Biometria Projekt 2 - Identyfikacja na podstawie obrazu tęczówki

Karolina Dunal i Zofia Kamińska

25 kwietnia 2025

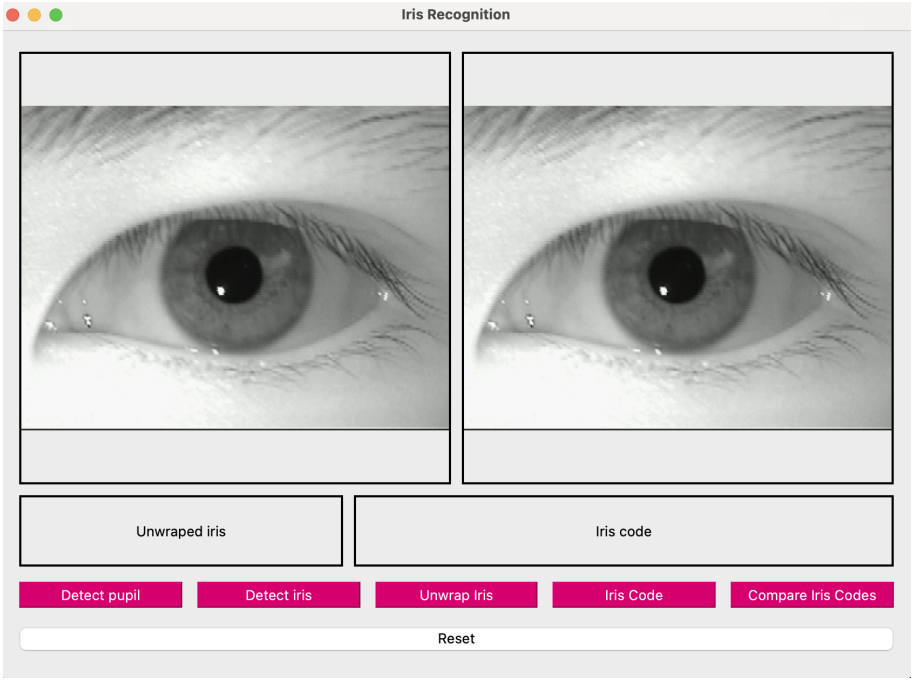
Spis treści

1 Cel projektu

Głównym celem projektu była analiza oraz identyfikacja tęczówki oka z wykorzystaniem technik biometrycznych. W celu ułatwienia korzystania z opracowanych rozwiązań oraz zapewnienia możliwości wizualnej prezentacji wyników, przygotowany został zestaw funkcji i metod umożliwiających realizację poszczególnych etapów przetwarzania. System ten pozwala m.in. na detekcję źrenicy i tęczówki, przekształcenie tęczówki do postaci spłaszczonej (rozwiniętej) oraz zastosowanie algorytmu Daugmana w celu ekstrakcji cech i identyfikacji na podstawie wzorca tęczówki.

2 Opis aplikacji

W celu ułatwienia korzystania z narzędzia oraz zapewnienia przejrzystej wizualizacji wyników, przygotowana została aplikacja okienkowa wyposażona w intuicyjny i prosty w obsłudze interfejs graficzny (Rysunek 1). Projekt został zrealizowany w języku *Python* z wykorzystaniem biblioteki *PyQt6* do stworzenia interfejsu graficznego. Do przetwarzania obrazów zastosowano bibliotekę *NumPy*, umożliwiającą szybkie operacje na macierzach, oraz *OpenCV* do wczytywania, zapisu i przetwarzania obrazów.



Rysunek 1: Zrzut ekranu aplikacji przedstawiający interfejs użytkownika.

3 Opis zaimplementowanych metod

3.1 Klasa ImageProcessor

3.1.1 Metoda `calculate_binarization_threshold`

Funkcja oblicza globalny próg binarizacji jako średnią jasność wszystkich pikseli obrazu w skali szarości. Wartość ta jest używana jako punkt odniesienia przy segmentacji źrenicy i tęczówki. Została zaimplementowana zgodnie z poniższym wzorem:

$$P = \frac{1}{h \cdot w} \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} A(i,j)$$

gdzie:

- P – globalny próg binarizacji,
- h – wysokość obrazu,
- w – szerokość obrazu,
- $A(i,j)$ – jasność piksela w skali szarości.

