WYDZIAŁ MATEMATYKI I NAUK INFORMACYJNYCH
POLITECHNIKA WARSZAWSKA

Detekcja tęczówki oka i jej dekodowanie.

BIOMETRIA
PROJEKT 2 - DOKUMENTACJA

Maciej Momot, Filip Szlingiert

24 kwietnia 2025

Spis treści

1 Wstęp

Rozpoznawanie człowieka na podstawie obrazu tęczówki stanowi jedno z najbardziej niezawodnych podejść w dziedzinie biometrii. Celem niniejszego projektu było opracowanie algorytmu umożliwiającego segmentację tęczówki z obrazu oka oraz przygotowanie jej do dalszej analizy identyfikacyjnej. W szczególności skupiliśmy się na wykryciu granic źrenicy i tęczówki, przekształciliśmy współrzędne do układu prostokątnego oraz przygotowaliśmy obraz do kodowania. Projekt oparty był na obrazach pochodzących ze zbioru danych MMU Iris Dataset, a realizacja poszczególnych etapów przebiegała zgodnie z wytycznymi przedstawionymi w specyfikacji zadania.

Nasz algorytm pisaliśmy w notebook'u i jego zawartość jest możliwa do przetestowania w pliku IrisRecognition.ipynb. Program działa na zasadzie skryptu, w którym należy podać ścieżkę do zdjęcia oka, które chcemy przetwarzać oraz progi X_p i X_i potrzebne do binaryzacji kolejno źrenicy i tęczówki.

2 Opis działania skryptu.

Skrypt rozpoczyna swoje działanie od wczytania obrazu oka w formacie `.bmp`, znajdującego się w katalogu zbioru danych MMU Iris Dataset. Następnie obraz ten przekształcany jest do skali szarości, co umożliwia dalsze przetwarzanie numeryczne.

W kolejnym etapie wykonywana jest binaryzacja obrazu przy wykorzystaniu odpowiednio dobranych progów dla źrenicy oraz tęczówki. Celem binaryzacji jest wyodrębnienie interesujących nas struktur na obrazie w sposób pozwalający na dalszą segmentację.

Po przygotowaniu binarnego obrazu źrenicy, wykonywane są operacje morfologiczne oraz eliminacja szumów, które umożliwiają poprawne wykrycie granic źrenicy. Analogiczna procedura wykonywana jest dla detekcji tęczówki, przy czym wykorzystywane są dodatkowo projekcje obrazu w celu poprawnego oszacowania środka i promienia.

W dalszej części algorytmu, na podstawie wyznaczonych granic, wykonywana jest transformacja współrzędnych biegunowych do prostokątnych, w celu rozwinięcia pierścienia tęczówki do postaci prostokątnego obrazu.

Ostatecznie, rozwinięty prostokątny obraz poddawany jest algorytmowi Daugmana, w którym następuje podział prostokąta na 8 radialnych pasów, zastosowanie transformaty falkowej Gabora i wyznaczenie kodu tęczówki. Tak powstały kod tęczówki porównujemy z innymi kodami przy użyciu odległości Hamminga.

W dalszej części dokumentacji przedstawiamy bardziej szczegółowy opis poszczególnych kroków algorytmu.

2.1 Binaryzacja.

Pierwszym krokiem przetwarzania obrazu była jego binaryzacja, przeprowadzana osobno dla źrenicy oraz tęczówki. Celem binaryzacji było wyodrębnienie struktur istotnych na potrzeby dalszej segmentacji. Binaryzację zrealizowano przy wykorzystaniu globalnego progu jasności wyznaczanego na podstawie średniej intensywności pikseli w obrazie.

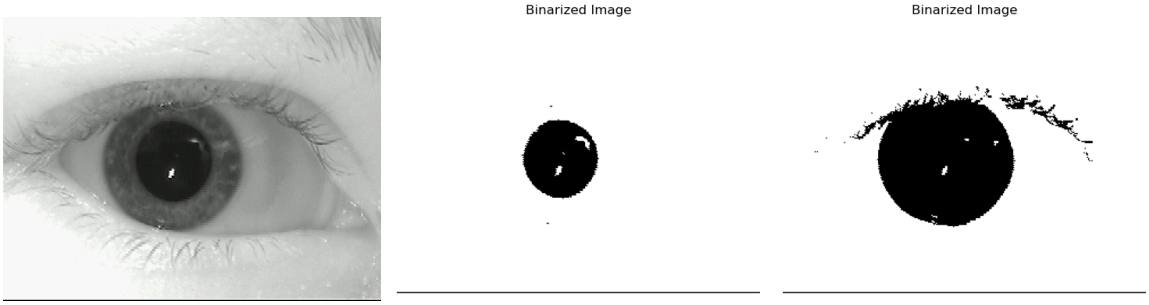
Próg binaryzacji T wyznaczany był ze wzoru:

