

Rozpoznawanie Tęczówki

Jakub Wolski (263619)

Contents

1 Wstęp	2
2 Akwizycja Danych	2
2.1 Baza danych - OFTA	2
2.2 Wybrane zdjęcia	2
3 Preprocessing i Redukcja Danych	3
3.1 Redukcja rozmiaru	3
3.2 Zamiana na skalę szarości	3
4 Segmentacja Obrazu	3
4.1 Detekcja krawędzi	3
Wygładzanie krawędzi	
4.2 Kołowa transformata Hough'a	4
Przykładowa implementacja • Akumulacja wartością krawędzi lub 1	
4.3 Operator różniczkowo-całkowy Daugmana	8
Przykładowa implementacja	
4.4 Omijanie przeszkodek podczas znajdywania okręgów	9
4.5 Nierównomierne oświetlenie	9
5 Rozwijanie Tęczówki	9
5.1 Normalizacja obrazu tęczówki - "rubber sheet model" Daugmana9	
6 Ekstrakcja i Selekция Cech - Kodowanie Wzoru Tęczówki	10
6.1 Uśrednianie w paski	10
6.2 Schemat Daugmana	10
6.3 Filtr Gabora	11
6.4 Ustalenie parametrów filtra Gabora	11
Długość fali - λ • Orientacja - Θ • Odchylenie standardowe - σ	
6.5 Filtrowanie filtrem Gabora	13
6.6 Binaryzacja - zakodowanie tęczówki	13
6.7 Niepełne dwubitowe kodowanie	13
7 Zapis Wzorca	14
7.1 Maska słabych i mocnych bitów	14
8 Moduł Decyzyjny	14
8.1 Odległość Hamminga	14
Ukazywanie różnic w kodzie	
8.2 Przesuwanie bitów	15
8.3 Inne metody porównywania	15
9 Preferowane Ustawienia Systemu	15
10 Ocena Jakości Systemu	15
10.1 Macierz konfuzji	15
10.2 Rozkłady wewnętrzne oraz międzyklasowy	16
Odległość d dla preferowanego ustawienia systemu	
10.3 Porównanie różnych ustawień systemu	16
Preferowane ustawienia • Bez wygładzania obrazu przed detekcją krawędzi • Akumulacja wartością 1 • Różne wartości parametru długości fali filtra Gabora	
10.4 Inne metody oceny jakości	17
11 Wnioski	17
12 Kod Programu	18
12.1 Plik main	18
12.2 Funkcje pomocnicze	18
decrease_resolution() • detect_edges() • find_iris() • hough_transform_iris() • find_pupil() • hough_transform_pupil() • daugman_script() • unwrap_iris() • split_strips_filter() • apply_gabor_filter() • hamming_distance()	

13 Linki

References 21

List of Figures

1	Przedstawienie zarysu systemu biometrycznego	2
2	Osoba 1	3
3	Osoba 2	3
4	Osoba 3	3
5	Osoba 4	3
6	Osoba 5	3
7	Osoba 6	3
8	3
9	3
10	4
11	4
12	4
13	Przykład rysowania okręgów w kołowej transformacji Hough'a[5]	4
14	Znalezienie koła o zadanym promieniu metodą Hough'a [4]	4
15	Wynik znajdywania okręgu dla różnych r [7]	5
16	Zlokalizowane okręgi tęczówki oraz żrenicy	5
17	5
18	Nieprawidłowe zlokalizowanie tęczówki dla progu 0.1 (metoda akumulacji jedynek)	5
19	Przestrzeń Hough'a dla nieprawidłowo zlokalizowanej tęczówki (2D) (metoda akumulacji jedynek)	5
20	Przestrzeń Hough'a dla nieprawidłowo zlokalizowanej tęczówki (3D) (metoda akumulacji jedynek)	6
21	Wykres 3D ze znalezionymi krawędziami	6
22	Prawidłowe zlokalizowanie tęczówki dla progu 0.2 (metoda akumulacji jedynek)	6
23	Przestrzeń Hough'a dla prawidłowo zlokalizowanej tęczówki (2D) (metoda akumulacji jedynek)	6
24	Przestrzeń Hough'a dla prawidłowo zlokalizowanej tęczówki (3D) (metoda akumulacji jedynek)	6
25	Nieprawidłowe zlokalizowanie żrenicy dla zbyt dużego zakresu $r \in \langle 15; 70 \rangle$ (metoda akumulacji wartości krawędzi)	7
26	Przestrzeń Hough'a dla nieprawidłowo zlokalizowanej żrenicy (2D)(metoda akumulacji wartości krawędzi)	7
27	Przestrzeń Hough'a dla nieprawidłowo zlokalizowanej żrenicy (3D)(metoda akumulacji wartości krawędzi)	7
28	Wartości krawędzi plamki oraz żrenicy na obrazie wykrytej krawędzi	7
29	Prawidłowe zlokalizowanie żrenicy dla zakresu $r \in \langle 35; 70 \rangle$ (metoda akumulacji wartości krawędzi)	7
30	Przestrzeń Hough'a dla prawidłowo zlokalizowanej żrenicy (2D) (metoda akumulacji wartości krawędzi)	7
31	Przestrzeń Hough'a dla prawidłowo zlokalizowanej żrenicy (3D) (metoda akumulacji wartości krawędzi)	8
32	Okrąg najbardziej pasujący do istniejącego posiada największą wartość sumy [7]	8
33	Wynik implementacji operatora Daugmana (oko 2a)	8
34	Wynik implementacji operatora Daugmana (oko 4a)	8
35	Przykłady występowania tęczówki oraz żrenicy na oku	9
36	Graficzne przedstawienie idei modelu "rubber sheet" Daugmana	9

37	Przedstawienie efektów rozwijania tęczówki metodą "rubber sheet"	10
38	Oryginalny obraz tęczówki	10
39	Podzielenie obrazu tęczówki na 8 pasków (powiększony obraz)	10
40	Pasek obrazu tęczówki przed i po filtracji Gausem	10
41	Graficzne przedstawienie schematu Daugmana	11
42	Filtr Gabora - $\lambda = 4$	11
43	Filtr Gabora - $\lambda = 8$	11
44	Filtr Gabora - $\lambda = 16$	12
45	Filtr Gabora - $\lambda = 32$	12
46	Widmo częstotliwościowe pierwszego paska z uśrednionego i podzielonego obrazu tęczówki	12
47	Filtr Gabora - $\Theta = 0$	12
48	Filtr Gabora - $\Theta = 45$	12
49	Filtr Gabora - $\Theta = 90$	12
50	Filtr Gabora - $\Theta = 135$	12
51	Filtr Gabora - $\sigma = 5$	13
52	Filtr Gabora - $\sigma = 10$	13
53	Filtr Gabora - $\sigma = 15$	13
54	Filtr Gabora - $\sigma = 20$	13
55	Obraz uśrednionej i podzielonej tęczówki po zastosowaniu filtra Gabora	13
56	Zakodowana tęczówka po zastosowaniu binaryzacji na zfiltrowanym obrazie	13
57	Zakodowana tęczówka z 3 różnymi wartościami	14
58	Różnice pomiędzy kodami tęczówek różnych osób: $HD_R = 0.49, HD_I = 0.51$	14
59	Różnice pomiędzy kodami tęczówek tej samej osoby, dla różnych ujęć tęczówki: $HD_R = 0.24, HD_I = 0.25$	14
60	Różnice pomiędzy kodami tęczówek tej samej osoby, dla tego samego zdjęcia tęczówki : $HD_R = 0.00, HD_I = 0.00$	15
61	Idea przesuwania bitów podczas obliczania odległości Hamminga [6]	15
62	Macierz konfuzji dla preferowanych ustnień systemu (część rzeczywista i urojona)	15
63	Nieprawidłowo znalezione okręgi tęczówki i żrenicy dla oka 7c	15
64	Rozkłady wewnętrz i międzyklasowe	16
65	Macierz konfuzji dla preferowanych ustnień systemu (część rzeczywista i urojona)	16
66	Macierz konfuzji dla systemu bez wygładzania obrazu przed detekcją krawędzi	17
67	Macierz konfuzji dla systemu z metodą akumulacji jedynkami w transformacie Hough'a	17
68	Macierz konfuzji dla filtra Gabora: $\lambda = 25$	17
69	Macierz konfuzji dla filtra Gabora: $\lambda = 33$	17
70	Macierz konfuzji dla filtra Gabora: $\lambda = 142$	17

1. Wstęp

Niniejszy dokument przedstawia opis, rozwój oraz wyniki projektu systemu biometrycznego, który jest oparty na rozpoznawaniu i klasyfikowaniu tęczówki na podstawie jej analizy. Skupiał się wykrywaniu tęczówki oraz jej unikalnych cech, które by pozwoliły na zakodowanie tych cech oraz porównanie ich między sobą. Na rysunku 1 przedstawiono ogólny zarys co obejmuje typowy system biometryczny. W tym projekcie pominięto kroki akwizycji danych oraz tworzenie wzorców i bazy wzorców.

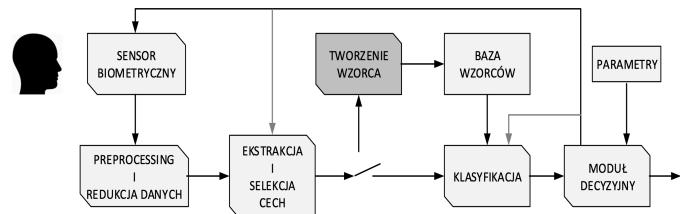


Figure 1. Przedstawienie zarysu systemu biometrycznego

Uwaga - główną jednostką wykorzystywaną w opisie tego projektu to piksele zdjęć (chyba, że zaznaczono inaczej).

2. Akwizycja Danych

Aby system mógł przetworzyć i zachować wzorzec oka lub porównać go ze wzorcem ze swojej bazy wzorców musi najpierw w jakiś sposób zdobyć te dane. Zazwyczaj są one pobierane poprzez odpowiednie kamery (lub inne sensory biometryczne), które robią zdjęcie oka danej osoby i używają tego zdjęcia jednak dla uproszczenia i ułatwienia tego procesu zdecydowano się skorzystać z gotowej bazy zdjęć.

2.1. Baza danych - OFTA

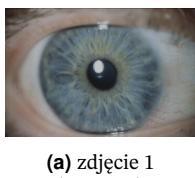
Do tego projektu skorzystano z bazy danych udostępnionej przez prowadzącego o nazwie *OFTA*. Zawiera 63 zdjęcia oczu pobranych z 21 osób (każdej osobie zrobiono po 3 zdjęcia jednego oka). Szerokości tych zdjęć różnią się pomiędzy siebie w zakresie $(800; 1000)$ pikseli, natomiast w wysokości wszystkie mają po 600 pikseli. Zdjęcia wydają się być bardziej oświetlone z lewej strony co można zauważać po charakterystycznym odbiciu lampy na obrzeżach żrenicy (na większości zdjęć znajduje się ono w lewej górnej części żrenicy) - to może spowodować pewne nierówności podczas przetwarzania tych obrazów (przewyższanie lewej strony nad prawą) co opisano szerzej w rozdziale 4.5.

2.2. Wybrane zdjęcia

Do celów tego projektu zdecydowano się wybrać 18 zdjęć (6 osób), gdzie wzorcem każdej osoby będzie pierwsze zdjęcie z bazy danych z tego powodu, że prościej jest ukazać wyniki systemu klasyfikującego na mniejszym zbiorze oraz dlatego, że przetwarzanie takiej bazy jest szybsze. Podczas wybierania zdjęć skupiono się na dobieraniu jak najbardziej różnorodnych przypadków aby ukazać działanie systemu dla różnych zdjęć - starano się dobierać zdjęcia z cechami takimi jak:

- przymknięta powieka
- brak widocznej powieki
- podobne ujęcia oka
- znacznie różniące się ujęcia oka
- różne położenie lampy oświetleniowej na oku, itp.

Poniżej przedstawiono wybrane zdjęcia:



(a) zdj. 1 (wzorzec)



(b) zdj. 2



(c) zdj. 3

Figure 2. Osoba 1



(a) zdj. 1 (wzorzec)



(b) zdj. 2



(c) zdj. 3

Figure 3. Osoba 2



(a) zdj. 1 (wzorzec)



(b) zdj. 2

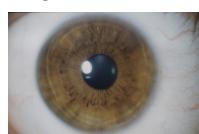


(c) zdj. 3

Figure 4. Osoba 3



(a) zdj. 1 (wzorzec)



(b) zdj. 2



(c) zdj. 3

Figure 5. Osoba 4



(a) zdj. 1 (wzorzec)

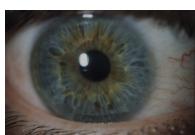


(b) zdj. 2



(c) zdj. 3

Figure 6. Osoba 5



(a) zdj. 1 (wzorzec)



(b) zdj. 2



(c) zdj. 3

Figure 7. Osoba 6

3. Preprocessing i Redukcja Danych

Pierwszym krokiem po akwizycji danych jest odpowiednie przetworzenie zdjęć w celu przyspieszenia procesu liczenia i pozbycie się zbędnych danych. Dlatego zdecydowano się na:

- zmniejszenie wymiaru zdjęć - to pomoże zmniejszyć ilość obliczeń do wykonania przy czym nie traci się kluczowych informacji o obrazie.
- zamiana obrazu na skalę szarości - informacja o kolorze jest zbędna do zakodowania tęczówki dlatego zamiana na skalę szarości pozwala zmniejszyć ilość macierzy z 3 do 1.

3.1. Redukcja rozmiaru

Aby osiągnąć redukcję rozmiaru zdjęcia napisano funkcję `decrease_resolution()`, która wykorzystuje wbudowaną funkcję Matlab `imresize()` (kod z rozdziału 12.2.1).

Po empirycznych eksperymentach zdecydowano się przyjąć wartość zmiennej `resolution_ratio = 0.5` (kod z rozdziału 12.1, linie 5, 25) - czyli wymiary tego zdjęcia zostaną zmniejszone o połowę. Jest to odpowiedni kompromis pomiędzy zmniejszeniem ilości danych a utrzymaniem potrzebnej informacji o zdjęciu.

3.2. Zamiana na skalę szarości

Aby zamienić zdjęcie ze skali RGB do skali szarości wykorzystano wbudowaną funkcję Matlab: `rgb2gray()`. Ten krok wykonano w funkcji detekcji krawędzi (kod z rozdziału 12.2.2, linie 6-10).