Na podstawie wyliczonego progu P, wartości progowe dla segmentacji źrenicy (P_P) oraz tęczówki (P_I) wyznacza się niezależnie według zależności:

$$P_P = \frac{P}{X_P}, \quad P_I = \frac{P}{X_I}$$

gdzie X_P i X_I są współczynnikami dobranymi eksperymentalnie.

3.1.2 Metoda `binarization`

Metoda binarizuje obraz na podstawie zadanego progu jasności. Piksele jaśniejsze od progu są ustawiane na 255, a ciemniejsze na 0. Wynik to obraz binarny używany do detekcji źrenicy oraz tęczówki.

3.1.3 Metoda `detect_pupil`

Funkcja służy do wykrywania źrenicy na obrazie z wykorzystaniem metod progowania i operacji morfologicznych. Na początku, jeśli obraz nie został przekazany jako argument, funkcja wykorzystuje domyślny obraz zapisany w obiekcie. Obliczany jest próg binarizacji, który następnie dzielony jest przez parametr `threshold_pupil`, ustawiony domyślnie na 4.1, co pozwala uzyskać odpowiedni poziom progowania. Obraz jest następnie binaryzowany oraz filtrowany medianowo w celu redukcji szumów.
W dalszym etapie wykonywane są operacje morfologiczne, w następującej kolejności:

1. **Erozja** prostokątnym elementem strukturalnym o rozmiarze 3×3, powtórzona dwukrotnie.
2. **Otwarcie** z użyciem prostokątnego jądra o wymiarach 10×1.
3. **Dylatacja** prostokątnym jądrem 3×3, powtórzona trzykrotnie.
4. **Zamknięcie** eliptycznym jądrem 5×5.
5. **Dylatacja** eliptycznym jądrem 5×5.

Po tych operacjach analizowane są projekcje poziome i pionowe, które pozwalają na wyznaczenie przybliżonego położenia środka źrenicy oraz oszacowanie jej rozmiaru na podstawie różnic między maksymalnymi i minimalnymi wartościami tych projekcji. Funkcja zwraca zbinaryzowany obraz źrenicy, współrzędne jej środka oraz promień.

3.1.4 Metoda `estimate_iris_radius_binary`

Funkcja służy do oszacowania promienia tęczówki na podstawie obrazu binarnego. Przyjmuje jako argumenty binarny obraz oka, współrzędne środka źrenicy, promień źrenicy oraz maksymalny promień, którego nie należy przekraczać. Dla kolejnych promieni, zaczynając od promienia źrenicy, funkcja wyznacza współrzędne punktów na okręgu o danym promieniu i sprawdza, jaki procent z nich należy do białych pikseli. Jeśli stosunek białych pikseli przekroczy zadany próg (`white_threshold_ratio`, domyślnie ustawiony na 0.5), to dany promień uznaje się za promień tęczówki. Jeśli taki promień nie zostanie znaleziony w zadanym zakresie, zwracana jest wartość `max_r`.

3.1.5 Metoda `detect_iris`

Funkcja służy do wykrywania obszaru tęczówki w obrazie oka z wykorzystaniem progowania oraz operacji morfologicznych. Najpierw obliczany jest próg binarizacji, który następnie dzielony jest przez wartość `threshold_iris`, ustawiony domyślnie na 1.3, co pozwala uzyskać obraz binarny z zaznaczonym obszarem potencjalnej tęczówki. Obraz ten jest wygładzany przy pomocy medianowego filtra.
Następnie wykonywana jest seria operacji morfologicznych, których celem jest usunięcie zakłóceń i wzmocnienie konturów:

1. **Dylatacja** z użyciem eliptycznego jądra 5×5.
2. **Otwarcie** prostokątnym jądrem 1×10, powtórzone dwukrotnie.
3. **Otwarcie** prostokątnym jądrem 10×1, również powtórzone dwukrotnie.
4. **Zamknięcie** eliptycznym jądrem 5×5, wykonane jedenastokrotnie.
5. **Otwarcie** eliptycznym jądrem 3×3.
6. **Dylatacja** eliptycznym jądrem 3×3, wykonana trzykrotnie.