$$T = \frac{P}{X}, \text{ gdzie}$$

- P — średnia jasność pikseli w obrazie (wartość średnia macierzy obrazu w skali szarości),
- X — parametr binaryzacji, odpowiednio X_p dla źrenicy oraz X_i dla tęczówki.

Po wyznaczeniu progu T każdy piksel obrazu był porównywany z tą wartością. Wybieraliśmy piksele ciemniejsze niż próg. Dzięki temu możliwe było wstępne oddzielenie interesujących obszarów (źrenicy lub tęczówki) od tła.

Wartości parametrów X_p i X_i dobieraliśmy eksperymentalnie dla każdego analizowanego zdjęcia, ponieważ przy próbie ustalenia pewnej stałej wartości zawsze znajdowały się przypadki, gdzie binaryzacja okazywała się niepoprawna z naszego punktu widzenia.
Poniżej przedstawiamy przykładową binaryzację oka.



Rysunek 1: Efekt zastosowania binaryzacji: (lewo) orginalne zdjęcia, (środek) binaryzacja źrenicy z parametrem $X_p = 3.5$, (prawo) binaryzacja tęczówki z parametrem $X_i = 1.5$.

Jak możemy zauważyć binaryzacja bardzo dobrze wyekstrahowała źrenicę, ale binaryzacja dla tęczówki nie była idealna. Pozostające po binaryzacji rzęsy to bardzo popularny problem w tego typu zadaniu, dlatego potrzebne nam były kolejne operacje na naszych zbinaryzowanych zdjęciach aby się tego pozbyć.

2.2 Detekcja granicy źrenicy i tęczówki.

2.2.1 Detekcja granicy źrenicy.

Po zbinaryzowaniu obrazu w celu wyodrębnienia źrenicy, kolejnym krokiem była eliminacja zakłóceń oraz dokładna segmentacja tego obszaru. W tym celu wykonaliśmy szereg operacji morfologicznych na obrazie binarnym źrenicy, których celem było oczyszczenie go z szumów oraz wzmocnienie krawędzi źrenicy.

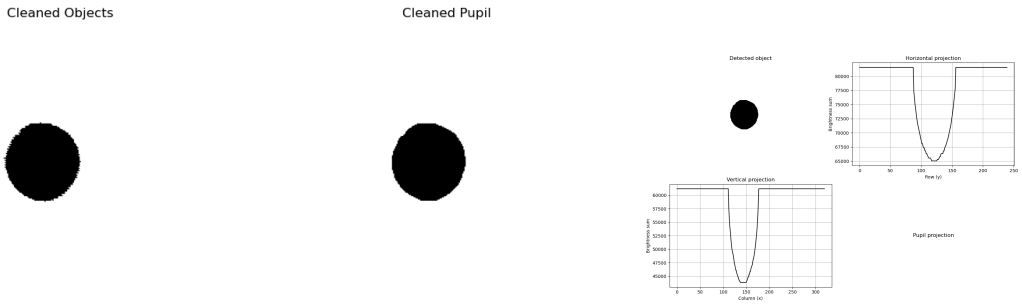
Na początku usuwaliśmy wszystkie obiekty poza największym konturem – zakładaliśmy, że źrenica będzie największym jednolitym obszarem w obrazie binarnym. Następnie stosowaliśmy morfologiczne domknięcie (ang. closing) z użyciem dużego eliptycznego jądra o rozmiarze 5×5, które miało za zadanie wypełnić niewielkie dziury wewnątrz źrenicy. Jądro to można zapisać jako:

$$\text{kernel\_large} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Po operacji domknięcia wykonaliśmy erozję i dylatację z mniejszym jądrem o rozmiarze 3×3, również o kształcie eliptycznym. Celem tych operacji było dalsze wygładzenie krawędzi źrenicy oraz usunięcie małych fragmentów tła. Następnie zastosowaliśmy filtr medianowy o rozmiarze 5×5 oraz otwarcie (ang. opening), które pozwalały na dalsze usunięcie drobnych zakłóceń i izolację wyłącznie źrenicy.