4. Segmentacja Obrazu

Celem tej części projektu jest wyciągnięcie informacji ze zdjęcia oka, które będzie potrzebna do zlokalizowania tęczówki. Zatem przed wszystkim należy znaleźć:

- krawędź tęczówki (promień oraz współrzędne środka okręgu)
- krawędź żrenicy (promień oraz współrzędne środka okręgu)

Główne metody wykorzystane do znajdywania okręgów to:

- kołowa transformata Hougha
- operator różniczkowo-całkowy Daugmanna

4.1. Detekcja krawędzi

Pierwszym krokiem zlokalizowania okręgów jest wyznaczenie krawędzi znajdujące się w obrazie. W tym celu wykorzystano metodę filtrowania krawędziowego do znajdywania pikseli gdzie wartości pikseli się gwałtownie zmieniają. Ta metoda polega na przepuszczaniu 2 filtrów (poziomy oraz pionowy), które obliczają różnice pomiędzy kolejnymi pikselami w tych kierunkach. Poniżej przedstawiono pseudokod realizujący tą filtrację:

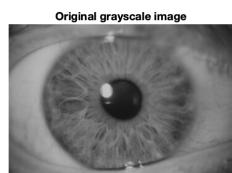
```

1 Gh = [-1, -1, -1, 0, 1, 1, 1]
2 Gv = Gh^T
3
4 horizontal = conv2D(in_image, Gh)
5 vertical = conv2D(in_image, Gv)
6
7 horizontal = horizontal*horizontal
8 vertical = vertical*vertical
9 out_image = sqrt(horizontal + vertical)

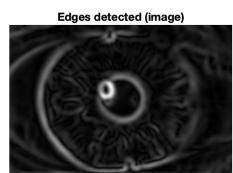
```

Code 1. Pseudokod realizujący filtrowanie krawędziowe

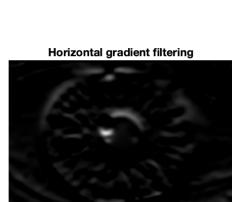
Wynik takiej filtracji przedstawiono na rysunkach 8a, 8b, 9a oraz 9b.



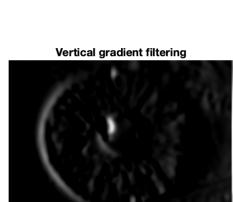
(a) Oryginalne zdjęcie



(b) Znalezione krawędzie (połączone)



(a) Znalezione poziome krawędzie



(b) Znalezione pionowe krawędzie

Figure 9

4.1.1. Wygładzanie krawędzi

Aby otrzymać odpowiednie krawędzie do wykorzystania w metodach Hough'a oraz Daugmana (rozdziały 4.2, 4.3) należy wygładzić zdjęcia przed potraktowaniem filtrami krawędziowymi.

Poniżej przedstawiono rysunki ukazujące znalezione krawędzie bez wygładzania na obrazie 3D (do tego wykorzystano sztucznie wygenerowane "oko"):

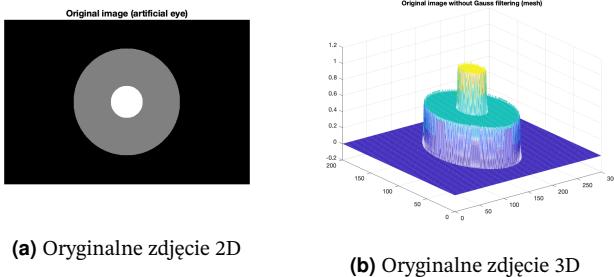


Figure 10

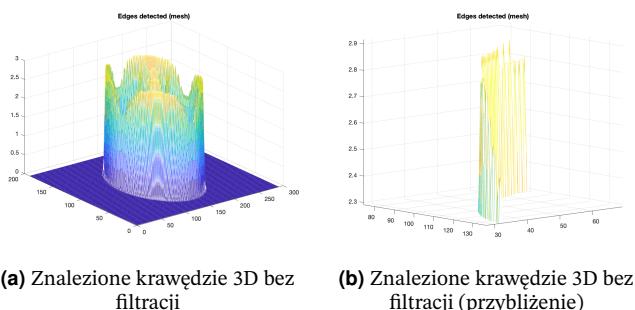


Figure 11

Z rysunków 11a i 11b wynioskowano, że krawędzie są zbyt ostre, żeby wyznaczyć okręgi tęczówki/źrenicy w sposób systematyczny i konsekwentny, ponieważ małe zmiany w detekcji krawędzi mogą spowodować dużą zmianę w znajdywaniu okręgu - dlatego należy wygładzić ten szczyt za pomocą filtra Gaussa.

Poniżej przedstawiono wpływ filtrowania zdjęcia na detekcję krawędzi:

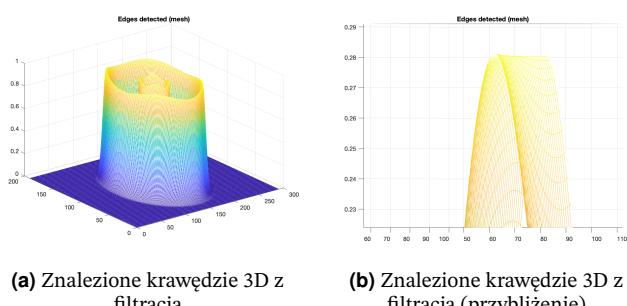


Figure 12

Jak można zauważać na rysunkach 12a i 12b znalezione krawędzie mają wygładzony kształt, podobny do funkcji gaussowskiej, co pomaga wyznaczyć środek (szczyt) krawędzi - to jest konieczne do wyznaczenia prawidłowych współrzędnych okręgu.

Ważną rzeczą na tym etapie była konieczność wycięcia z końcowego zdjęcia obszaru brzegowego - czyli tej części gdzie filtr musiał się "rozkręcić". Jeśli tego nie zrobiono to w kolejnych etapach znajdywanie okręgów stawało się wręcz niemożliwe, gdyż wartości na

tych brzegach znacznie wpływały na obliczenia. W tym celu obliczono wartość przesunięcia, które trzeba wziąć pod uwagę podczas wycinania brzegów obrazu.

Zaimplementowany kod do wykrywania krawędzi umieszczono w rozdziale 12.2.2.

Teraz posiadając już odpowiednią metodę na detekcję krawędzi należy przejść do procesu znajdywania okręgów nalepiej opisujących tęczówkę oraz żrenicę.

4.2. Kołowa transformata Hough'a

Jedną z metod wyznaczania okręgów to kołowa transformata Hough'a, która transformuje zbiór $P(x, y)$ punktów spełniających równanie 3 w punkt (a, b, r) w tzw. przestrzeni Hough'a i również działa w drugą stronę, jak rozpisano we wzorach 1 i 2.

$$P(x, y) \mapsto O(a, b, r) \quad (1)$$

$$O(a, b, r) \mapsto P(x, y) \quad (2)$$

$$r^2 = (x - a)^2 + (y - b)^2 \quad (3)$$

Działa ona w taki sposób, że dla każdego punktu wykrytej krawędzi rysuje okrąg o danym promieniu na podstawie wzoru 3 (przykład widoczny na rysunku 13) gdzie

- a, b - współrzędne punktu wykrytej krawędzi
- r - zadany promień
- x, y - punkty, przez które przechodzi okrąg o danych współrzędnych i danym promieniu

oraz kumuluje wszystkie wyrysowane wartości okręgów w przestrzeni Hough'a (przykład widoczny na rysunku 14a). Na koniec, ten punkt w przestrzeni Hough'a, który ma największą wartość jest środkiem okręgu o zadanym promieniu r (rysunek 14b) - z tego powodu ta metoda wymaga obliczenia dla wielu promieni w celu znalezienia odpowiednich współrzędnych (wtedy z wielu przestrzeni Hough'a wybierana jest ta, która ma największą wartość środka okręgu)[4].

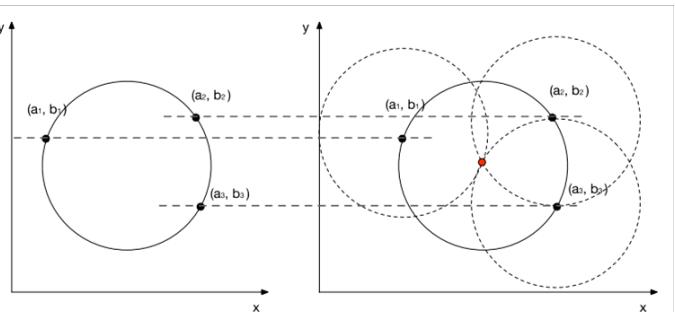


Figure 13. Przykład rysowania okręgów w kołowej transformacie Hough'a[5]

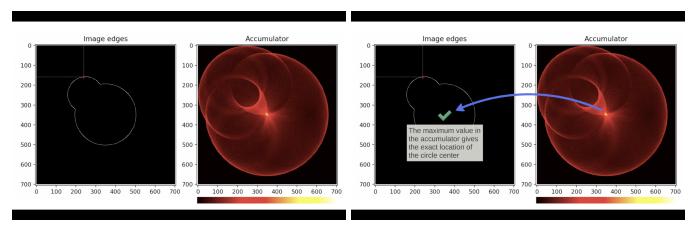


Figure 14. Znalezienie koła o zadanym promieniu metodą Hough'a [4]

Jeżeli promień jest zbyt mały lub zbyt duży to wtedy transformata nie znajdzie poszukiwanego okręgu, dlatego należy przejść po wielu

promieniach. Wynik transformacji Hough'a dla różnych promieni przedstawiono na rysunku 15, z których można wywnioskować, że ta transformata znajduje jedynie ten jeden okrąg o specyficzny promieniu.

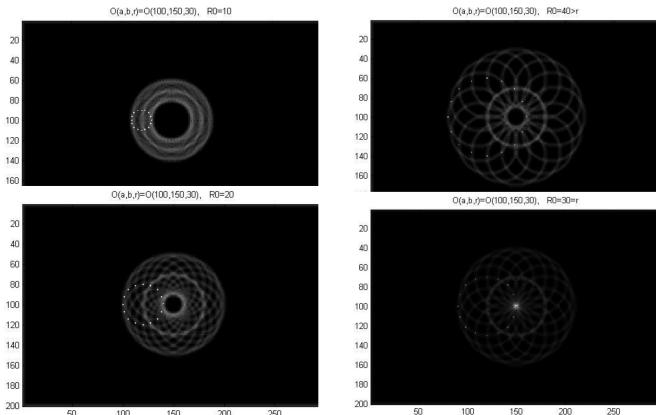


Figure 15. Wynik znajdywania okręgu dla różnych r [7]

4.2.1. Przykładowa implementacja

W kodzie poniżej ukazano przykładową implementację kołowej transformaty Hough'a.

```

1  for all r in (r_min..r_max)
2      for all x
3          for all y
4              if (x,y) is an edge point
5                  for all theta in (0..2pi)
6                      a=x-r*cos(theta)
7                      b=y-r*sin(theta)
8                      HS(a,b,r)=+1
9                      %lub
10                     HS(a,b,r)=+v(x,y)
11                 end
12             end
13         end
14     end
15 
```

Code 2. Kołowa Transformacja Hough'a - pseudokod [7]

Wyniki implementacji tego pseudokodu w Matlabie przedstawiono na rysunkach 16, 17a oraz 17b. Odpowiedni kod jest udostępniony w rozdziałach 12.2.4 oraz 12.2.6.

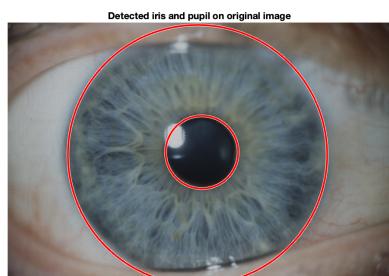
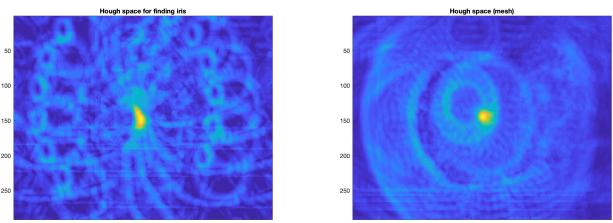


Figure 16. Zlokalizowane okręgi tęczówki oraz żrenicy



(a) Obliczona przestrzeń Hough'a dla **(b)** Obliczona przestrzeń Hough'a dla
tęczówki ($r = 150$ pikseli) żrenicy ($r = 42$ piksele)

Figure 17

4.2.2. Akumulacja wartością krawędzi lub 1

W pseudokodzie w rozdziale 4.2.1 (oraz w zaimplementowanym kodzie w Matlabie) wykorzystano dwa sposoby akumulowania wartości podczas obliczania przestrzeni Hough'a:

1. dodawanie jedynki
2. dodawanie wartości punktu (a, b) z obrazu krawędzi

Pierwsza metoda musi korzystać z warunku przynależności do krawędzi (kod 4.2.1 - linia 4), ponieważ dodawanie jedynki dla każdego punktu rysowanego okręgu nie ma sensu - potrzebuje wysyłać tylko wtedy kiedy jest pewność, że dany punkt należy do znalezionej krawędzi. Z tego powodu ta metoda polega na dobraniu odpowiedniego warunku przynależenia do krawędzi oraz na podaniu odpowiedniego zakresu szukanych promieni. Poniżej przedstawiono sytuację gdzie warunek, w postaci progu wartości (12.2.4 linia 10), został nieprawidłowo zdefiniowany (rysunki 25, 26 i 27).

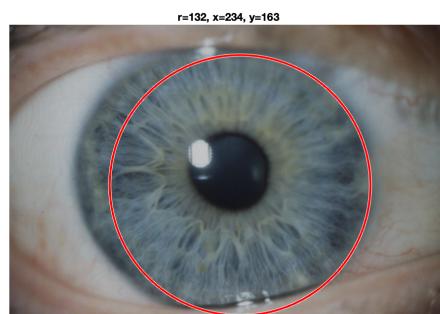


Figure 18. Nieprawidłowe zlokalizowanie tęczówki dla progu 0.1 (metoda akumulacji jedynek)

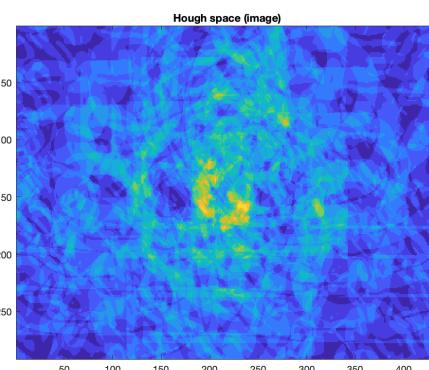


Figure 19. Przestrzeń Hough'a dla nieprawidłowo zlokalizowanej tęczówki (2D) (metoda akumulacji jedynek)

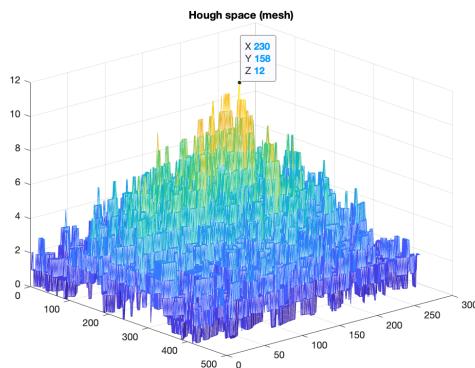


Figure 20. Przestrzeń Hough'a dla nieprawidłowo zlokalizowanej tęczówki (3D) (metoda akumulacji jedynek)

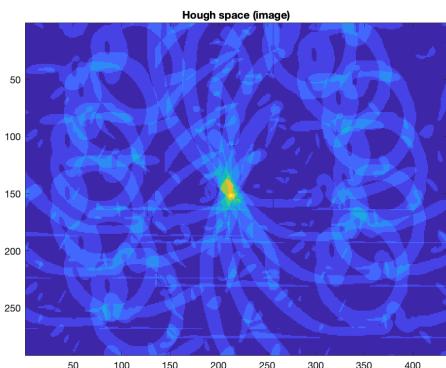


Figure 23. Przestrzeń Hough'a dla prawidłowo zlokalizowanej tęczówki (2D) (metoda akumulacji jedynek)

Można wyraźnie zauważyć, że ta metoda nie zadziałała zgodnie z oczekiwaniemi z tego powodu, że próg o wartości 0.1 jest zbyt niski - dlatego, posługując się wykresem wartości krawędzi (widocznym na rysunku 21), zwiększo jego wartość do 0.2 aby pozbyć się szumu, który zniekształca na ostateczny wynik przestrzeni Hough'a. Wynik tej operacji przedstawiono na rysunkach 21, 22, 23 oraz 24.