Po zakończeniu przetwarzania morfologicznego wykonywane jest wykrycie źrenicy za pomocą funkcji `detect_pupil`, w celu dokładniejszego wyznaczenia środka tęczówki. Promień tęczówki obliczany jest za pomocą funkcji `estimate_iris_radius_binary`, ponieważ bazowanie wyłącznie na zbinaryzowanym obrazie po operacjach morfologicznych nie pozwala na uzyskanie wystarczającej dokładności. Obecność cieni, powiek lub jasnej barwy tęczówki często prowadzi do tego, że wynikowy obraz zawiera także obszary nienależące do samej tęczówki, przez co jej kształt zostaje zniekształcony. Funkcja zwraca zbinaryzowany obraz tęczówki, współrzędne jej środka oraz promień.

3.1.6 Metoda `unwrap_iris`

Metoda rozwija obraz tęczówki do formatu prostokątnego, używając metody `normalize_iris` opsianej w sekcji 3.2.1 poniżej. Normalizuje ona również wartości pikseli w celu uzyskania lepszej reprezentacji, z użyciem skalowania *min-max*. Funkcja zwraca obraz tęczówki w formie 2D.

3.1.7 Metoda `rings_division`

Metoda wykorzystuje funkcje `detect_pupil` (sekcja 3.1.3) i `detect_iris` (sekcja 3.1.5) do wyznaczenia lokalizacji źrenicy oraz tęczówki. Następnie na podstawie tych informacji wywołuje metodę `create_iris_code` (sekcja 3.2.4), aby wygenerować kod tęczówki. Zwraca zakodowaną reprezentację tęczówki w postaci binarnej.

3.2 Klasa IrisRingDivider

Klasa `IrisRingDivider` została zaprojektowana do przetwarzania obrazu oka w celu wyodrębnienia kodu tęczówki. Jej konstruktor przyjmuje obraz oka oraz parametry geometryczne: współrzędne środka i promienie źrenicy oraz tęczówki. Jeśli dostarczony obraz jest w kolorze, zostaje automatycznie przekształcony do odcieni szarości w celu dalszej analizy.

3.2.1 Metoda `normalize_iris`

Normalizuje obraz tęczówki poprzez przekształcenie współrzędnych biegunowych na prostokątne. W wyniku uzyskujemy obraz 2D reprezentujący spłaszczoną tęczówkę. W celu uniknięcia zakłóceń spowodowanych przez powieki i rzęsy, stosowane są maski kątowe zależne od indeksu pierścienia.

3.2.2 Metoda `compute_valid_angles_for_radius`

Oblicza dopuszczalne kąty dla danego promienia r. Obraz dzielony jest na 8 pierścieni, z których każdy ma przypisane blokowane kąty:

- Pierścienie 0–3: blokowane kąty $\theta \in [75°, 105°]$,
- Pierścienie 4–5: blokowane kąty $\theta \in [56{,}5°, 123{,}5°] \cup [236{,}5°, 303{,}5°]$,
- Pierścienie 6–7: blokowane kąty $\theta \in [45°, 135°] \cup [225°, 315°]$.

Blokowanie określonych zakresów kątów θ wynika z położenia górnej i dolnej powieki w biegunowym układzie współrzędnych względem źrenicy.

3.2.3 Metoda `gabor_filter_1d`

Pozwala zastosować filtr Gabora do wejściowego sygnału.
Filtr Gabora definiuje się jako:

$$g(x) = \exp\left(-\frac{x^2}{\sigma^2}\right) \cdot \cos(2\pi f x)$$

gdzie f to częstotliwość, a σ kontroluje rozmycie (rozpiętość) funkcji. Odpowiedź zespolona filtru składa się z części rzeczywistej (cosinusowej) oraz urojonej (sinusowej):

$$g_{\text{real}}(x) = \exp\left(-\frac{x^2}{\sigma^2}\right)\cos(2\pi f x), \quad g_{\text{imag}}(x) = \exp\left(-\frac{x^2}{\sigma^2}\right)\sin(2\pi f x)$$

Sygnał filtrowany jest osobno obydwoma wersjami, a wynikowe odpowiedzi służą do stworzenia kodu tęczówki.

