**Rozpoznawanie człowieka metodami biometrii**

**Sprawozdanie 1**

Michalik Piotr

Spis treści

[Wstęp 3](#_Toc511994914)

[Segmentacja tęczówki 3](#_Toc511994915)

[Kod tęczówki 6](#_Toc511994916)

[Porównanie 8](#_Toc511994917)

[Wyniki 9](#_Toc511994918)

[Wnioski 11](#_Toc511994920)

[Źródła 11](#_Toc511994921)

# Wstęp

W poniższym dokumencie przedstawię środki użyte do realizacji pierwszego projektu na przedmiot ‘Wykrywanie człowieka metodami biometrii’. Po kolei opisze i przedstawię metody zaimplementowane w tym projekcie. Następnie skupie się na części właściwej, to znaczy na zbadaniu zaproponowanych i zaimplementowanych przeze mnie algorytmów. Na samym końcu zamieszczę krótkie podsumowanie etapu. Dla każdej opisanej metody zostaną zamieszczone screenshoty prezentujące działanie poszczególnej funkcjonalności.

W tym sprawozdaniu opisane, zaprezentowane i ocenione zostaną dwa algorytmy: algorytm segmentacji tęczówki oraz algorytm kodowania tęczówki.

# Segmentacja tęczówki

## Teoria

Aby przystąpić do procesu kodowania tęczówki, niezbędne jest najpierw znalezienie tęczówki na zdjęciu oka. Problem ten został już opracowany na poprzedzającym przedmiocie (‘Analiza i przetwarzanie obrazów biometrycznych) jednak poprzednie rozwiązanie było mało dokładne oraz niewydajne. Z tych przyczyn postanowiłem opracować metodę segmentacji ponownie. Algorytm:

1. Przekształcamy zdjęcie do skali szarości.
2. Czarno białe zdjęcie zamieniamy na zdjęcie czterokolorowe. Kolory: biały, czarny oraz dwa kolory położone pośrodku pomiędzy średnią wartością kolory, a białym i czarnym.
3. Dwukrotnie na przemian aplikujemy filtr Gaussa oraz każdy pixel który ma więcej niż 5 czarnych sąsiadów kolorujemy na czarno.
4. Usuwamy pojedyncze czarne pixele.
5. Następnie w otrzymanym obrazie dla każdego potencjalnego środka źrenicy:
   1. Obliczamy średnią wartość koloru zebranego po okręgu w różnych odległościach od tego środka.
   2. Korzystając z wartości koloru określonego w pkt. 2 (jako środek między kolorem średnim a czarnym) pomnożonego przez współczynnik 0.28 znajdujemy potencjalny promień w miejscu zmiany koloru z czarnego na wymieniony kolor idąc od potencjalnego środka źrenicy na zewnątrz.
6. Wybieramy środkiem i promieniem źrenicy tą parę (potencjalny środek i potencjalny promień) dla których promień jest największy.

W ten sposób znaleźliśmy środek i promień źrenicy. Teraz należy wyznaczyć promień tęczówki. Dla uproszczenia zakładamy że oba okręgi ograniczające tęczówkę są współśrodkowe, zatem możemy wykorzystać znany środek źrenicy. Algorytm:

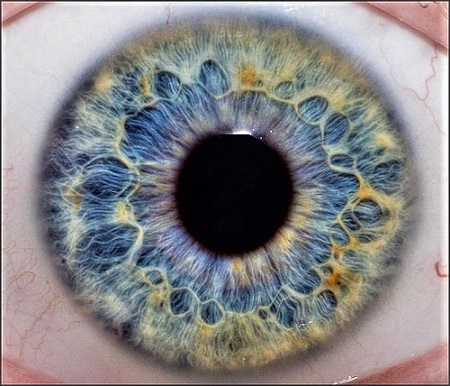
1. Na oryginalnym zdjęciu, zwiększamy jego kontrast.
2. Przekształcamy zdjęcie do skali szarości.
3. Ponownie zwiększamy kontrast zdjęcia.
4. Aplikujemy Filtr Gaussa.
5. Każdy pixel który ma więcej niż 5 czarnych sąsiadów kolorujemy na czarno.
6. Otrzymane zdjęcie zamieniamy na zdjęcie czterokolorowe. Kolory: biały, czarny oraz dwa kolory położone pośrodku pomiędzy średnią wartością kolory, a białym i czarnym.
7. Usuwamy pojedyncze czarne pixele.
8. Obliczamy średnią wartość koloru zebranego po okręgu w różnych odległościach (większych od promienia źrenicy) od środka źrenicy.
9. Korzystając z wartości koloru określonego w pkt. 6 (jako środek między kolorem średnim a czarnym) pomnożonego przez współczynnik 0.6 znajdujemy potencjalny promień w miejscu zmiany koloru z czarnego na wymieniony kolor idąc od środka źrenicy na zewnątrz.
10. Wybrany promień jest promieniem tęczówki.