Po zakończeniu procesu oczyszczania obrazu przystępowaliśmy do właściwej detekcji granicy źrenicy. W tym celu analizowaliśmy sumy wartości pikseli wzdłuż wierszy (projekcja pozioma) oraz kolumn (projekcja pionowa). Na podstawie tych projekcji wybieraliśmy współrzędne odpowiadające najmniejszym wartościom – odpowiadały one obszarom najciemniejszym, czyli prawdopodobnej lokalizacji środka źrenicy.

Poniżej przedstawiamy wygląd obrazów po poszczególnych etapach czyszczenia źrenicy oraz projekcje.



Rysunek 2: Proces czyszczenia: (lewo) największy kontur, (środek) zaawansowane funkcje i operacje morfologiczne, (prawo) wizualizacja projekcji pionowej i poziomej

Na podstawie obliczonych różnic w projekcjach, wyznaczaliśmy punkty, w których następował gwałtowny skok jasności – interpretowaliśmy je jako granice źrenicy w danym kierunku. Odległość między tymi granicami pozwalała nam oszacować promień źrenicy w poziomie oraz w pionie, a następnie przyjmowaliśmy ich średnią jako ostateczną wartość promienia.

2.2.2 Detekcja granicy tęczówki.

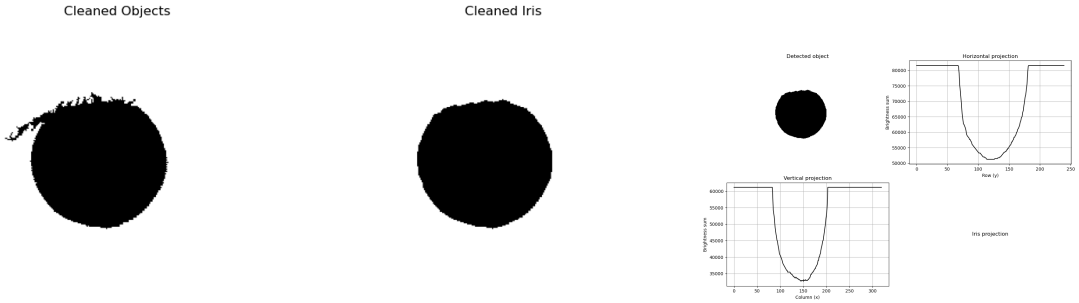
Detekcja granicy tęczówki była bardziej wymagająca niż detekcja źrenicy, ponieważ tęczówka zajmuje znacznie większy obszar, a jej zewnętrzne granice są często częściowo przysłonięte przez powiekę górną lub dolną. W związku z tym konieczne było zastosowanie bardziej zaawansowanego podejścia.

Proces rozpoczynaliśmy od binaryzacji obrazu z wykorzystaniem odpowiednio dobranego parametru X_i. Następnie wykonywaliśmy czyszczenie binarnego obrazu poprzez pozostawienie jedynie naj-

większego obiektu oraz zastosowanie morfologicznego domknięcia i otwarcia z dużym eliptycznym jądrem o rozmiarze 9×9. Jądro to miało na celu wygładzenie granic oraz wypełnienie niewielkich luk w strukturze tęczówki.

Po przygotowaniu oczyszczonego obrazu przechodziliśmy do analizy projekcji obrazu w czterech kierunkach: pionowym, poziomym, ukośnym pod kątem 45° oraz ukośnym pod kątem 135°. Projekcje te obliczaliśmy jako sumy wartości pikseli wzdłuż danego kierunku.

Poniżej przedstawiamy jak obraz zawierający tęczówkę zmieniał się podczas wyżej opisanych operacji jak również projekcje pionową i poziomą.

