(3)/2 bothEyesbbox(4)]);
    end
end
```

```

        if ~isempty(lEyebbox)
            i_f = insertObjectAnnotation(i_f, 'rectangle', lEyebbox,
'LeftEye');
            xl = lEyebbox(1) + lEyebbox(3)/2; yl = lEyebbox(2) +
lEyebbox(4)/2;
        end
    end
    rEyebbox = [];
    if ~isempty(bothEyesbbox)
        rEyebbox = rEye(i, [bothEyesbbox(1) bothEyesbbox(2)
bothEyesbbox(3)/2 bothEyesbbox(4)]);
    end
    if ~isempty(rEyebbox)
        i_f = insertObjectAnnotation(i_f, 'rectangle', rEyebbox,
'RightEye');
        xr = rEyebbox(1) + rEyebbox(3)/2; yr = rEyebbox(2) +
rEyebbox(4)/2;
    end
end
if ~isempty(nosebbox)
    i_f = insertObjectAnnotation(i_f, 'rectangle', nosebbox, 'Nose');
    xn = nosebbox(1) + nosebbox(3)/2; yn=nosebbox(2)+nosebbox(4)/2;
end
if ~isempty(rEyebbox) && ~isempty(lEyebbox)
    ys = (yr+yl)/2; xs = (xr+xl)/2;
    L = sqrt((xl-xr)^2+(yl-yr)^2);
    i_f = insertText(i_f, [xs, ys*0.85], ['L=' num2str(L)],
'AnchorPoint','Center');
end
if ~isempty(rEyebbox) && ~isempty(lEyebbox) && ~isempty(nosebbox)
    L2 = sqrt((xs-xn)^2+(ys-yn)^2);
    i_f = insertText(i_f, [xn, yn*1.05], ['L=' num2str(L2)],
'AnchorPoint','Center');
end

imshow(i_f); hold on;
if ~isempty(rEyebbox) && ~isempty(lEyebbox)
    line([xr, xl], [yr, yl], 'Color', 'red', 'Marker', 'o');
end
if ~isempty(rEyebbox) && ~isempty(lEyebbox) && ~isempty(nosebbox)
    line([xn xn], [yn yn], 'Color','orange', 'Marker', 'o'); hold off;
end

frame = getframe(gcf);
writeVideo(v, frame);
end
close(v);

```