3.2.4 Metoda `create_iris_code`

Metoda tworzy binarny kod reprezentujący cechy tęczówki. Na początku obraz jest normalizowany i dzielony na 8 poziomych pasów. Dla każdego z pasów przeprowadzane jest wygładzanie Gaussa w kierunku radialnym, po czym dane są przetwarzane za pomocą jednowymiarowego filtru Gabora. Następnie wyznaczane są odpowiedzi rzeczywista i urojona, które są dzielone na 128 równych segmentów kątowych. Dla każdego segmentu obliczana jest wartość średnia. Dzięki temu rozwiązaniu kod cech charakteryzuje się większą odpornością na błędy wynikające z jakości obrazu, jego obrotu czy sposobu wykonania zdjęcia.
Średnie wartości sygnału są następnie binarnie kodowane:

- 1 – jeśli wartość średnia jest dodatnia,
- 0 – jeśli wartość średnia jest niedodatnia.

Ostatecznie powstaje kod o wymiarach $8 \times 256 \times 1$, zawierający naprzemiennie bity pochodzące z odpowiedzi rzeczywistej i urojonej.

3.2.5 Metoda `calculate_hamming_distance`

Oblicza dystans Hamminga pomiędzy dwoma kodami tęczówki. Miara ta określa różnice pomiędzy odpowiadającymi sobie bitami i opisana jest wzorem:

$$HD = \frac{1}{N} \sum_{i=1}^{N} \left(c_i^{(1)} \oplus c_i^{(2)} \right)$$

gdzie:

- HD – dystans Hamminga,
- $c_i^{(1)}$, $c_i^{(2)}$ – odpowiadające sobie bity dwóch kodów,
- N – całkowita liczba bitów.

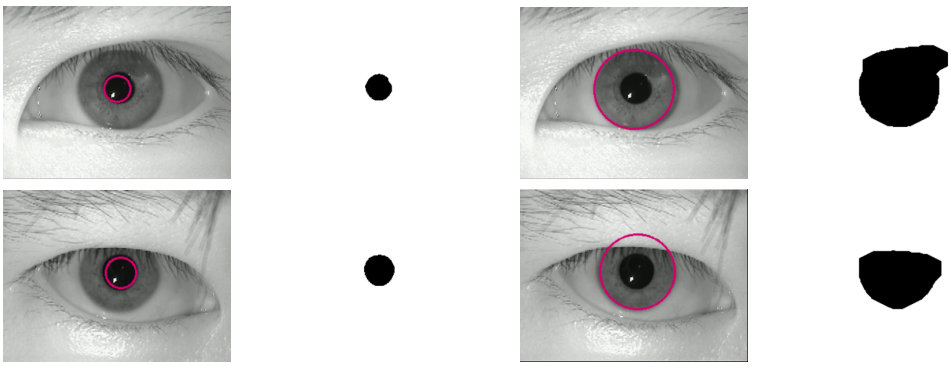
Zwracana wartość to stosunek liczby różnych bitów do liczby wszystkich bitów — im mniejsza wartość, tym większe podobieństwo między kodami.

4 Przykłady i wnioski

Do oceny skuteczności zastosowanych metod, doboru progów i parametrów dla niektórych z metod oraz przeprowadzenia przykładowej wizualizacji wyników wykorzystano obrazy pochodzące ze zbioru *MMU Iris Database*, który został pobrany z platformy *Kaggle*.

4.1 Detekcja źrenicy i tęczówki

Poniżej przedstawiono wyniki detekcji źrenicy oraz tęczówki (Rysunek 2), uzyskane przy wykorzystaniu operacji morfologicznych. W przypadku detekcji tęczówki poza operacjami morfologicznymi zastosowano również podejście polegające na analizie wartości binarnego obrazu wzdłuż okręgów o rosnącym promieniu, poczynając od promienia źrenicy. Dla każdego okręgu sprawdzana jest liczba jasnych pikseli, a proces kończy się w momencie, gdy ich udział przekroczy ustalony próg. Promień odpowiadający temu okręgowi przyjmowany jest jako promień tęczówki. Metoda ta znacząco poprawia skuteczność wykrywania tęczówki, ponieważ, jak można zaobserwować na poniższych przykładach, wynik binarnej segmentacji nie zawsze dokładnie odwzorowuje rzeczywisty kształt tęczówki. Obecność refleksów świetlnych, cieni oraz jasna barwa tęczówki mogą powodować jej znaczne zniekształcenie w obrazie binarnym, co sprawia, że bazowanie wyłącznie na operacjach morfologicznych okazuje się niewystarczające.