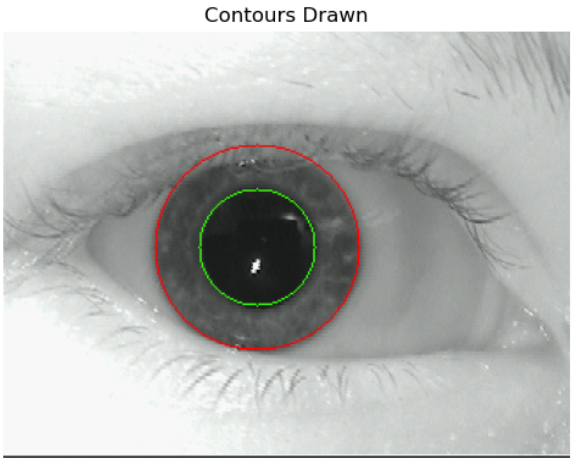


Rysunek 3: Proces czyszczenia: (lewo) największy kontur, (środek) zaawansowane funkcje i operacje morfologiczne, (prawo) wizualizacja projekcji pionowej i poziomej

Środek tęczówki oczywiście mieliśmy już obliczony jako środek źrenicy natomiast na podstawie obliczonych projekcji obliczaliśmy różnice sąsiadujących wartości projekcji w każdym z czterech kierunków, a następnie identyfikowaliśmy pozycje, w których występowały największe skoki wartości. Zidentyfikowane w ten sposób krawędzie interpretowaliśmy jako przybliżone granice tęczówki. Promienie wyznaczaliśmy osobno dla każdego kierunku jako połowę odległości pomiędzy skrajnymi krawędziami, a końcowy promień był średnią z czterech kierunków.

Takie podejście pozwalało zniwelować wpływ pojedynczych zakłóceń lub zasłoniętych fragmentów obrazu, co było szczególnie istotne przy analizie tęczówki w obecności powiek lub rzęs.

Ostatecznie otrzymaliśmy satysfakcjonujący wynik z bardzo dobrze dopasowanymi granicami źrenicy i tęczówki na zdjęciu. Poniżej przedstawiamy efekt końcowy.



Rysunek 4: Wizualizacja znalezionych granic źrenicy i tęczówki

2.3 Rozwinięcie tęczówki do prostokąta.

Po wyznaczeniu środka i promieni źrenicy oraz tęczówki, wykonaliśmy transformację współrzędnych w celu rozwinięcia pierścienia tęczówki do prostokątnego obrazu. Transformacja ta miała na celu uproszczenie dalszego przetwarzania obrazu poprzez przedstawienie okrągłej struktury w układzie prostokątnym.

Rozwinięcie polegało na przejściu z układu biegunowego (r, θ) do układu kartezjańskiego (x, y). Dla każdego punktu wyjściowego (x, y) w rozwiniętym obrazie (o zadanym rozmiarze, domyślnie 64×256), obliczaliśmy odpowiadające mu współrzędne w obrazie wejściowym zgodnie z poniższymi zależnościami:

$$\theta = \frac{2\pi x}{\text{width}}, \quad r = r_p + \left(\frac{y}{\text{height}}\right)(r_i - r_p) \tag{1}$$

$$x_i = c_x + r \cdot \cos(\theta), \quad y_i = c_y + r \cdot \sin(\theta) \tag{2}$$

gdzie:

- r_p — promień źrenicy,
- r_i — promień tęczówki,
- (c_x, c_y) — środek źrenicy,
- (x_i, y_i) — współrzędne w obrazie wejściowym,
- `width`, `height` — szerokość i wysokość rozwiniętego obrazu.

Dla każdego piksela w obrazie rozwiniętym pobieraliśmy odpowiadający mu piksel z obrazu wejściowego i przypisywaliśmy jego wartość. Operacja ta była wykonywana tylko wtedy, gdy obliczone współrzędne znajdowały się w granicach obrazu źródłowego.

Rysunek 5: Efekt rozwinięcia tęczówki do prostokąta.

Dzięki tej transformacji uzyskaliśmy prostokątny obraz rozwiniętej tęczówki, w którym każda kolumna odpowiada określonemu kątowi θ, a każdy wiersz odpowiada odległości od środka źrenicy. Obraz ten był gotowy do dalszego przetwarzania w procesie ekstrakcji cech.