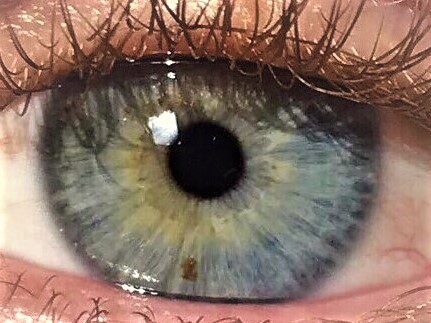
## Implementacja

Algorytmy ze względy na swoją dużą złożoność obliczeniową zostały zaimplementowane tak aby pracować nie bezpośrednio na zdjęciu, ale na tablicy bajtów reprezentującej zdjęcie. Reprezentacja taka pozwala zaoszczędzić dużo czasu. Po drugie metody zostały zaimplementowane wielowątkowo aby możliwie przyspieszyć algorytm. Pomimo tego obliczenia zajmują dużo czasu. Implementacja dokładnie odzwierciedla opisany powyżej algorytm.

Rezultaty tego algorytmu przedstawiam w następnym akapicie.

## ScreenShot







# Kod tęczówki

## Teoria

Drugim algorytmem niezbędny w tym projekcie jest algorytm wyznaczania kodu z tęczówki. W oparciu o teorie Daugmana, tęczówkę będziemy reprezentować jako ciąg 2048 bitów. Wyznaczymy je z ośmiu okręgów rozchodzących się od środka źrenicy. Każdy taki okrąg będziemy traktować jako ciąg poziomów jasności i kodować będziemy zmianę tych poziomów w ciągu. Każdy okrąg będzie reprezentowany przez 256 bitów, zatem każda zmiana jasności będzie reprezentowana dwoma bitami a okrąg będzie miał 128 elementów. Algorytm:

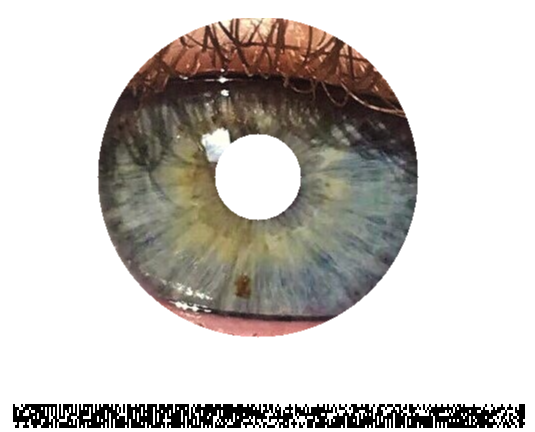
1. Przekształcamy zdjęcie do skali szarości.
2. Wyznaczamy 8 równo odległych od siebie okręgów na tęczówce.
3. Każdy okrąg przycinamy zgodnie z algorytmem Daugmana tak że pierwsze cztery okręgi są pełne, za wyjątkiem 30 stopniowego wycięcia na dole. Kolejne dwa okręgi tworzą symetrycznie rozmieszczony kąt 226 stopni, a ostatnie dwa również symetrycznie tworzą 180 stopni
4. Dla każdego pixela w każdym okręgu obliczamy jego średnią wartość w sąsiedztwie (4x4) i te okręgi zapamiętujemy jako tablice średnich kolorów z danego okręgu o długości odpowiadającej każdemu okręgowi.
5. Ponieważ każdy okrąg może mieć różną długość, normalizujemy je do długości 128 wartości. Dla każdego z docelowych 128 punktów znajdujemy na otrzymanym okręgu jego najbardziej przybliżoną pozycje (procentowa odległość od początku). Znajdujemy również wartość o jedną w lewo i o jedną w prawo od znalezionego punktu. Następnie wykorzystujemy interpolacje kwadratową aby określić dokładną wartość punktu jaką potrzebujemy. W ten sposób jesteśmy w stanie dłuższy ciąg skrócić, a za krótki wydłużyć.
6. Dla ośmiu okręgów równej długości obliczamy ich zmianę natężenia (gradient). Definiujemy 4 rodzaje zmiany gradientu: gwałtowny wzrost, wzrost, spadek i gwałtowny spadek. Gradienty kolejno będziemy kodować jako 00, 01, 11, 10. W ten sposób binarnie kodujemy tendencje zmiany wartości koloru w każdym okręgu.
7. Osiem ciągów binarnych łączymy w jeden gotowy kod binarny.

