**Rozpoznawanie człowieka metodami biometrii**

**Sprawozdanie 1**

Michalik Piotr

Spis treści

[Wstęp 3](#_Toc503826854)

[KMM 4](#_Toc503826855)

[K3M 9](#_Toc503826856)

[Porównanie metod 13](#_Toc503826858)

[Listy 14](#_Toc503826859)

[Źródła 14](#_Toc503826860)

# 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) 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.

## Implementacja

Algorytm został zaimplementowany dosłownie tak jak w opisie teoretycznym. Dlatego poniższy opis odnosi się do poszczególnych punktów.

## ScreenShot



Rysunek 1- Przykład Preprocessingu

# Kod tęczówki

## Teoria

Drugim algorytmem zaproponowanym do problemu ścieniania jest K3M. Algorytm został opracowany przez Panów: Khalida Saeed, Mariusza Rybnik, Marka Tabędzki oraz Marcina Adamski jako odpowiedź na niewystarczającą skuteczność algorytmu KMM. Jest to poniekąd jego rozwinięcie oraz udoskonalenie.

Idea K3M opiera się na eliminacji zbędnych pixeli w kolejnych fazach w oparciu o opisane wcześniej listy sum wag sąsiadów. Cały zestaw faz możemy regularnie powtarzać aż uzyskamy obraz o grubości jednego pixela.

Ze względu na podobieństwo algorytmów, opis przygotowania zdjęcia, definicja sąsiedztwa oraz definicja wag jest taki sam.

Algorytm prezentuje się następująco:

1. Każdy pixel który ma chodziaż jednego zerowego sąsiada oznaczamy jako 2.
2. Faza 1 – usuwamy wszystkie pixele oznaczone jako 2, które posiadają 3 przylegających do siebie sąsiadów.
3. Faza 2 – usuwamy wszystkie pixele oznaczone jako 2, które posiadają 3 lub 4 przylegających do siebie sąsiadów.
4. Faza 3 – usuwamy wszystkie pixele oznaczone jako 2, które posiadają 3, 4 lub 5 przylegających do siebie sąsiadów.
5. Faza 4 – usuwamy wszystkie pixele oznaczone jako 2, które posiadają 3, 4, 5 lub 6 przylegających do siebie sąsiadów.
6. Faza 5 – usuwamy wszystkie pixele oznaczone jako 2, które posiadają 3, 4, 5, 6 lub 7 przylegających do siebie sąsiadów.
7. Wszystkie pozostałe 2 oznaczamy jako 1.
8. Powtarzamy kroki 1 – 7 dopóki nie zostanie usunięty ani jeden pixel.
9. Dla każdego pixela oznaczonego jako 2 sprawdzamy czy jest on w linii o grubości jednego pixela. W przeciwnym wypadku usuwamy go

Punkty 1-9 powtarzamy dopóki nie zostanie zmieniony ani jeden pixel.

W celu określenia zarówno każdej liczby przylegających sąsiadów, jak również w celu sprawdzenia grubości linii posługujemy się siedmioma listami sum wag. Listy wyglądają analogicznie jak w poprzednim algorytmie. Różnią się jedynie wartościami, których nie będziemy tutaj przytaczać, a ich treść znajdzie się przed źródłami.

## Implementacja

Algorytm został zaimplementowany dosłownie tak jak w opisie teoretycznym. Dlatego poniższy opis odnosi się do poszczególnych punktów.

1. W podwójnej pętli przechodzimy po obrazie. Dla każdego pixela który nie jest zerem sprawdzamy czy chociaż jeden jego sąsiad jest zerem. W tym celu obliczamy sumę wag jego sąsiadów i sprawdzamy czy wartość jest w liście A0. Jeżeli tak, zamieniamy jedynkę na dwójkę.

W punktach: 2, 3, 4, 5, oraz 6 zaimplementowano następujące kroki:

W podwójnej pętli przechodzimy po obrazie. Dla każdego pixela który jest dwójką liczymy sumę wag odpowiadającą jego sąsiadom. Jeżeli waga znajduje się w odpowiedniej liście wag (kolejno A1, A2, A3, A4, A5), zamieniamy wartość pixela na 0.

1. W podwójnej pętli przechodzimy po obrazie. Dla każdego pixela który ma wartość 2 zmieniamy mu wartość na 1.
2. W podwójnej pętli przechodzimy po obrazie. Dla każdego pixela który nie jest zerem liczymy sumę wag odpowiadającą jego sąsiadom. Jeżeli waga znajduje się w liście wag OnePixel, zamieniamy wartość pixela na 0.

# Wyniki i Wnioski

W poniższym akapicie porównane zostaną oba algorytmy.

Jak widać na powyższym obrazie oraz wcześniejszych screenshotach, oba algorytmy spełniają swoje zadanie. I w jednym i w drugim przypadku możemy mówić o sukcesie w ścienianiu obrazu. Jednak już na pierwszy rzut oka widać różnice. Algorytm K3M daje wizualnie lepszy efekt niż KMM. Lepszą skuteczność można zauważyć na 3 płaszczyznach:

* Lepsze zachowanie ciągłości linii
* Większa liczba szczegółów
* Lepsze odwzorowanie kształtów

Z drugiej jednak strony z samej nazwy wynika, że głównym zadaniem obu algorytmów jest ścienianie. Pomimo niezachowania ciągłości linii, algorytm KMM wydaje się generować obraz o cieńszych krawędziach niż K3M kosztem jakości.

# Źródła

* „Algorytm do ścieniania obrazów: Implementacja i zastosowania”: Khalid Saeed, Mariusz Rybnik, Marek Tabędzki, Marcin Adamski.
* „K3M: A universal algorithm for image skeletonization and a review of thinning techniques” : Khalid Saeed, Mariusz Rybnik, Marek Tabędzki, Marcin Adamski.