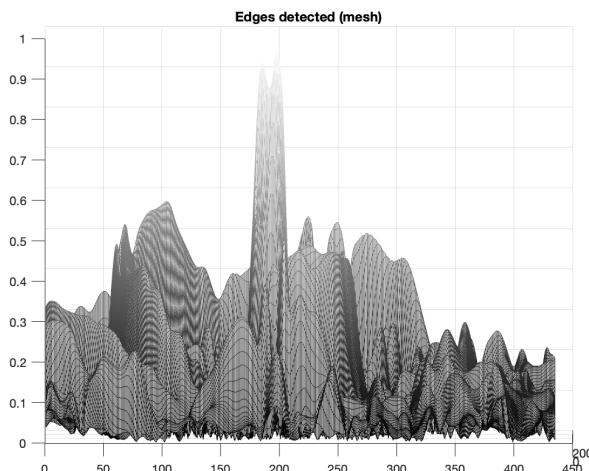


Figure 21. Wykres 3D ze znalezionymi krawędziami

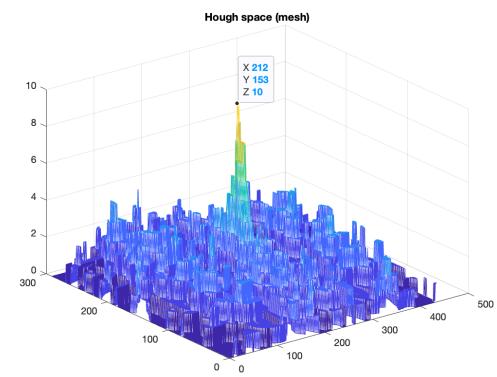


Figure 24. Przestrzeń Hough'a dla prawidłowo zlokalizowanej tęczówki (3D) (metoda akumulacji jedynek)

Operacja zwiększenia progu okazała się skuteczna.

W drugim sposobie obliczanie przestrzeni Hough'a nie ma potrzeby korzystać z warunki przynależności do obrazu krawędziowego, ponieważ polega ono na obecnej wartości tego obrazu (tam gdzie będzie krawędź będzie większa wartość), chociaż ten warunek może znacznie przyspieszyć proces obliczeń. Natomiast jej efektywność zależy od innych czynników:

- dobrej detekcji krawędzi
- wygładzenia krawędzi
- odpowiedni dobór zakresu szukanych promieni

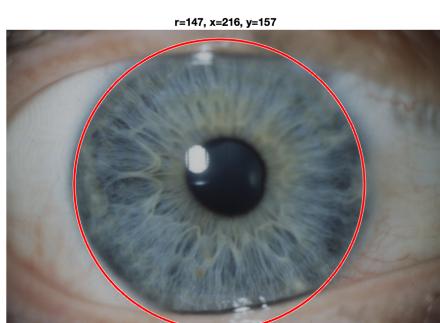


Figure 22. Prawidłowe zlokalizowanie tęczówki dla progu 0.2 (metoda akumulacji jedynek)

Dobra detekcja krawędzi okręgu, którego chcemy znaleźć znacznie zwiększa szansę jego znalezienia dlatego, że ta metoda polega na "wysyłaniu" wartości tej krawędzi jako okrąg - zatem im większa wartość tej krawędzi tym lepsza szansa, że maksymalny punkt przestrzeni Hough'a będzie środkiem tego okręgu.

Wygładzenie krawędzi (rozdział 4.1.1) pomaga znaleźć i rozesłać prawdziwy szczyt tej krawędzi co usprawnia przybliżenie się do prawdziwego punktu środka szukanego okręgu.

Natomiast odpowiedni dobór zakresu szukanych promieni zapewnia, że inna część znalezionych krawędzi, która ma większą wartość, nie wpłynie negatywnie na ostateczny wynik. Przykładowe sytuacje i wyniki tej metody przedstawiono poniżej.

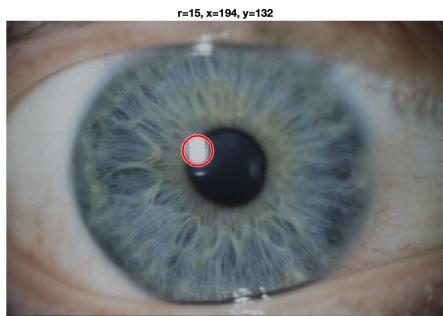


Figure 25. Nieprawidłowe zlokalizowanie żrenicy dla zbyt dużego zakresu $r \in \langle 15; 70 \rangle$ (metoda akumulacji wartości krawędzi)

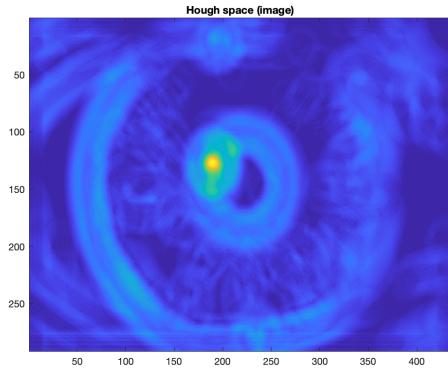


Figure 26. Przestrzeń Hough'a dla nieprawidłowo zlokalizowanej żrenicy (2D)(metoda akumulacji wartości krawędzi)

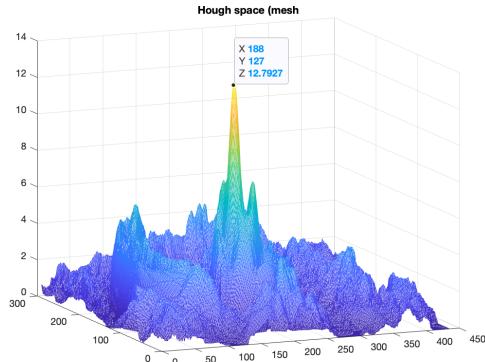


Figure 27. Przestrzeń Hough'a dla nieprawidłowo zlokalizowanej żrenicy (3D)(metoda akumulacji wartości krawędzi)

Na powyższych rysunkach (25, 26 i 27) przedstawiono sytuację, w której transformata Hough'a znalazła środek żrenicy w niepozdanym miejscu, spowodowane przez wyłapanie i "wysyłanie" wysokiej wartości jaka znajduje się w plamce odbicia światła kamery w lewym górnym rogu żrenicy (widocznej na rysunku 25) - to spowodowało wysoki wzrost wartości przestrzeni Hough'a w punkcie $x = 163$ i $y = 160$ dla $r = 100$ i zgodnie z algorytmem ten punkt, posiadający maksymalną wartość spośród wszystkich płaszczyzn z innymi wartościami r , został wybrany jako środek żrenicy. Dla porównania poniżej przedstawiono wartości tych okręgów oraz na rysunku 28.

- wartość krawędzi plamki - 0.91
- wartość krawędzi żrenicy - 0.47



Figure 28. Wartości krawędzi plamki oraz żrenicy na obrazie wykrytej krawędzi

Aby naprawić ten błąd należy zmniejszyć zakres r aby rozproszyć "rozsypane" wartości tej jasnej plamki. Wyniki tej operacji przedstawiono na rysunkach 29, 30 oraz 31.

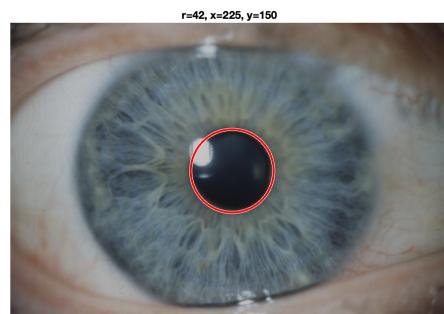


Figure 29. Prawidłowe zlokalizowanie żrenicy dla zakresu $r \in \langle 35; 70 \rangle$ (metoda akumulacji wartości krawędzi)

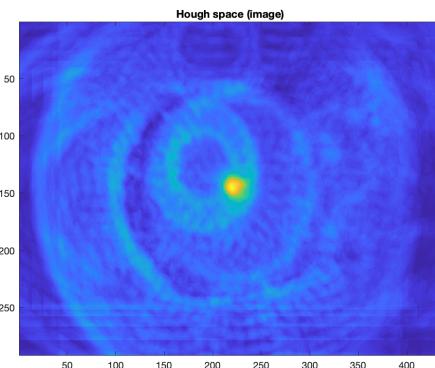


Figure 30. Przestrzeń Hough'a dla prawidłowo zlokalizowanej żrenicy (2D) (metoda akumulacji wartości krawędzi)

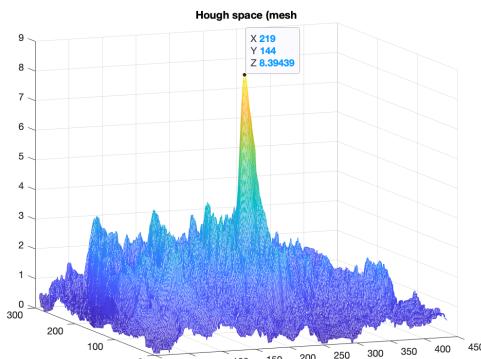


Figure 31. Przestrzeń Hough'a dla prawidłowo zlokalizowanej żrenicy (3D) (metoda akumulacji wartości krawędzi)

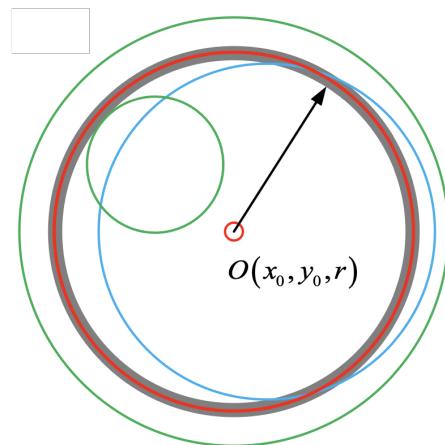


Figure 32. Okrąg najbardziej pasujący do istniejącego posiada największą wartość sumy [7]

4.3.1. Przykładowa implementacja

W kodzie poniżej ukazano przykładową implementację operatora różniczkowo-całkowego Daugmana.

```

1 V = G([sigma],x,y).*I(x,y)
2 for all x in (x_min..x_max)
3   for all y (y_min..y_max)
4     for all r in (r_min..r_max)
5       for all theta in (0..2pi|dpi)
6         A(x,y,r) = A(x,y,r)+ V(x+r*cos(theta),y+r*sin(theta))
7       end
8     end
9   end
10 end
11 [x0,y0,r0] = Find_Ind_of_Max(A)

```

Code 3. Operator różniczkowo-całkowy Daugmana - pseudokod [7]

Wyniki implementacji tego pseudokodu w Matlabie przedstawiono na rysunkach 33 i 34. Odpowiedni kod jest udostępniony w rozdziałach 12.2.7.

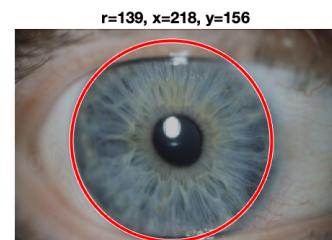


Figure 33. Wynik implementacji operatora Daugmana (oko 2a)

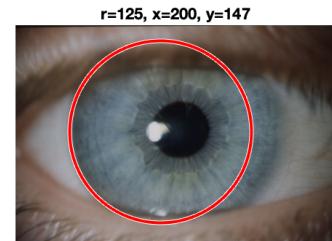


Figure 34. Wynik implementacji operatora Daugmana (oko 4a)

Teraz transformacja Hough'a znajduje prawidłowy środek żrenicy.

4.3. Operator różniczkowo-całkowy Daugmana

Inną metodą do znajdywania okręgów to operator różniczkowo-całkowy Daugmana. Na początku Daugman rozpisał swój pomysł na znalezienie tęczówki i żrenicy wzorem 4.

$$V(x, y) = \max_{(x_0, y_0, r)} \left| G_\sigma(r) * \frac{\partial}{\partial r} \oint_{(x_0, y_0, r)} \frac{I(x, y)}{2\pi r} ds \right| \quad (4)$$

gdzie

- $I(x, y)$ - obraz oka
- $G(r)$ - filtr Gaussa
- x_0, y_0 - współrzędne punktu środka (parametry)
- r - promień szukanego okręgu (parametr)
- ds - krok kątowy

Który ogólnie polega na całkowaniu po krzywej okrężnej obrazu oka w danym punkcie i danym promieniu z danym krokiem kątowym i następnie obliczenie pochodnej tej całki w celu otrzymania dwóch szczytów (peaków), które by ukazywały granicę pomiędzy żrenicą i tęczówką oraz pomiędzy tęczówką i białkiem oka. Z tych wszystkich obliczonych x_0, y_0 i r ten z największą wartością wskazuje na okrąg tęczówki lub żrenicy. Był to dobry pomysł lecz istniał jeden problem, mianowicie ta metoda działała jedynie dla sztucznych przykładów oczu (jak można zobaczyć na rysunku 10a) natomiast w realistycznych przypadkach, gdzie często w obraz wchodziły takie przeszkody jak rzęsy, powieki, itp. to nie działało to dobrze.

Dlatego Daugman postanowił zamienić kolejność różniczkowania i całkowania. W ten sposób najpierw jest liczona pochodna obrazu oka (innymi słowy, jest przeprowadzana detekcja krawędzi) i następnie jest liczona całka po różnych x_0, y_0 i r (reszta tak samo). To, jak się okazało, znacznie podniosło efektywność tej metody. Graficzne przedstawienie ukazano na rysunku 32.

Wynik tej metody w większości przypadków działa (jak na rysunku 33), natomiast zdarzają się takie zdjęcia jak na rysunku 34, gdzie ta metoda jest nie efektywna. Dzieje się tak dlatego, że kontrast pomiędzy lewą a prawą stroną oka sprawia, że obliczana całka jest przyciągana do lewej strony, przez co znaleziony okrąg jest mniejszy niż ten rzeczywisty. Ponadto krawędzie, które zostały znalezione, nie są wąskie, co pozwalałoby na idealne dopasowanie okręgów - są one grubsze i dlatego inne okręgi mogą się zmieścić podczas obliczeń tą metodą.