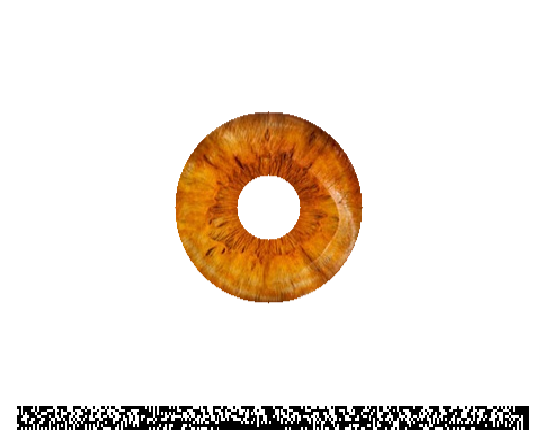
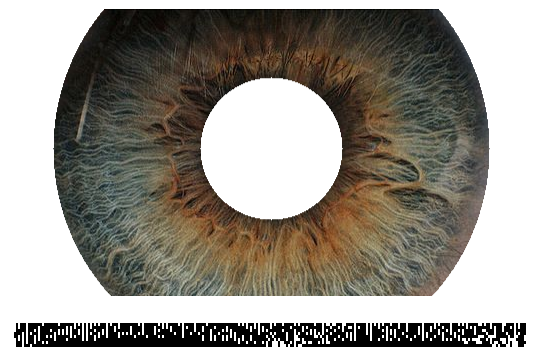
## Implementacja

Algorytm został zaimplementowany zgodnie z opisem teoretycznym. Analogicznie jak w poprzednich akapitach, program jest złożony obliczeniowo dlatego występują liczne próby optymalizacji w postaci równoległych obliczeń.

Rezultaty tego algorytmu przedstawiam w następnym akapicie.

## ScreenShot





# Porównanie

## Teoria

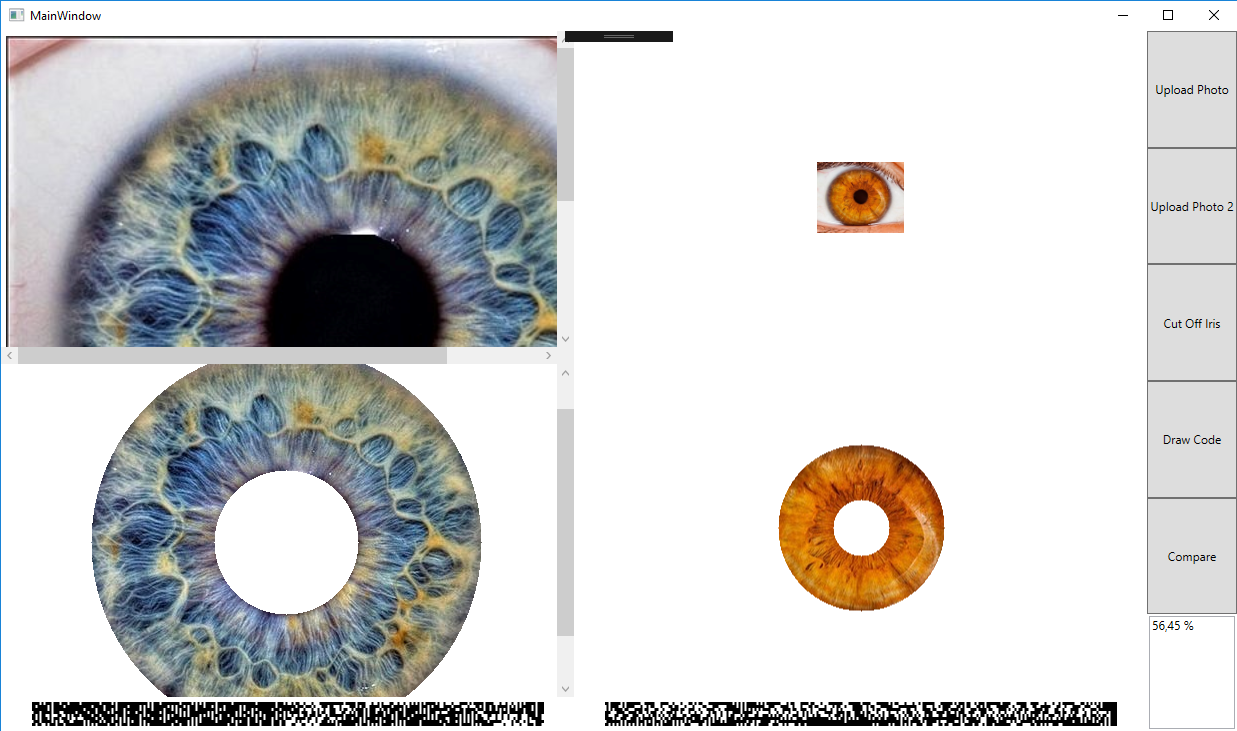
Jako porównanie dwóch kodów tęczówek używam tutaj metryki Hamminga. Odległość Hamminga jest różnicą w bitach pomiędzy oboma kodami. Jako wynik aplikacja wyświetla procentowo zgodność obu porównywanych kodów.