2.4 Algorytm Daugmana

2.4.1 Transformacja falkowa Gabora

Do analizy tekstury tęczówki zastosowano transformację Gabora z następującymi parametrami:

$$G(x, y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)\exp(2\pi i f x) \tag{3}$$

parametry przyjęte przez nas podczas analizy:

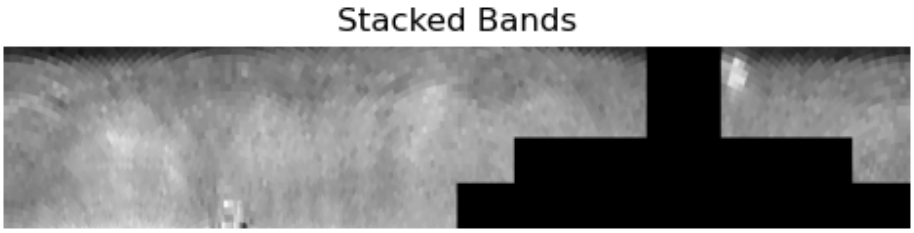
- $\sigma = \frac{1}{2}\pi f$ - zgodnie z zaleceniem
- $f = \pi$ - częstotliwość centralna

Filtr został zaimplementowany jako jądro o rozmiarze 128 pikseli (wielkość można dostosowywać), z orientacją radialną ($\theta = 0$)

2.4.2 Generowanie kodu tęczówki

Proces kodowania obejmuje:

1. Podział rozwiniętej tęczówki na 8 pasów radialnych
2. Wykluczenie 30-stopniowego sektora poniżej źrenicy (dla górnych pasów)
3. Ograniczenie analizy do sektorów 226° i 180° dla środkowych i dolnych pasów
4. Kwantyzacja odpowiedzi filtracji na 2 bity (znaki części rzeczywistej i urojonej)



Rysunek 6: Efekt wyłączenia odpowiednich części tęczówki z procesu kodowania.

Rysunek 7: Wynik kwantyfikacji tęczówki (16 wierszy po 128 bitów)

2.4.3 Porównywanie kodów

Porównanie kodów realizowane jest poprzez:

- Obliczenie odległości Hamminga z uwzględnieniem maski ważności bitów:

$$H = \frac{\sum (A \oplus B) \cap M_A \cap M_B}{\sum M_A \cap M_B} \tag{4}$$

- Testowanie przesunięć kątowych (± X pikseli) dla kompensacji rotacji oka
- Wybór minimalnej odległości spośród wszystkich testowanych przesunięć

3 Implementacja

3.1 Struktura kodu

System został zaimplementowany w języku Python z wykorzystaniem bibliotek:

- OpenCV - przetwarzanie obrazu
- NumPy - operacje macierzowe
- SciPy - filtracja sygnałów
- Matplotlib - wizualizacja wyników

Główne klasy:

- `IrisSegmenter` - klasa realizująca wykrywanie i rozwijanie tęczówki.
- `IrisRecognizer` - wyznaczanie kodu tęczówki, wyliczanie odległości Hamminga.

3.2 Parametry IrisSegmenter

Parametr
Parametr binaryzacji dla źrenicy
Parametr binaryzacji dla tęczówki

3.3 Parametry IrisRecognizer

Parametr	Użyta wartość
Rozmiar jądra Gabora	128
Częstotliwość centralna (f)	π
Maksymalne przesunięcie kątowe	8

4 Podsumowanie

Prace zakończyły się stworzeniem działającego algorytmu rozpoznawania tęczówki, który implementuje algorytm Daugmana. Kod jest w stanie wyodrębnić tęczówkę z obrazu oka, przekształcić jej teksturę na kod binarny i porównywać różne tęczówki z uwzględnieniem naturalnych przesunięć kątowych. Jednak rozwiązanie nie jest idealne. Głównym wyzwaniem pozostaje precyzyjne dostrojenie parametrów - częstotliwości filtrów Gabora, wielkości jąder czy progów binaryzacji. W obecnej wersji wymagają one ręcznego dopasowania. Klasy mogłyby też lepiej radzić sobie z różnymi warunkami oświetleniowymi czy artefaktami na obrazach. Mimo pewnych ograniczeń kod stanowi solidną podstawę, która po ewentualnych usprawnieniach mogłaby stać się pełnoprawnym systemem biometrycznym. Projekt pokazuje, że nawet stosunkowo prosta implementacja algorytmu Daugmana jest w stanie dać dobre rezultaty w rozpoznawaniu tęczówki.