Niestety zabrakło czasu, żeby zaimplementować kod, który by naprawił tę sytuację (przykładowe rozwiązanie podano w rozdziale 4.5) - dlatego zdecydowano się użyć w procesie rozwoju systemu metodę kołowej transformacji Hough'a, która ma znacznie mniej błędów i która została bardziej rozwinięta.

4.4. Omijanie przeszkód podczas znajdywania okręgów

Zarówno w metodzie Hough'a jak i Daugmana przeszkodą w znajdywaniu okręgów potrafią być różne fragmenty oka, których nie chcemy wyłapać. Mogą być to rzęsy, przymknięte powieki, itp. Jednym sposobem do omijenia takich przeszkód jest wybranie powierzchni tęczówki, która omija te miejsca gdzie te przeszkody występują.

Ten sposób zaimplementowano w metodzie Hough'a dla poszukiwania okręgu tęczówki (rozdział 12.2.4) w taki sposób, że podczas "rozsyłania" okręgów wybrano takie kąty Θ , żeby omijały górną i dolną część oka (tam gdzie zazwyczaj występują powieki/rzęsy). Jako zakres Θ wybrano:

- $\Theta \in \langle -45^\circ; 60^\circ \rangle$ dla prawej strony oka
- $\Theta \in \langle 120^\circ; 225^\circ \rangle$ dla lewej strony oka

gdzie 0° to prosta w kierunku dodatnich wartości osi x . Te wartości wybrano dlatego, że zazwyczaj górne powieki bardziej zachodzą na oko, niż te niższe, dlatego wzięto mniejsze pole dla górnej części oka.

Natomiast pominięto tą metodę w kodzie do szukania okręgu żrenicy, ponieważ tam powieki nie sięgają.

4.5. Nierównomierne oświetlenie

Jak wspomniano w rozdziale 2.2 zdjęcia z bazy OFTA mają wyraźnie bardziej oświetloną lewą część oka - to może wpływać na to jak algorytm znajdzie krawędź, znajdzie okrąg, itp. generalnie lewa strona będzie bardziej preferowana, ze względu na to, że jej wartości będą większe od prawej strony.

Aby zrównoważyć szansę dla znalezienia odpowiedniego okręgu można pomnożyć prawą część zdjęcia oka przez pewien współczynnik $\alpha > 1$, który by mógł wyważić te dwie strony. Proces wybrania odpowiedniej wartości współczynnika prawdopodobnie byłby procesem empirycznym, żeby sprawdzić jak segmentacja obrazu oka by się zachowywała w poszczególnych sytuacjach dla danych wartości. Innym sposobem mogło by być sprawdzenie średniej wartości prawej i lewej strony lub podzielenie tych stron na bloki i obliczenie ich średnich oraz porównanie ze sobą.

W tym projekcie jednak nie stało się czasu, żeby tą funkcję zaimplementować.

5. Rozwijanie Tęczówki

W momencie kiedy obraz oka zostanie posegmentowany na okręgi wyznaczające tęczówkę oraz żrenicę należy przejść do następnego kroku jakim jest wyekstrahowanie tęczówki i przemienienie z postaci okrągłej do postaci prostokątnej.

Najprostszym sposobem to pobranie pewnej ilości próbek po okręgu $r_i - r_p$ gdzie

- $r_p \rightarrow$ promień żrenicy
- $r_i \rightarrow$ promień tęczówki

(którego przedłużenie przechodzi przez środek tęczówki) po okręgu od 0 do 2π . Jednak takie podejście jest nieefektywne ze względu na

to, że położenie środka tęczówki i żrenicy często jest różne - wtedy takie okręgi są niecentryczne i rozwinięty obraz nie posiada całego, rzeczywistej tęczówki oka. Albo nie działa również w sytuacji gdzie tęczówka i/lub żrenica nie ma kształtu okręgu tylko bardziej owalny. Reprezentację takich występowanów podano na rysunkach 35a, 35b i 35c.

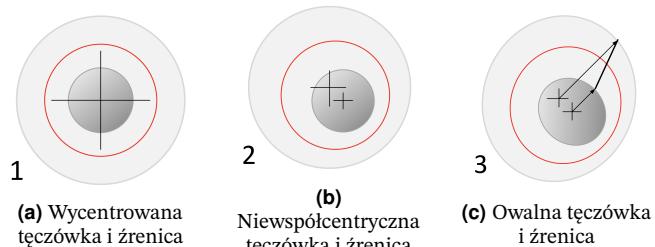


Figure 35. Przykłady występowania tęczówki oraz żrenicy na oku

Rozwiązanie tego problemu podano w rozdziale 5.1.

5.1. Normalizacja obrazu tęczówki - "rubber sheet model" Daugmana

Model "rubber sheet" Daugmana to technika normalizacji obrazu tęczówki, która umożliwia porównywanie tęczówek pomimo ich zniekształceń. Zniekształcenia te mogą wynikać z niecentryczności granic tęczówki lub ich innej formy niż okręgi. Model ten pozwala na przekształcenie tęczówki w sposób, który minimalizuje błędy wynikające z przesunięć i różnic w skali obrazów, ale jest wrażliwy na błędy związane z rotacją obrazów. Ponadto umożliwia porównywanie obrazów tęczówek ze sobą, ponieważ zawsze zwraca je o takim samym rozmiarze dzięki znormalizowanym wymiarom.

Zdefiniowany jest tymi wzorami:

$$x(r, \Theta) = (1 - r)x_p(\Theta) + rx_i(\Theta) \quad (5)$$

$$y(r, \Theta) = (1 - r)y_p(\Theta) + ry_i(\Theta) \quad (6)$$

gdzie

- $x_p(\Theta)$ i $y_p(\Theta)$ są współrzednymi punktów na granicy żrenicy
- $x_i(\Theta)$ i $y_i(\Theta)$ są współrzednymi punktów na granicy tęczówki
- r jest znormalizowanym promieniem $r \in [0, 1]$,
- Θ jest kątem $\Theta \in [0, 2\pi]$

Wzory 5 i 6 wskazują na przemianę poszczególnych próbek tęczówki ze współrzędnych polarnych (r, Θ) do kartezjańskich (x, y) posługując się znormalizowanym promieniem od granicy żrenicy do granicy tęczówki. Na rysunkach 36a oraz 36b przedstawiono ideę działania tego algorytmu.

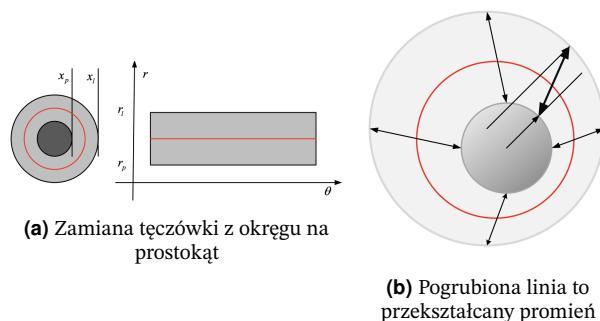


Figure 36. Graficzne przedstawienie idei modelu "rubber sheet" Daugmana

Zaimplementowany kod udostępniono w rozdziale 12.2.8. Ten kod (podobnie jak wzory 5 i 6) przyjmuje na wejściu obraz oka i zwraca znormalizowany obraz tęczówki o wymiarach 100 x 360. Dodatkowo założono liniową zmianę rozmiaru w kierunku radialnym tęczówki,

co nie do końca jest zgodne z rzeczywistością, ponieważ tęczówka jest mięśniem, który się kurczy i rozkurcza - przez to różne zdjęcia oka tej samej osoby mogą różnić się rozmiarem tęczówki. Zabrakło czasu aby rozwiązać ten problem w kodzie.

Wynik implementacji kodu przedstawiono na rysunkach 37a i 37b.

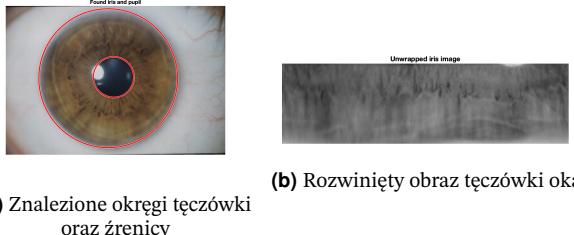


Figure 37. Przedstawienie efektów rozwijania tęczówki metodą "rubber sheet"

Pomimo lekkiego przesunięcia żrenicy względem tęczówki w oku na rysunku 37a algorytm poprawnie rozwiniął obraz tęczówki - nie ma śladów czarnej żrenicy na rozwiniętym obrazie.

6. Ekstrakcja i Selekcja Cech - Kodowanie Wzoru Tęczówki

Posiadając rozwinięty obraz tęczówki można przystąpić do wyekstrahowania cech z tęczówki do tego, żeby ją zakodować. Do tego wykorzystano schemat Daugmana oraz filtr Gabora.

6.1. Uśrednianie w paski

Zanim zakodowano obraz tęczówki najpierw zdecydowano się uśrednić ten obraz na 8 poziomych pasków - tak, aby otrzymać 8 jednowymiarowych funkcji. Podjęto taką decyzję ze względu na to, że wtedy można skorzystać z filtracji jednowymiarowej (szerzej opisane w rozdziale 6.3) oraz jest po prostu mniej obliczeń do wykonania co przyspiesza cały ten proces.

W tym celu najpierw podzielono wysokość obrazu na 8 pasków, sfiltrowano każdy pasek wzdłuż osi Y filtrem Gaussa i obliczono średnią dla każdej kolumny - w ten sposób otrzymano obraz o rozmiarze 8 x 360. Wyniki kodu przedstawiono na rysunkach 38 i 39.

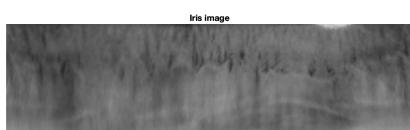


Figure 38. Oryginalny obraz tęczówki

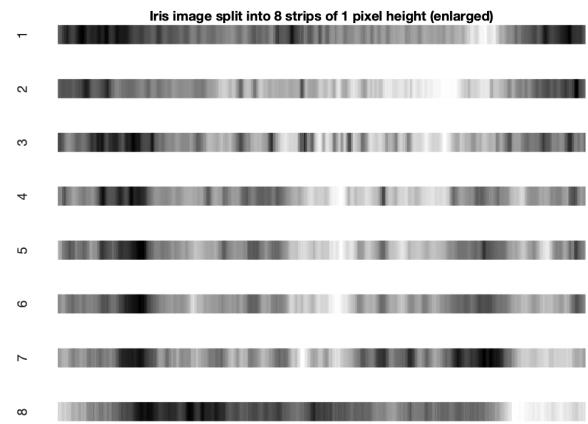


Figure 39. Podzielenie obrazu tęczówki na 8 pasków (powiększony obraz)

Poniżej, na rysunku 40, przedstawiono sam wynik procesu wygładzania paska filtrem Gaussa jeszcze przed uśrednianiem. Można zauważyć, że jeden pasek miał około 13 pikseli wysokości i 360 pikseli długości.

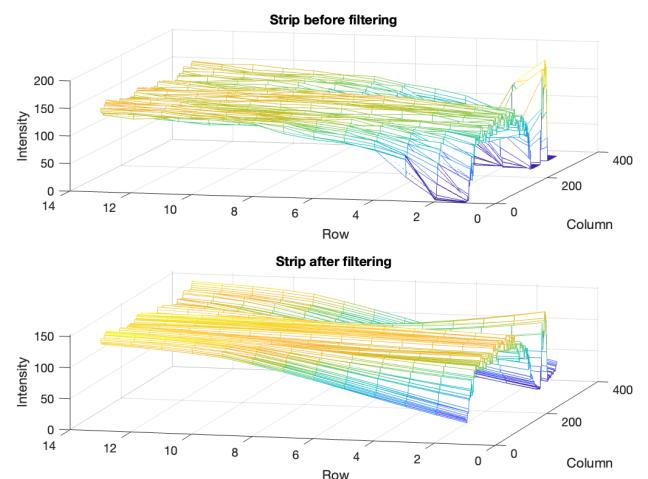


Figure 40. Pasek obrazu tęczówki przed i po filtracji Gausem

6.2. Schemat Daugmana

Schemat Daugmana to metoda kodowania wzorów tęczówki, która jest opisany ogólnym wzorem 7. Podstawowym krokiem w algorytmie Daugmana jest wyznaczenie wartości (a dokładniej znaku) tego wzoru dla części rzeczywistej i urojonej (wzory 8 i 9). W taki sposób obraz jest kodowany binarnie.

$$h = h_R + jh_I = \int_{\rho} \int_{\varphi} I(\rho, \varphi) e^{\frac{-(r_0-\rho)^2}{\alpha^2}} e^{\frac{-(\Theta-\varphi)^2}{\beta^2}} e^{-j\omega(\Theta_0-\varphi)} \rho d\rho d\varphi \quad (7)$$

gdzie

- $r_0 \in [r_p = 0, r_i = 1]$
- $\Theta \in [0, 2\pi)$
- α, β - odchylenia standardowe

$$\begin{aligned} h_R = 1 & \text{ if } \operatorname{Re} \left(\int_{\rho} \int_{\varphi} I(\rho, \varphi) e^{\frac{-(r_0-\rho)^2}{\alpha^2}} e^{\frac{-(\Theta-\varphi)^2}{\beta^2}} e^{-j\omega(\Theta_0-\varphi)} \rho d\rho d\varphi \right) \geq 0 \\ h_R = 0 & \text{ if } \operatorname{Re} \left(\int_{\rho} \int_{\varphi} I(\rho, \varphi) e^{\frac{-(r_0-\rho)^2}{\alpha^2}} e^{\frac{-(\Theta-\varphi)^2}{\beta^2}} e^{-j\omega(\Theta_0-\varphi)} \rho d\rho d\varphi \right) < 0 \end{aligned} \quad (8)$$

$$\begin{aligned} h_I = 1 & \text{ if } \operatorname{Im} \left(\int_{\rho} \int_{\varphi} I(\rho, \varphi) e^{\frac{-(r_0-\rho)^2}{\alpha^2}} e^{\frac{-(\Theta-\varphi)^2}{\beta^2}} e^{-j\omega(\Theta_0-\varphi)} \rho d\rho d\varphi \right) \geq 0 \\ h_I = 0 & \text{ if } \operatorname{Im} \left(\int_{\rho} \int_{\varphi} I(\rho, \varphi) e^{\frac{-(r_0-\rho)^2}{\alpha^2}} e^{\frac{-(\Theta-\varphi)^2}{\beta^2}} e^{-j\omega(\Theta_0-\varphi)} \rho d\rho d\varphi \right) < 0 \end{aligned} \quad (9)$$

Graficzne przedstawienie idei ukazano na rysunku 41.