## Przykład

Poniższe dwa kody algorytm obliczył na podobieństwo 63,19 %.



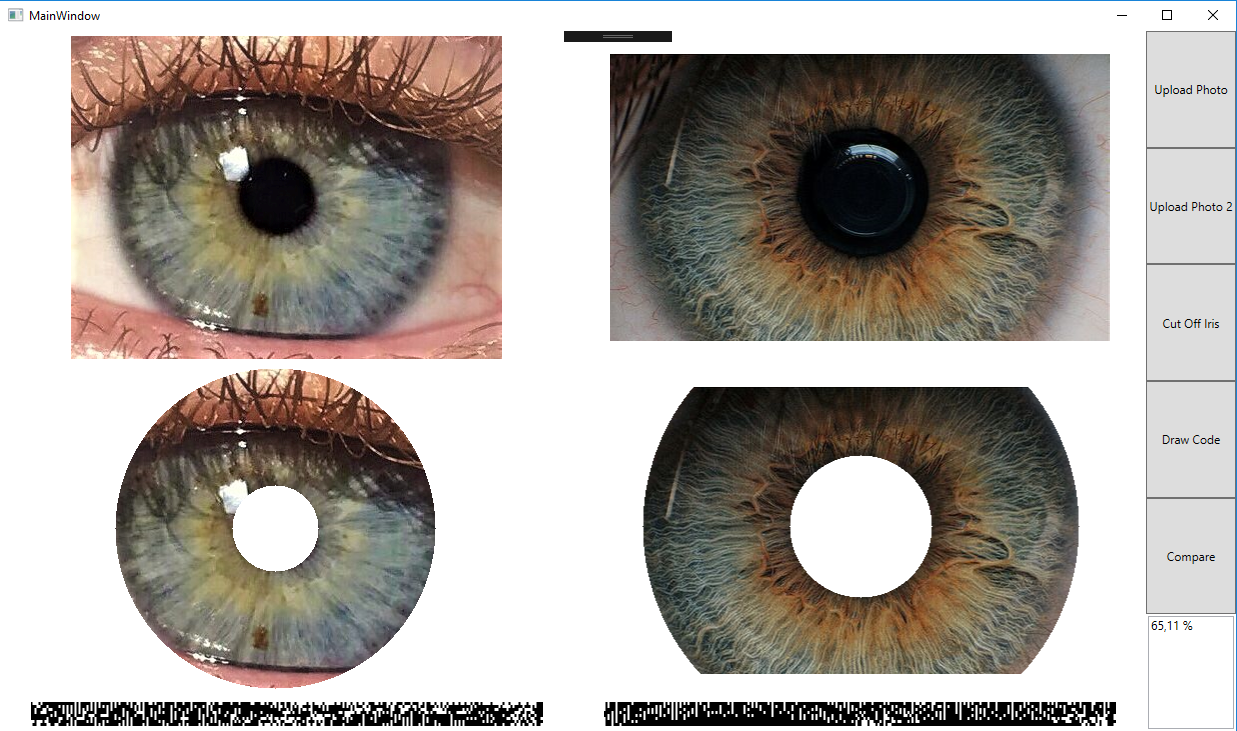
# Wyniki

Porównanie dwóch różnych zdjęć parami daje wyniki satysfakcjonujące, co znaczy że pokazuje różnicę miedzy nimi. Np:

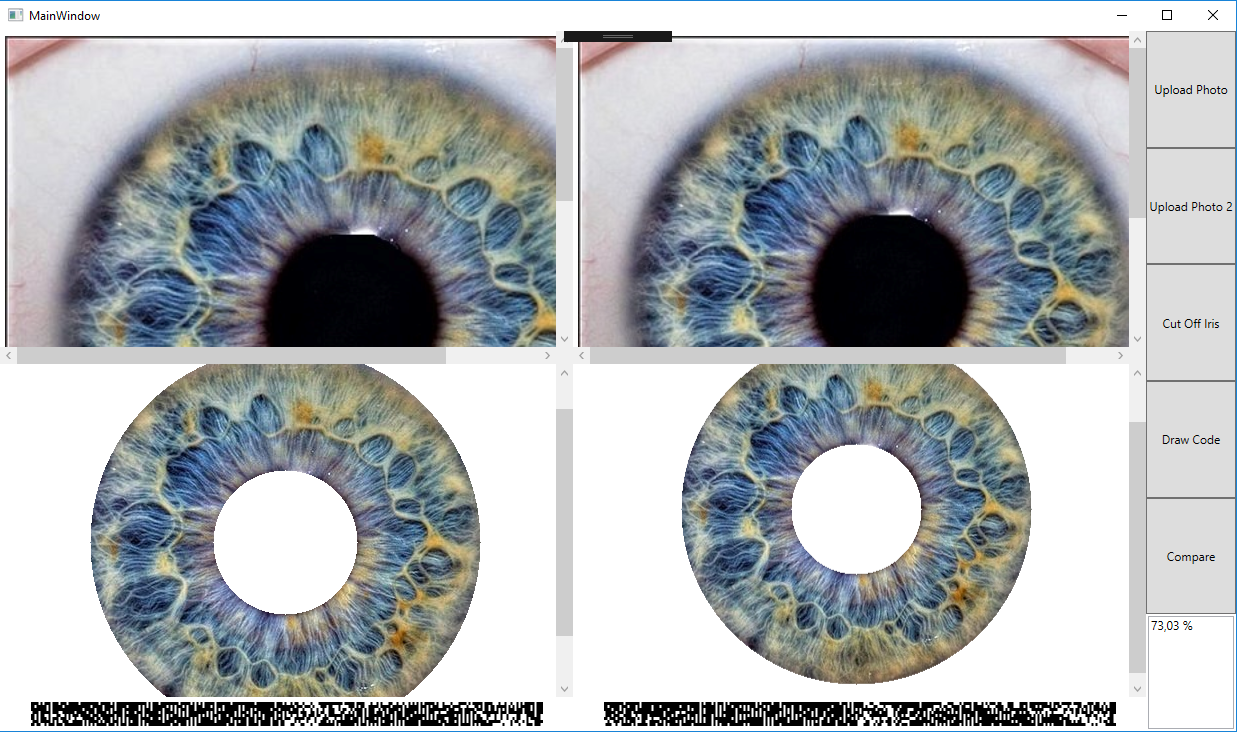
Rysunek - Podobieństwo 56.45 %

# 

Rysunek - Podobieństwo 63.19 %



Rysunek - Podobieństwo 65.11 %

Jednak jeżeli porównamy zdjęcia tej samej tęczówki, ale jedno będzie zmniejszone o 90% to wyniki wyglądają następująco:

Rysunek - Podobieństwo 73.03 %

# Wnioski

Metoda przeze mnie zaimplementowana brzmiała pomysłowo i teoretycznie miała szanse powodzenia. Jednak na konkretnych przykładach widać liczne niedokładności tej metody. Gołym okiem widać kiedy kody tęczówek różnią się od siebie a kiedy są do siebie bardzo podobne. Przykład zaprezentowany na Rysunku 4 pokazuje że kody są do siebie podobne. Jednak pojedyncze wartości kodu są lekko przesunięte na obu obrazkach. W tym wypadku metryka Hamminga słusznie pokazuje dużą różnice między oboma kodami. Należałoby opracować lepszą metody porównywania kodów, która nie sprawdzała by dosłownie wartości binarnych, ale tendencje zmiany wartości tego kodu. Wtedy dla różnych tęczówek wynik nadal by wskazywał dużą różnice, ale na zdjęciach takich samych jednak przekształconych (skala, obrót) skuteczność by drastycznie wzrosła. Jedną z metod którą można by zaadaptować jest DTW. Jednak tą metodę będziemy implementować w kolejnym projekcie.

# Źródła

* Krzysztof Ślot – „Wybrane zagadnienia biometrii”