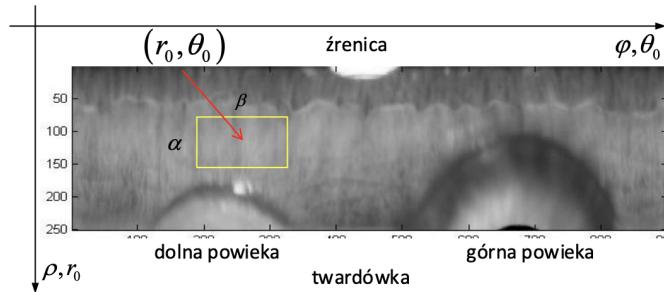


Figure 41. Graficzne przedstawienie schematu Daugmana

Jednak po pewnych przekształceniach wzoru 7 (szerzej opisanych w źródłach [7] i [2]) można dojść do stwierdzenia, że ten wzór realizuje pewną transformację przestrzenno-częstotliwościową - inaczej, zachowuje się jak filtr Gabora.

6.3. Filtr Gabora

Filtr Gabora jest jednym z narzędzi używanych w przetwarzaniu obrazów oraz ekstrakcji cech. Jest to filtr liniowy, który stosuje się do analizy lokalnej częstotliwości i orientacji struktur w obrazach. Filtry Gabora są bardzo skuteczne w wykrywaniu krawędzi oraz tekstur, co czyni je skutecznym narzędziem do analizy obrazów tęczówki.

Filtr Gabora można opisać funkcją falową w dziedzinie czasowej oraz częstotliwościowej. Ogólnie rzecz biorąc, funkcja Gabora jest funkcją Gaussa modulowaną falą sinusoidalną. Wzór na filtr Gabora można podzielić na [3] [1]:

- część rzeczywistą:

$$g(x, y; \lambda, \Theta, \psi, \sigma, \gamma) = e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}} \cos\left(2\pi \frac{x'}{\gamma} + \psi\right) \quad (10)$$

- część urojoną:

$$g(x, y; \lambda, \Theta, \psi, \sigma, \gamma) = e^{-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}} \sin\left(2\pi \frac{x'}{\gamma} + \psi\right) \quad (11)$$

gdzie

- $x' = x \cos \Theta + y \sin \Theta$
- $y' = -x \sin \Theta + y \cos \Theta$
- λ - długość fali (częstotliwość przestrzenna filtru)
- Θ - orientacja filtru
- ψ - faza przesunięcia fali sinusoidalnej
- σ - odchylenie standardowe gaussowskiego okna
- γ - współczynnik aspektu (kontroluje eliptyczność)

Przechodząc z filtrem Gabora do problemu zakodowania tęczówki jasnym staje się, że należy znaleźć te zmiany i cechy w tęczówce jakimi są jej pionowe rzęski - to jest ta cecha biometryczna, która jest unikalna dla każdej osoby. Zatem zadaniem filtru Gabora będzie wyekstrahowanie tą informację.

6.4. Ustalenie parametrów filtru Gabora

Najważniejszymi parametrami do ustalenia dla potrzeby zakodowania tęczówki to: λ , Θ i σ . Do wizualizacji filtrów Gabora w kolejnych podrozdziałach użyto rozmiar filtru 51.

6.4.1. Długość fali - λ

Parametr długości fali (λ) decyduje o tym jakie zmiany wyłapuje filtr Gabora - a konkretniej jakie częstotliwości są te zmiany. Inaczej - im większa lambda tym większe "fale" filtru. Relację pomiędzy λ a częstotliwością zmian cechy obrazu można przedstawić następująco:

$$\lambda = \frac{1}{f} \quad (12)$$

gdzie

- f - częstotliwość zmian cechy na obrazie

Zatem jeśli chcemy znaleźć cechy, które zmieniają się bardzo szybko to potrzebna jest mała wartość długości fali - i vice versa. Poniżej, na rysunkach 42, 43, 44 oraz 45 przedstawiono wykres jednowymiarowego filtru Gabora dla różnych parametrów λ i dla niezmiennych parametrów:

- $\Theta = 0^\circ$
- $\psi = 0$
- $\sigma = 10$
- $\gamma = 1$

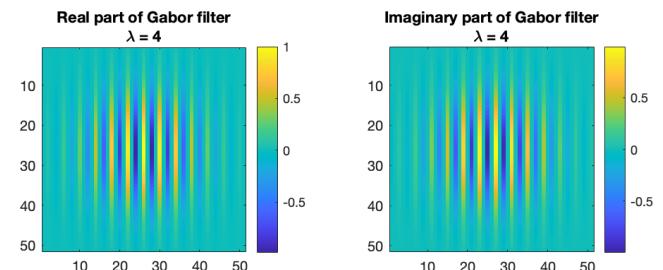


Figure 42. Filtr Gabora - $\lambda = 4$

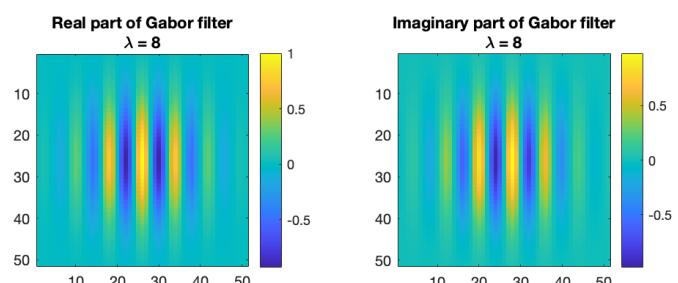
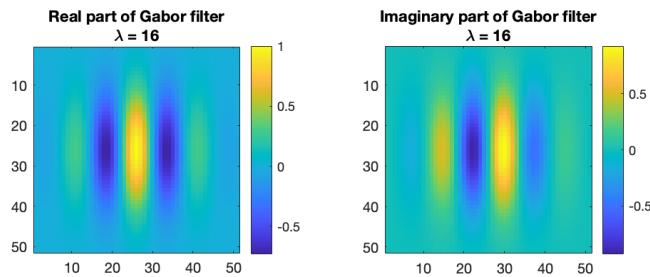
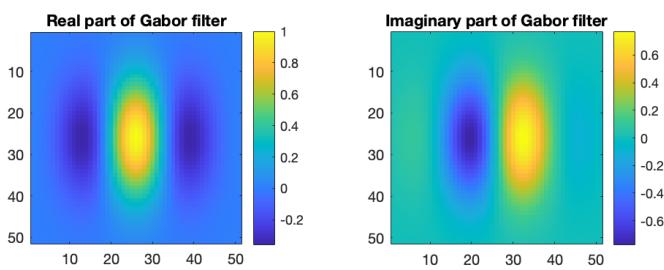


Figure 43. Filtr Gabora - $\lambda = 8$

Figure 44. Filtr Gabora - $\lambda = 16$ Figure 45. Filtr Gabora - $\lambda = 32$

Można zauważyc, że wraz ze zwiększeniem λ filtr ma coraz grubsze fale, co pozwala na wyłapanie coraz wolniejszych zmian.

Spoglądając na sam obraz tęczówki trudno stwierdzić jak często zmieniają się pionowe rzęski, których szukamy, zatem zdecydowano się spróbować wyrysować widmo częstotliwościowe jednego paska, żeby zorientować się jakich z grubsza częstotliwości się spodziewać.

Aby tego dokonać skorzystano z pliku ze zdjęciem (rysunek 2a) jako odniesienie. Odczytano DPI tego zdjęcia jako 72 piksele/cal aby poprawnie obliczyć wartość częstotliwości. Widmo częstotliwościowe przedstawiono na rysunku 46.

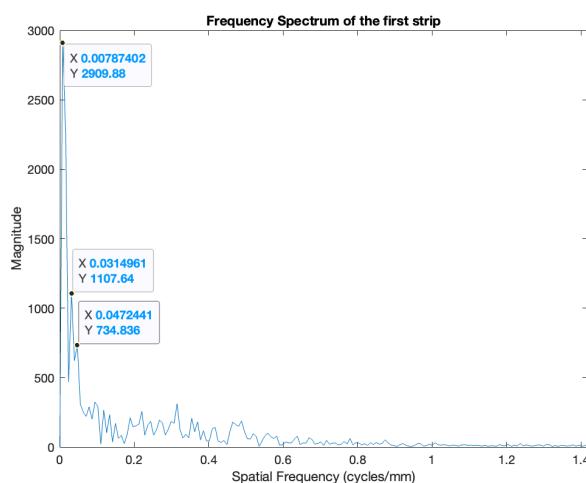


Figure 46. Widmo częstotliwościowe pierwszego paska z uśrednionego i podzielonego obrazu tęczówki

Z tego powyższego wykresu można odczytać 3 najwyższe wartości częstotliwości i zamienić je na długość fali używając wzoru 12:

- $f = 0.007 \rightarrow \lambda = \frac{1}{0.007} \approx 142$
- $f = 0.03 \rightarrow \lambda = \frac{1}{0.03} \approx 33$
- $f = 0.04 \rightarrow \lambda = \frac{1}{0.04} = 25$

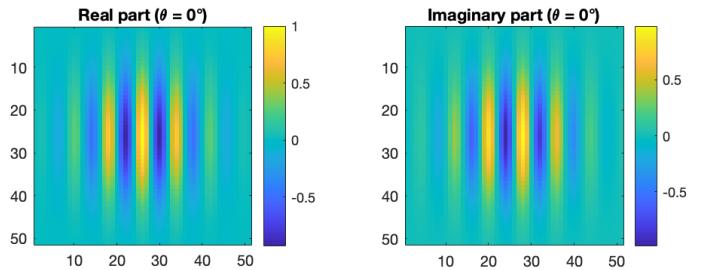
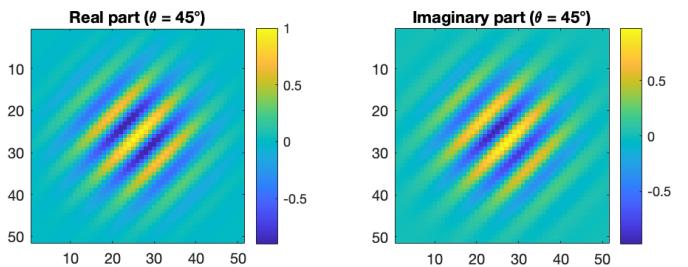
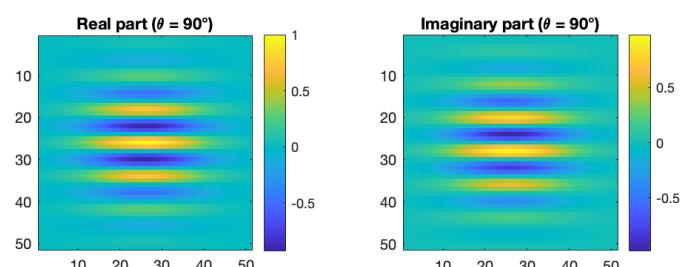
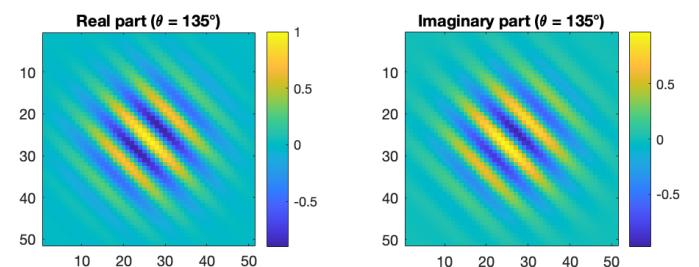
Porównanie tych wartości i ich skuteczności opisano szerzej w rozdziale 10.3.4.

6.4.2. Orientacja - Θ

Ten parametr wskazuje na to jakie ułożenie kątowe mają te cechy, których szukamy. Wizualizację filtra Gabora z różnymi wartościami Θ i niezmiennymi parametrami:

- $\lambda = 8$
- $\sigma = 10$
- $\psi = 0$
- $\gamma = 1$

przedstawiono na rysunkach 47, 48, 49 i 50.

Figure 47. Filtr Gabora - $\Theta = 0$ Figure 48. Filtr Gabora - $\Theta = 45$ Figure 49. Filtr Gabora - $\Theta = 90$ Figure 50. Filtr Gabora - $\Theta = 135$

Dla pionowych rzęs tęczówki oczywiście wybierzemy filtr zwrócony w pion, czyli dla $\Theta = 0^\circ$ (rysunek 47).

6.4.3. Odchylenie standardowe - σ

Ten parametr definiuje jaki jest rozrzut tego filtru. Może mieć bardzo małą powierzchnię albo bardzo dużą. Efekt odchylenia jest powiązany z rozmiarem kernela filtru. Na rysunkach 51, 52, 53 oraz 54 przedstawiono wizualizację filtru Gabora z różnymi wartościami σ i z niezmiennymi parametrami:

- $\lambda = 8$
- $\Theta = 0^\circ$
- $\psi = 0$
- $\gamma = 1$

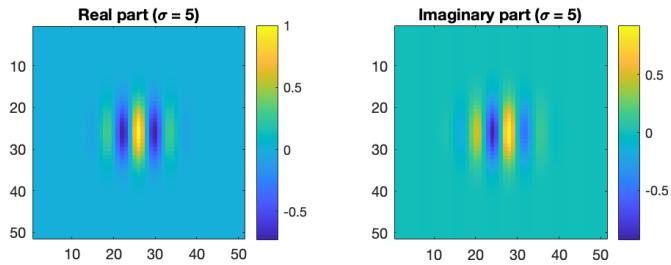


Figure 51. Filtr Gabora - $\sigma = 5$

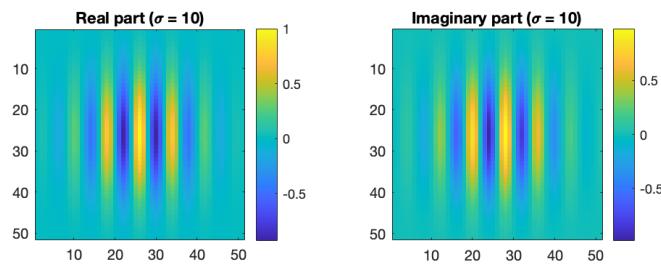


Figure 52. Filtr Gabora - $\sigma = 10$

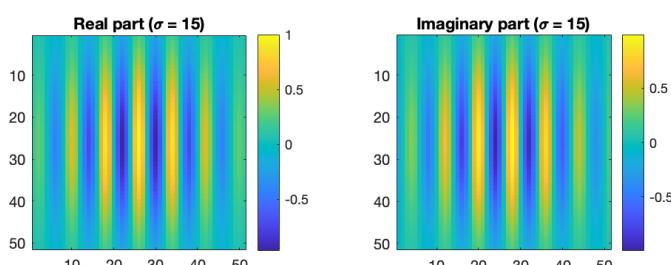


Figure 53. Filtr Gabora - $\sigma = 15$

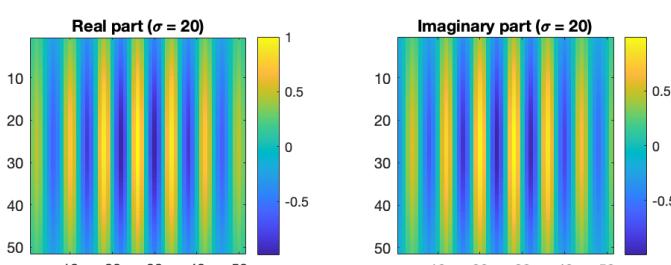


Figure 54. Filtr Gabora - $\sigma = 20$

Dla poszukiwania cech tęczówki użyto wartość $\sigma = 8$, żeby nie zawierać zbyt dużego pola i nie wyłapywać niepotrzebnych szumów.

6.5. Filtrowanie filtrem Gabora

Posiadając już wiadomość o tym jakie parametry dobierać dla filtra Gabora zaimplementowano kod, który filtruje uśredniony i podzielony obraz. Odpowiedni kod udostępniono w rozdziale 12.2.10. Poniżej, na rysunku 55, zaprezentowano skutki filtrowania filtrem Gabora.

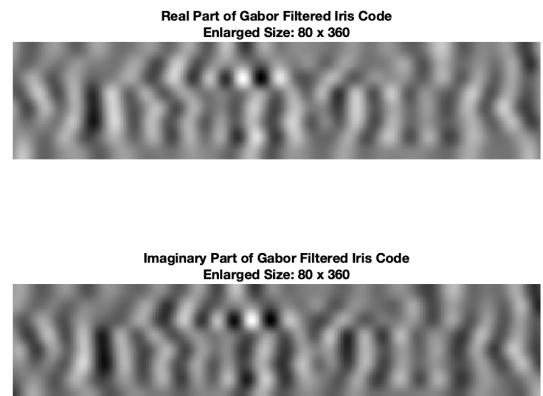


Figure 55. Obraz uśrednionej i podzielonej tęczówki po zastosowaniu filtru Gabora

6.6. Binaryzacja - zakodowanie tęczówki

Na tym etapie już następuje pełne zakodowanie tęczówki. Zastosowano wzory 8 i 9, żeby zakodować część rzeczywistą oraz urojoną zfiltrowanych obrazów tych tęczówek. Na rysunku 56 przedstawiono wyniki binaryzacji

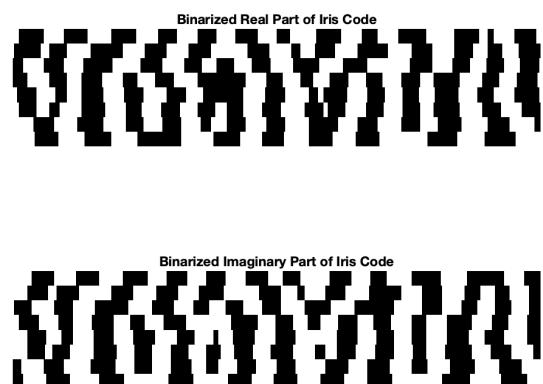


Figure 56. Zakodowana tęczówka po zastosowaniu binaryzacji na zfiltrowanym obrazie

6.7. Niepełne dwubitowe kodowanie

Podczas gdy dotychczasowa metoda kodowania korzystała z 2×1 bitu do kodowania tęczówki (0 lub 1 dla części rzeczywistej i urojonej) to istnieją inne metody do zakodowania ze swoimi słabymi i mocnymi stronami. Jedną z takich metod to kodowanie za pomocą 3 wartości:

- 0_{dec} (00_b) gdy wartości są około -1
- 1_{dec} (01_b) gdy wartości są pomiędzy -1 a 1
- 2_{dec} (10_b) gdy wartości są około 1

Wtedy te niepewne bity są zakodowane jako szare obszary i nie są brane pod uwagę albo mają mniejszą wagę na ostateczny wynik. Wynik takiej implementacji zaprezentowano na rysunku 57 [7].

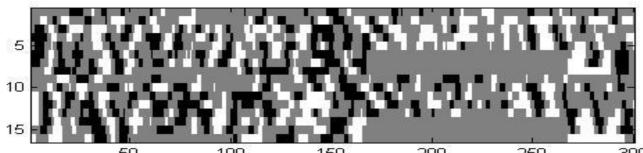


Figure 57. Zakodowana tęczówka z 3 różnymi wartościami

Niestety zabrakło czasu aby zaimplementować tą metodę i porównać ją z dotychczasowo opisaną metodą.

7. Zapis Wzorca

Na tym etapie systemu można wprowadzić funkcjonalność tworzenia wzorców i dodawaniem ich do bazy wzorców. Jednak podczas tworzenia tego projektu zabrakło czasu, żeby to wprowadzić i postanowiono, że wzorcami dla każdej osoby będą pierwsze zdjęcia ich oczu.

Często robi się tak, że tych wzorców się robi dużo aby była duża różnorodność sytuacji w jakich zdjęcia mogą być pobierane. Służy to również do tego, żeby prawidłowo stworzyć wzorzec tęczówki, np. kiedy korzysta się z metody słabych bitów.

7.1. Maska słabych i mocnych bitów

Kiedy zapisuje się wzorce tęczówek aby potem porównywać podawane tęczówki można zastosować metodę tworzenia maski słabych i mocnych bitów. Służy to do tego aby wyznaczyć te bity, które dla tego samego oka ale dla różnych zdjęć tego oka, są niestabilne (słabe bity) a które są stabilne (mocne bity). Wtedy przy porównywaniu poszczególnych kodów tęczówek bit po bicie (np. odległość Hamming co opisano szerzej w rozdziale 8.1) wtedy wartość błędu na słabym bicie nie wpływa tak bardzo na ostateczny wynik jak błędą na mocnym bicie.

8. Moduł Decyzyjny

Posiadając już możliwość zakodowania tęczówki teraz należy opracować metodę do porównywania podobności kodów między sobą. Im bardziej podobne kody do siebie tym większa szansa, że należą do tej samej osoby.

8.1. Odległość Hamminga

Najbardziej znaną metodą do porównywania kodów tęczówki to odległość Hamminga, która jest sumą różnic bitowych między dwoma kodami. Dokładniej, odległość Hamminga mierzy liczbę pozycji, na których odpowiadające sobie bity w dwóch kodach tęczówki są różne. Wyrażenie to można zapisać jako :

$$H(C_A, C_B) = \frac{\sum_{n=1}^N (\text{XOR}(C_A(n), C_B(n)) = 1)}{N} \quad (13)$$

[7] gdzie

- C_A - kod A
- C_B - kod B
- N - ilość bitów w kodach

Natomiast implementację w kodzie udostępniono w rozdziale 12.2.11.

Wyniki odległości Hamminga wskazują jak bardzo podobne są dwa kody tęczówek względem siebie:

- jeżeli wszystkie bity są takie same: $HD = 0$
- jeden kod jest negatywem drugiego: $HD = 1$

- najdalsza możliwa odległość pomiędzy kodami: $HD = 0.5$

Na podstawie tej wartości można stwierdzić jak dobrze system przetwarza zdjęcia tęczówek i je klasyfikuje (opisane szerzej w rozdziale 10).

Natomiast jest pewna słabość tego podejścia, o której opisano w rozdziale 8.2.

8.1.1. Ukazywanie różnic w kodzie

Porównując dwa kody (lub więcej) między sobą wygodnie jest ukazać różnicę pomiędzy nimi za pomocą koloru w miejscach gdzie się nie zgadzają.

Na rysunkach 58, 59 oraz 60 przedstawiono ukazanie różnic pomiędzy kodami w 3 różnych sytuacjach:

- różne osoby
- ta sama osoba, różne ujęcia oka
- ta sama osoba, to samo zdjęcie oka

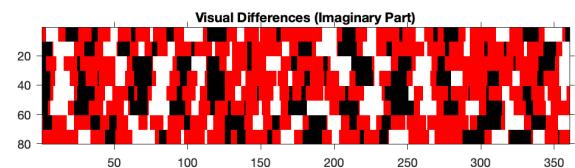
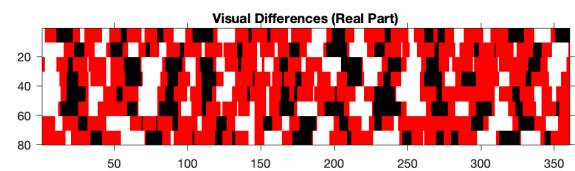


Figure 58. Różnice pomiędzy kodami tęczówek różnych osób: $HD_R = 0.49$, $HD_I = 0.51$

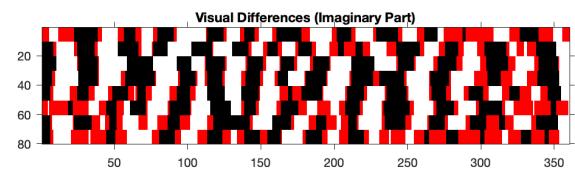
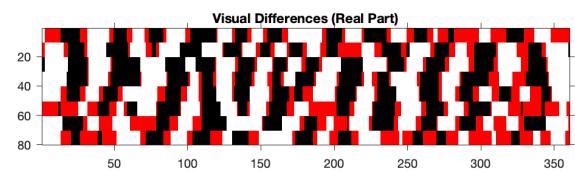


Figure 59. Różnice pomiędzy kodami tęczówek tej samej osoby, dla różnych ujęć tęczówki: $HD_R = 0.24$, $HD_I = 0.25$

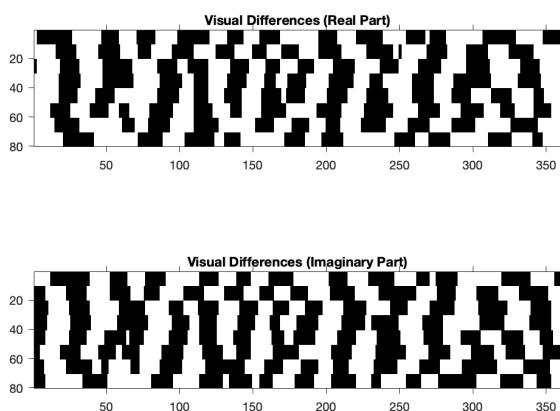


Figure 60. Różnice pomiędzy kodami tęczówek tej samej osoby, dla tego samego zdjecia tęczówki : $HD_R = 0.00$, $HD_I = 0.00$

Wyniki tych porównań wyszły zgodnie z oczekiwaniemi:

- dla kodów tęczówek różnych osób odległość Hamminga wyniosła około 0.50, co jest maksymalną wartością dla różnych tęczówek
- dla kodów tęczówek tej samej osoby, dla różnych ujęć tęczówki odległość Hamming wyniosła około 0.25
- dla kodów tęczówek tej samej osoby, dla tego samego zdjecia tęczówki odległość Hamminga wyniosła 0.00

8.2. Przesuwanie bitów

Jednak dla metody odległości Hamminga jest pewna słabość, mianowicie taka, że jeśli osoba, której jest robione zdjecie tęczówki przechyli swoją głowę nawet nieznacznie to może się okazać, że bity tego zdjecia i wzorca będą znacznie się różnić ($HD \approx 0.5$). Aby zwalczyć ten problem należy zaimplementować przesuwanie bitów zakodowanej tęczówki w lewo i prawo o pewną ilość bitów. Graficzna reprezentacja idei tej metody przedstawiona na rysunku 61. W kodzie zaimplementowano tą metodę w rozdziale 12.2.11 w liniach 7-21.

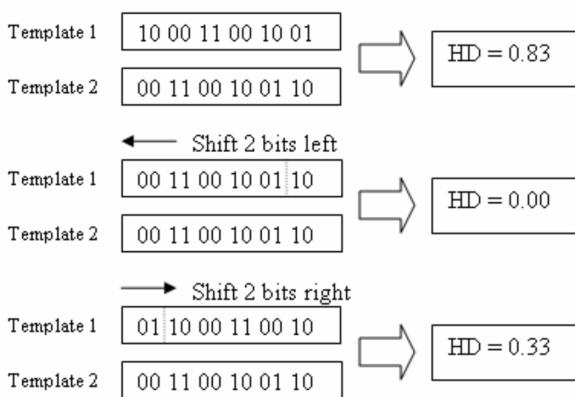


Figure 61. Idea przesuwania bitów podczas obliczania odległości Hamminga [6]

8.3. Inne metody porównywania

Inne metody do porównania podobieństwa kodów tęczówki między sobą to m.in.:

- analiza dyskryminacyjna Fischera
- globalna energia sygnału przejsc przez 0
- wazona odleglosc euklidesowa

9. Preferowane Ustawienia Systemu

Do celów porównawczych oraz do ostatecznej oceny wybrano najlepsze ustawienia systemu aby mieć wzór do których można porównywać inne metody/parametry. Poniżej przedstawiono ustawienia dla tego systemu:

- redukcja rozmiaru
- zamiana na skalę szarości
- detekcja krawędzi filtrami krawędziowymi
- wygładzanie krawędzi po jej detekcji
- detekcja okrągów za pomocą kołowej transformaty Hough'a
 - akumulacja wartością krawędzi
 - omijanie powiek podczas szukania okrągów tęczówki
 - brak omijania powiek podczas szukania okrągów żrenicy
- korzystanie z modelu "rubber sheet" przy rozwijaniu tęczówki
- uśrednianie i filtrowanie w paski
- filtrowanie filtrem Gabora
 - $\lambda = 28$
 - $\Theta = 0^\circ$
 - $\sigma = 8$
 - $\psi = 0$
 - $\gamma = 0.5$
- binaryzowanie 2 wartościami
- porównywanie odległości Hamminga
 - porównywanie z przesuwaniem bitów (4 bity w obie strony)

Dzięki temu można porównać z innymi ustawieniami parametrów czy z wykorzystaniem innych metod.

10. Ocena Jakości Systemu

Ostatnim etapem projektowania tego systemu to sprawdzenie jego jakości. Podając na wejściu bazę zebranych zdjęć opisanych w rozdziale 2.2 sprawdzana jest odległość Hamminga pomiędzy każdym zdjeciem (każdy z każdym). Na tej podstawie można stwierdzić jak system przetworzył i zklassyfikował podane zdjecia.

10.1. Macierz konfuzji

Macierz konfuzji (lub macierz błędów) pokazuje wartość Hamminga pomiędzy każdym zdjeciem za pomocą odcieni koloru (im ciemniejsza kratka tym mniejsza wartość HD). W przekątnej tej macierzy HD powinna być równa 0, ponieważ porównujemy między sobą to samo zdjecie. Powinna również być "większa" przekątna złożona z większych kratek wielkości 3x3 małych kratek, które reprezentują porównywane kody tej samej osoby - w bazie jest 6 różnych osób, więc tych większych kratek powinno też być 6. Poniżej, na rysunku 65, przedstawiono wynik w postaci macierzy dla preferowanych ustawień systemu.

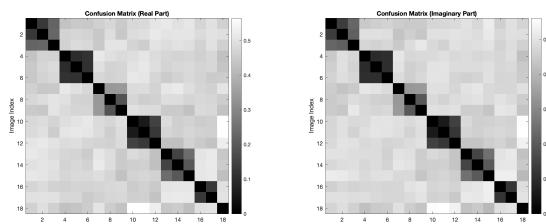


Figure 62. Macierz konfuzji dla preferowanych ustawień systemu (część rzeczywista i urojona)

Wyniki systemu są dość zadowalające. Można zauważyć na macierzach, że:

- dla części rzeczywistej oraz urojonej odległości Hamminga są podobne, lecz nie identyczne

- dla większości porównań wewnętrz tej samej klasy (osoby) odległości są małe - zdarzają się trochę większe odległości Hamminga (maksymalnie rzędu 0.33), które wskazują, że dla niektórych osób zdjęcia tej samej tęczówki sporo się różniły i system nie mógł wyłapać tych zmian ze zdjęcia zbyt skutecznie
- trzecie zdjęcie oka, należące do ostatniej (szóstej) osoby, wcale nie pasuje do reszty tęczówek

Odnosząc się do ostatniego punktu - dzieje się tak dlatego, że powieka oka na tym zdjęciu za bardzo nachodzi na oko i transformata Hough'a nie jest w stanie wyłapać odpowiednio tęczówkę i żrenicę. Dlatego automatycznie wszystkie poprzedzające kroki są nieefektywne i odległość Hamminga osiąga maksymalną wartość - 0.5. Wynik samej transformaty Hough'a dla tego oka ukazano na rysunku 63.

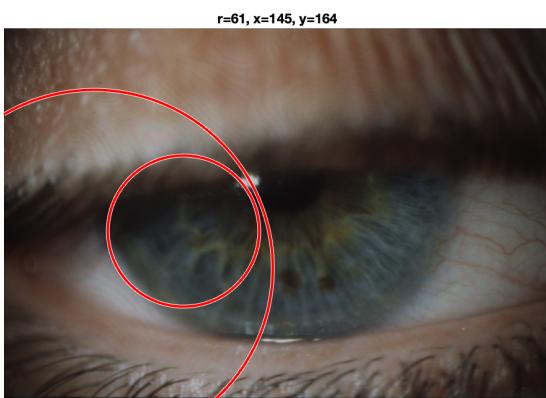


Figure 63. Nieprawidłowo znalezione okręgi tęczówki i żrenicy dla oka 7c

Zatem można stwierdzić, że w przypadku źle znalezionej tęczówki i żrenicy system zadziałał według oczekiwania.

10.2. Rozkłady wewnętrzklasowe oraz międzyklasowy

Innym sposobem do oceny jakości klasyfikacji, znacznie bardziej wyraźnym i liczbowym, jest analiza odległości rozkładów wewnętrzklasowych i międzyklasowych. Generalna idea tej metody polega na obliczeniu odległości pomiędzy rozkładami, gdzie porównuje się sytuację, w której klasyfikowane są różne ujęcia tęczówek tej samej osoby (rozkłady wewnętrzklasowe) oraz sytuację, w której klasyfikowane są tęczówki różnych osób (rozkłady międzyklasowe).

Wysoka wartość odległości (d) oznacza, że odległości Hamminga międzyklasowe są znacznie większe niż wewnętrzklasowe, co wskazuje na dobrą separację klas, co jest pożądanym wynikiem w klasyfikacji. Odległość pomiędzy tymi rozkładami oblicza się za pomocą wzoru:

$$d = \frac{\mu_1 + \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2}} \quad (14)$$

gdzie

- μ - wartość średnia w rozkładach wewnętrzklasowych i międzyklasowych
- σ - odchylenia standardowe tych rozkładów.

Graficzne przedstawienie dwóch rozkładów przyrównanych za pomocą histogramów na rysunku 64.

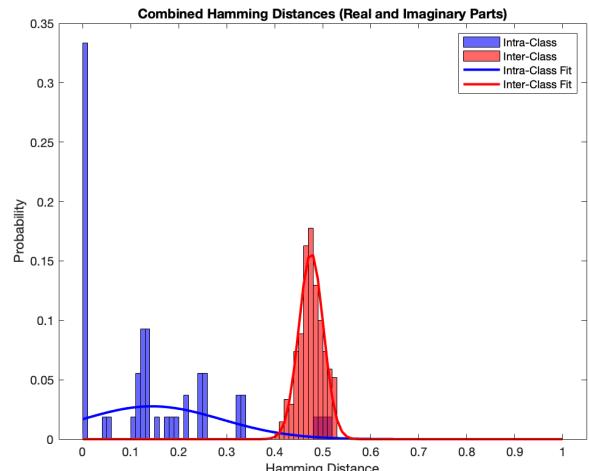


Figure 64. Rozkłady wewnętrz i międzyklasowe

10.2.1. Odległość d dla preferowanego ustawienia systemu

Obliczono tą odległość dla preferowanych ustawień systemu i wyniosła ona:

- dla części rzeczywistej: $d = 2.0492$
- dla części urojonej: $d = 2.0608$

10.3. Porównanie różnych ustawień systemu

Istnieje wiele różnych kombinacji związanych z omówionymi metodami/parametrami dla tego systemu. Poniżej przedstawiono macierze konfuzji oraz odległości pomiędzy rozkładami wewnętrzklasowymi i międzyklasowymi dla wybranych ustawień systemu (wszystkie nie wspomniane metody/parametry są takie same jak dla preferowanych).

10.3.1. Preferowane ustawienia

Ustawienia preferowane jak opisane w rozdziale 9

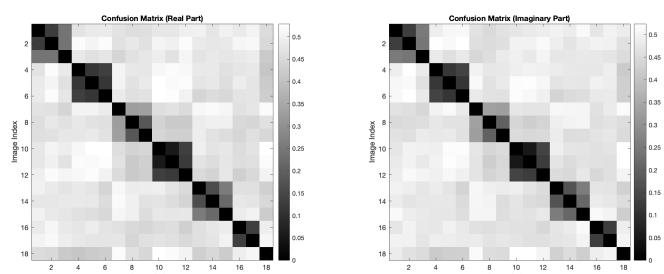


Figure 65. Macierz konfuzji dla preferowanych ustawień systemu (część rzeczywista i urojona)

- $d_R = 2.0051$
- $d_I = 2.0401$

10.3.2. Bez wygładzania obrazu przed detekcją krawędzi

Jak opisano w rozdziale 4.1.1, w tym podejściu nie przepuszczono obrazu przez filtr Gaussa przed detekcją krawędzi.

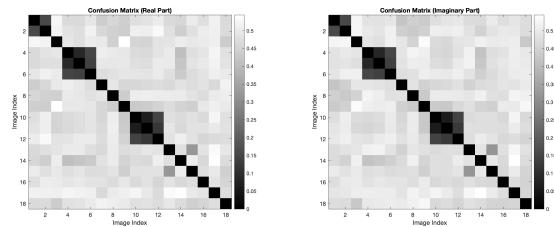


Figure 66. Macierz konfuzji dla systemu bez wygładzania obrazu przed detekcją krawędzi

- $d_R = 0.8312$
- $d_I = 0.8626$

Jak można zauważyc, brak wygładzania obrazu dla niektórych przypadków okazał dewastujący – najprawdopodobniej przy wykrywaniu tęczówek za pomocą transformaty Hough'a.

10.3.3. Akumulacja wartością 1

Jak opisano w rozdziale 4.2.2 można różnie podchodzić do działania kołowej transformaty Hough'a. Tutaj przetestowano akumulację jedynkami. Na rysunku 67 przedstawiono wyniki jakości systemu dla tego ustawienia.

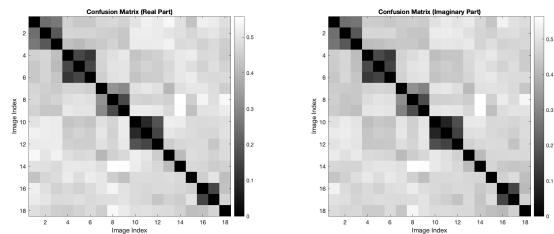


Figure 67. Macierz konfuzji dla systemu z metodą akumulacji jedynkami w transformacie Hough'a

- $d_R = 1.5807$
- $d_I = 1.6124$

Można zatem wywnioskować, że ta metoda akumulacji jest gorsza dla tego systemu od akumulacji wartością krawędzi.

10.3.4. Różne wartości parametru długości fali filtru Gabora

W tej próbie przetestowano wszystkie parametry długości fali λ dla filtru Gabora, które wyznaczono w rozdziale 6.4.1. Wyznaczono jakość klasyfikacji dla:

- $\lambda = 25$: rysunek 68
- $\lambda = 33$: rysunek 69
- $\lambda = 142$: rysunek 70

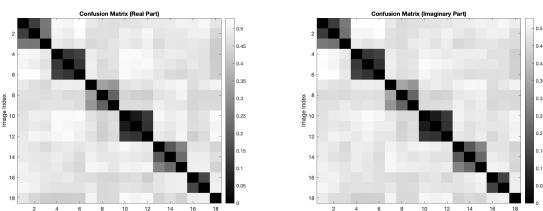


Figure 68. Macierz konfuzji dla filtru Gabora: $\lambda = 25$

- $d_R = 2.0433$
- $d_I = 2.0568$

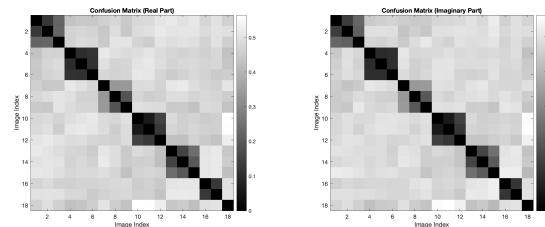


Figure 69. Macierz konfuzji dla filtru Gabora: $\lambda = 33$

- $d_R = 2.0492$
- $d_I = 2.0608$

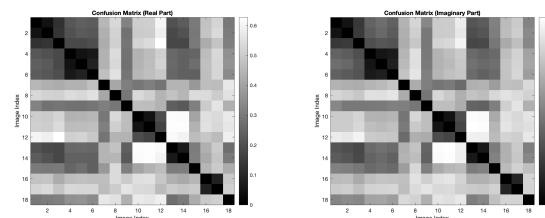


Figure 70. Macierz konfuzji dla filtru Gabora: $\lambda = 142$

- $d_R = 1.0526$
- $d_I = 1.0611$

Jak można zauważyc najlepsze wyniki miał system dla filtru Gabora z $\lambda = 33$.

10.4. Inne metody oceny jakości

Inne wartości, które podają informację na temat jakości działania tego systemu są następujące:

- False Acceptance Rate (FAR) - wskaźnik fałszywych akceptacji dzieje się wtedy, kiedy system źle sklasyfikował podane zdjęcie tęczówki jako należące do istniejącej tęczówki w bazie, podczas gdy ta tęczówka powinna zostać odrzucona.
- False Rejection Rate (FRR) - współczynnik fałszywych odrzuceń wskazuje na liczbę razy kiedy system źle sklasyfikował podane zdjęcie tęczówki jako nienależące do istniejącej tęczówki w bazie, podczas gdy ta tęczówka powinna zostać zaakceptowana
- Correct Acceptance Rate (CAR) - współczynnik prawidłowych akceptacji
- Correct Rejection Rate (CRR) - współczynnik prawidłowych odrzuceń
- Total Error Rate (TER) - współczynnik wszystkich błędów razem wziętych: $TER = FAR + FRR$

Niestety zabrakło czasu, żeby zaimplementować te oceny jakości.

11. Wnioski

System rozpoznawania i identyfikacji tęczówek sprawuje się dość dobrze. Jego metody dojścia do decyzji zostały w sposób pracowity rozwinięte i zoptymalizowane na ile czas pozwolił. Sprawdzona jakość tego systemu jest na poziomie zadowalającym jak na organiczne moźliwości rozwoju tego programu.

Oczywiście w kontekście dzisiejszych czasów i systemów bezpieczeństwa ten system jest daleki od idealnego zatem dalszy rozwój jest kolejnym, sensownym krokiem do tego, żeby udoskonalić ten system i zmniejszyć jego współczynnik błędu (tym samym zwiększając jego bezpieczeństwo) oraz zoptymalizować go dla większego zbioru danych.

12. Kod Programu

12.1. Plik main

```

1 close all;
2 clear all;
3
4 % CONSTANTS
5 resolution_ratio = 0.5;
6
7 filter_len = 4;
8 rgb2gray = true;
9 sigma = 4;
10 filter_ratio = 7;
11
12 R_min = 140;
13 R_max = 170;
14 iris_threshold = 0.1;
15
16 r_min = 35;
17 r_max = 70;
18 pupil_threshold = 0.1;
19
20
21 % Iris Detection & Recognition
22
23 % READ AND PREPROCESS IMAGES
24 img_1 = imread("my-iris-database/person1/o_sr11.bmp");
25 img_dec_1 = decrease_resolution(img_1,
26     resolution_ratio);
27 [img_edges_1, shift_1] = detect_edges(img_dec_1,
28     filter_len, rgb2gray, sigma, filter_ratio);
29
30 img_2 = imread("my-iris-database/person1/o_sr13.bmp");
31 img_dec_2 = decrease_resolution(img_2,
32     resolution_ratio);
33 [img_edges_2, shift_2] = detect_edges(img_dec_2,
34     filter_len, rgb2gray, sigma, filter_ratio);
35
36
37 % FIND IRIS CIRCLE
38 [iris_center_1, iris_radius_1] = find_iris(
39     img_edges_1, shift_1, R_min, R_max,
40     iris_threshold);
41 [iris_center_2, iris_radius_2] = find_iris(
42     img_edges_2, shift_2, R_min, R_max,
43     iris_threshold);
44
45 % FIND PUPIL CIRCLE
46 [pupil_center_1, pupil_radius_1] = find_pupil(
47     img_edges_1, shift_1, r_min, r_max,
48     pupil_threshold);
49 [pupil_center_2, pupil_radius_2] = find_pupil(
50     img_edges_2, shift_2, r_min, r_max,
51     pupil_threshold);
52
53 show_I_P(img_dec_1, iris_center_1,
54     pupil_center_1, iris_radius_1,
55     pupil_radius_1, img_dec_2, iris_center_2,
56     pupil_center_2, iris_radius_2,
57     pupil_radius_2);
58
59 % UNWRAP IRIS
60 unwrapped_iris_image_1 = new_unwrap_iris_3(
61     img_dec_1, iris_center_1, iris_radius_1,
62     pupil_center_1, pupil_radius_1, 1);
63 unwrapped_iris_image_2 = new_unwrap_iris_3(
64     img_dec_2, iris_center_2, iris_radius_2,
65     pupil_center_2, pupil_radius_2, 2);
66
67
68 % SPLIT UNWRAPPED IRIS IMAGE INTO 8 STRIPS
69 combined_image_1 = split_strips_filter(
70     unwrapped_iris_image_1);
71 combined_image_2 = split_strips_filter(
72     unwrapped_iris_image_2);

```

```

53 % ENCODE THE UNWRAPPED IRIS IMAGE USING GABOR
54 FILTER
55 [real_1, imag_1] = apply_gabor_filter(
56     combined_image_1);
57 [real_2, imag_2] = apply_gabor_filter(
58     combined_image_2);
59
60 % COMPARE
61 [hd_real, hd_imag] = hamming_distance_n(real_1,
62     imag_1, real_2, imag_2);
63
64 fprintf('Hamming Distance (Real): %.2f\n',
65     hd_real);
66 fprintf('Hamming Distance (Imaginary): %.2f\n',
67     hd_imag);
68
69 check_white_black_ratio(real_1, imag_1)
70 check_white_black_ratio(real_2, imag_2)
71
72
73 visualize_code_difference(real_1, imag_1, real_2,
74     imag_2)

```

Code 4. main.m

12.2. Funkcje pomocnicze

12.2.1. decrease_resolution()

```

1 function [outImage] = decrease_resolution(
2     inImage, resize_factor)
3
4 outImage = imresize(inImage, resize_factor);
5

```

Code 5. Funkcja do zmniejszenia wymiaru zdjęcia

12.2.2. detect_edges()

```

1 function [outImage, shift] = detect_edges(
2     inImage, filter_len, rgb2gray_true, sigma,
3     filter_ratio)
4
5 Gh = [-ones(1, filter_len), 0, ones(1,
6     filter_len)];
7 Gv = Gh';
8
9 if rgb2gray_true == true
10     img_gray = rgb2gray(inImage);
11 else
12     img_gray = inImage;
13 end
14
15 img_gray = im2double(img_gray);
16
17 filtersize = filter_ratio * sigma;
18 kernel = fspecial("gaussian", filtersize,
19     sigma);
20 image = imfilter(img_gray, kernel, 'conv',
21     'replicate');
22
23 horizontal = conv2(double(image), Gh, 'same');
24 vertical = conv2(double(image), Gv, 'same');
25
26 horizontal = horizontal .* horizontal;
27 vertical = vertical .* vertical;
28
29 result = horizontal + vertical;
30 result = sqrt(result);
31
32 [rows, cols, ~] = size(result);
33 border_width = filter_len+1;
34 result = imcrop(result, [border_width
35     border_width cols-(2*border_width) rows-(2*
36     border_width)]);

```

```

31     shift = [border_width, border_width];
32
33     outImage = result;
34 end

```

Code 6. Funkcja detekcji krawędzi

12.2.3. *find_iris()*

```

1 function [iris_center, iris_radius] = find_iris(
2     img_edges, shift, R_min, R_max,
3     iris_threshold)
4
5 HS_iris = hough_transform(img_edges, R_min,
6     R_max, iris_threshold);
7
8 [max_value, max_index] = max(HS_iris(:));
9 [Y, X, R_index] = ind2sub(size(HS_iris),
10    max_index);
11 R = R_min + R_index - 1;
12 X = X + shift(1);
13 Y= Y + shift(2);
14
15 iris_center = [X,Y];
16 iris_radius = R;
17
18 end

```

Code 7. Funkcja do znalezienia okręgu tęczówki

12.2.4. *hough_transform_iris()*

```

1 function HS = hough_transform(img, r_min, r_max,
2     threshold)
3
4 [rows, cols] = size(img);
5
6 HS = zeros(rows,cols,r_max - r_min + 1);
7
8 for r = r_min:r_max
9     for row = 1:rows
10         for col = 1:cols
11             if img(row,col)>threshold
12                 for theta = -pi/4:0.1*pi:pi/3
13                     a = round(col - r*cos(theta));
14                     b = round(row - r*sin(theta));
15                     if a >= 1 && a <= cols && b
16                         >= 1 && b <= rows
17                         HS(b,a,r - r_min + 1) =
18                         HS(b,a,r-r_min + 1) + img(row,col);
19                         % HS(b,a,r - r_min + 1)
20                         = HS(b,a,r-r_min + 1) + 1;
21                     end
22                     end
23                     for theta = 2/3*pi:0.1*pi:1.25*
24                         pi
25                         a = round(col - r*cos(theta));
26                         b = round(row - r*sin(theta));
27                         if a >= 1 && a <= cols && b
28                             >= 1 && b <= rows
29                             HS(b,a,r - r_min + 1) =
30                             HS(b,a,r-r_min + 1) + img(row,col);
31                             % HS(b,a,r - r_min + 1)
32                             = HS(b,a,r-r_min + 1) + 1;
33                         end
34                     end
35                 end
36             end
37         end
38     end
39 end

```

Code 8. Funkcja implementująca kołową transformację Hough'a dla tęczówki

12.2.5. *find_pupil()*

```

1 function [pupil_center, pupil_radius] =
2     find_pupil(img_edges, shift, r_min, r_max,
3     pupil_threshold)
4
5 HS_pupil = hough_transform_pupil(img_edges,
6     r_min, r_max, pupil_threshold);
7
8 [max_value, max_index] = max(HS_pupil(:));
9 [y, x, r_index] = ind2sub(size(HS_pupil),
10    max_index);
11 r = r_min + r_index - 1;
12 x = x + shift(1);
13 y = y + shift(2);
14
15 pupil_center = [x,y];
16 pupil_radius = r;
17
18 end

```

Code 9. Funkcja do znalezienia okręgu żrenicy

12.2.6. *hough_transform_pupil()*

```

1 function HS = hough_transform_pupil(img, r_min,
2     r_max, threshold)
3
4 [rows, cols] = size(img);
5
6 HS = zeros(rows,cols,r_max - r_min + 1);
7
8 for r = r_min:r_max
9     for row = 1:rows
10         for col = 1:cols
11             if img(row,col)>threshold
12                 for theta = 0.1*pi:0.1*pi:2*pi
13                     a = round(col - r*cos(theta));
14                     b = round(row - r*sin(theta));
15                     if a >= 1 && a <= cols && b
16                         >= 1 && b <= rows
17                         HS(b,a,r - r_min + 1) =
18                         HS(b,a,r-r_min + 1) + img(row,col);
19                         % HS(b,a,r - r_min + 1)
20                         = HS(b,a,r-r_min + 1) + 1;
21                     end
22                 end
23             end
24         end
25     end
26 end

```

Code 10. Funkcja implementująca kołową transformację Hough'a dla żrenicy

12.2.7. *daugman_script()*

```

1 r_min = 115;
2 r_max = 170;
3 A = zeros(rows, cols, r_max - r_min + 1);
4
5 V = img_edges;
6
7 for x = 1:cols
8     for y = 1:rows
9         for r = r_min:r_max
10             for theta = 0:pi/45:2*pi-(pi/180)
11                 xc = round(x + r * cos(theta));
12                 yc = round(y + r * sin(theta));
13
14                 if xc >= 1 && xc <= cols && yc
15                     >= 1 && yc <= rows
16                     A(y, x, r - r_min + 1) = A(y,
17                     , x, r - r_min + 1) + V(yc, xc);
18                 end
19             end
20         end
21     end
22 end

```

```

18     end
19 end
20
21 [max_value, max_index] = max(A(:));
22 [y0, x0, r0_index] = ind2sub(size(A), max_index)
23 ;
24 r0 = r_min + r0_index - 1;
25
26 x0 = x0 + shift(1) - 1;
27 y0 = y0 + shift(2) - 1;

```

Code 11. Główna część funkcji realizującej operator Daugmana

12.2.8. unwrap_iris()

```

1 function unwrappedIris = unwrap_iris(img_gray,
2     irisCenter, irisRadius, pupilCenter,
3     pupilRadius, num)
4     fixedThetaSamples = 360;
5     fixedRadialSamples = 100;
6
7     unwrappedIris = zeros(fixedRadialSamples,
8         fixedThetaSamples);
9
10    for i = 1:fixedThetaSamples
11        theta = (i - 1) * 2 * pi /
12            fixedThetaSamples + (270/fixedThetaSamples)
13        * 2 * pi;
14        cosTheta = cos(theta);
15        sinTheta = sin(theta);
16        for r = 1:fixedRadialSamples
17            rNorm = r / fixedRadialSamples;
18            actualRadius = pupilRadius + rNorm *
19                (irisRadius - pupilRadius);
20            x = round(pupilCenter(1) +
21                actualRadius * cosTheta + rNorm * (
22                    irisCenter(1) - pupilCenter(1)));
23            y = round(pupilCenter(2) +
24                actualRadius * sinTheta + rNorm * (
25                    irisCenter(2) - pupilCenter(2)));
26            if x > 0 && x <= size(img_gray, 2)
27            && y > 0 && y <= size(img_gray, 1)
28                unwrappedIris(r, i) = img_gray(y
29                , x);
30            end
31        end
32    end
33 end

```

Code 12. Funkcja rozwijająca tęczówkę metodą "rubber sheet"

12.2.9. split_strips_filter()

```

1 function combinedImage = split_strips_filter(
2     unwrappedIris, selectedStripIndex)
3     if nargin < 2
4         selectedStripIndex = 1;
5     end
6
7     [height, width] = size(unwrappedIris);
8
9     num_strips = 8;
10
11    strip_height = round(height / num_strips);
12
13    gaussFilter = fspecial('gaussian', [
14        strip_height, 1], strip_height / 2);
15
16    strips = zeros(num_strips, width);
17    weightedStrips = cell(num_strips, 1);
18
19    for i = 1:num_strips
20        start_row = (i - 1) * strip_height + 1;
21        if i == num_strips
22            end_row = height;
23        else
24            end_row = i * strip_height;
25
26        end
27    end
28
29    strip_mean = mean(stri

```

```

23     end
24
25     strip = unwrappedIris(start_row:end_row,
26     :);
27
28     gaussFilterResized = imresize(
29         gaussFilter, [size(strip, 1), 1], 'nearest');
30
31     weightedStrip = sum(strip .* gaussFilterResized, 1) / sum(
32         gaussFilterResized);
33     strips(i, :) = weightedStrip;
34     weightedStrips{i} = weightedStrip;
35
36 end
37
38 start_row = (selectedStripIndex - 1) *
39 strip_height + 1;
40 if selectedStripIndex == num_strips
41     end_row = height;
42 else
43     end_row = selectedStripIndex *
44 strip_height;
45 end
46
47 selected_strip = unwrappedIris(start_row:
48 end_row, :);
49
50 combinedImage = strips;
51
52 end

```

Code 13. Funkcja do rozdzielenia obrazu tęczówki na 8 uśrednionych Gaussem pasków

12.2.10. apply_gabor_filter()

```

1 function [binary_real_iris_code,
2     binary_imag_iris_code] = apply_gabor_filter(
3     combinedImage)
4     wavelength = 33;
5     orientation = 0;
6     aspectRatio = 0.5;
7     bandwidth = 1;
8
9     num_strips = size(combinedImage, 1);
10
11    gaborRealResponses = cell(num_strips, 1);
12    gaborImagResponses = cell(num_strips, 1);
13
14    for i = 1:num_strips
15        strip = combinedImage(i, :);
16
17        stripMean = mean(strip, 'all');
18        strip = strip - stripMean;
19
20        theta = orientation * pi / 180;
21        sigma = wavelength * sqrt(log(2) / 2) *
22            ((2^bandwidth + 1) / (2^bandwidth - 1)) / pi
23        ;
24        sigma_x = sigma;
25        sigma_y = sigma / aspectRatio;
26        nstds = 3;
27        xmax = max(abs(nstds * sigma_x * cos(
28            theta)), abs(nstds * sigma_y * sin(theta)));
29        xmax = ceil(max(1, xmax));
30        ymax = max(abs(nstds * sigma_x * sin(
31            theta)), abs(nstds * sigma_y * cos(theta)));
32        ymax = ceil(max(1, ymax));
33        [x, y] = meshgrid(-xmax:xmax, -ymax:ymax
34        );
35        x_theta = x * cos(theta) + y * sin(theta
36        );
37        y_theta = -x * sin(theta) + y * cos(
38            theta);
39        gaborReal = exp(-0.5 * (x_theta.^2 /
40            sigma_x.^2 + y_theta.^2 / sigma_y.^2)) .* cos(
41            2 * pi / wavelength * x_theta);
42        gaborImag = exp(-0.5 * (x_theta.^2 /
43            sigma_x.^2 + y_theta.^2 / sigma_y.^2)) .* sin(
44            2 * pi / wavelength * x_theta);
45
46    end
47
48 end

```

```

32         gaborRealFiltered = conv2(strip,
33             gaborReal, 'same');
34         gaborImagFiltered = conv2(strip,
35             gaborImag, 'same');

36     gaborRealResponses{i} =
37         gaborRealFiltered;
38     gaborImagResponses{i} =
39         gaborImagFiltered;
40 end

41 real_iris_code = cell2mat(gaborRealResponses);
42 imag_iris_code = cell2mat(gaborImagResponses);

43 [height, width] = size(real_iris_code);

44 stretched_real_iris_code = imresize(
45     real_iris_code, [height*10, width]);
46 stretched_imag_iris_code = imresize(
47     imag_iris_code, [height*10, width]);

48 binary_real_iris_code = real_iris_code > 0;
49 binary_imag_iris_code = imag_iris_code > 0;

50 stretched_binary_real_iris_code = imresize(
51     binary_real_iris_code, [height*10, width]);
52 stretched_binary_imag_iris_code = imresize(
53     binary_imag_iris_code, [height*10, width]);
end

```

Code 14. Funkcja realizująca filtrację Gabora

12.2.11. hamming_distance()

```

1 function [HD_real, HD_imag] = hamming_distance_n
2     (iris_code1_real, iris_code1_imag,
3      iris_code2_real, iris_code2_imag)
4     function HD = calc_hd(code1, code2)
5         XOR_code = xor(code1, code2);
6         HD = sum(XOR_code(:)) / numel(code1);
7     end

8     shifts = -4:4;
9     num_shifts = length(shifts);
10    HD_real_vals = zeros(1, num_shifts);
11    HD_imag_vals = zeros(1, num_shifts);

12    for i = 1:num_shifts
13        shift_val = shifts(i);
14        if shift_val < 0
15            shift_val = abs(shift_val);
16            shifted_real = circshift(
17                iris_code2_real, [0, -shift_val]);
18            shifted_imag = circshift(
19                iris_code2_imag, [0, -shift_val]);
20        else
21            shifted_real = circshift(
22                iris_code2_real, [0, shift_val]);
23            shifted_imag = circshift(
24                iris_code2_imag, [0, shift_val]);
25        end

26        HD_real_vals(i) = calc_hd(
27            iris_code1_real, shifted_real);
28        HD_imag_vals(i) = calc_hd(
29            iris_code1_imag, shifted_imag);
30    end

31    HD_real = min(HD_real_vals);
32    HD_imag = min(HD_imag_vals);
33 end

```

Code 15. Funkcja obliczająca odległość Hamminga

13. Linki

Link do całego projektowego kodu na [githubie](#).

References

- [1] S. B., 58 - *What are Gabor filters?* [Online]. Available: <https://www.youtube.com/watch?v=QEz4bG9P3Qs>.
- [2] *Encyclopedia of Biometrics*. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-1-4899-7488-4_307.
- [3] *Gabor Filter*. [Online]. Available: https://en.wikipedia.org/wiki/Gabor_filter.
- [4] *How Circle Hough Transform works*. [Online]. Available: <https://www.youtube.com/watch?v=Ltqt24SQoI>.
- [5] *Illustration of Hough Circle Transform*. [Online]. Available: https://www.researchgate.net/figure/Illustration-of-Hough-Circle-Transform_fig3_329683536.
- [6] L. Masek, *Recognition of Human Iris Patterns for Biometric Identification*. [Online]. Available: <https://www.peterkovesi.com/studentprojects/libor/LiborMasekThesis.pdf>.
- [7] *Podstawy Biometrii (slajdy)*, Dr. Jan Mazur.