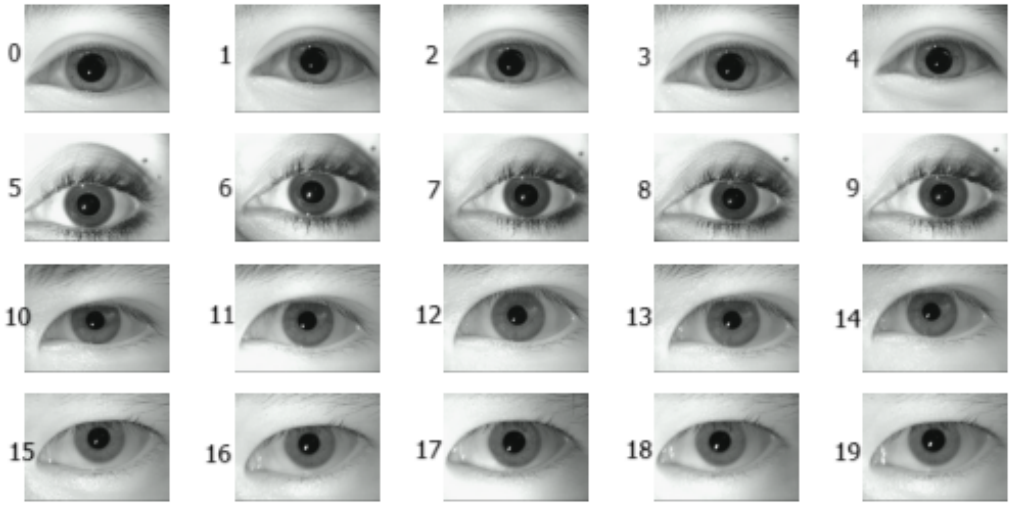


Rysunek 2: Przykładowe wyniki detekcji źrenicy oraz tęczówki wraz z odpowiadającymi im zbinaryzowanymi wynikami operacji detekcji.

4.2 Analiza tęczówki

Poniżej zaprezentowano przykłady różnych obazów oczu, na podstawie których dostosowywane były odpowiednie parametry oraz algorytmy.
W celu ułatwienia dalszej analizy (porównywania kodów tęczówek i tworzenia wizualizacji), obrazy zostały oznaczone indeksami od 0 do 19 zgodnie z kolejnością ich rozmieszczenia w grafice (Rysunek 3) — wierszami, od lewej do prawej. Przypisanie indeksów do danych wygląda następująco:

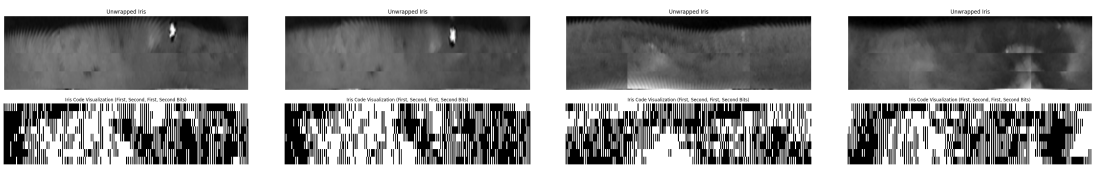
- indeksy 0–4: *33 left*,
- indeksy 5–9: *9 left*,
- indeksy 10–14: *29 left*,
- indeksy 15–19: *39 left*.



Rysunek 3: Przykładowe zdjęcia oczu wykorzystane w analizie (odczytywane wierszami, od lewej do prawej).

4.2.1 Rozwinięcie oraz kod tęczówki

Poniżej przedstawiono przykłady rozwinięć oraz odpowiadających im kodów tęczówki dla różnych oczu, na podstawie zdjęć o indeksach 0, 1, 5 i 10 (od lewej). Pierwsze dwa obrazy (indeksy 0 i 1) przedstawiają to samo oko, natomiast kolejne dwa (indeksy 5 i 10) pochodzą od różnych osób, co przekłada się na wyraźne różnice zarówno w strukturze tęczówki, jak i w generowanych kodach binarnych.
Co istotne, nawet w przypadku tego samego oka (jak w parze 0 i 1) możliwe są różnice pomiędzy wygenerowanymi kodami, wynikające m.in. z różnych warunków obrazowania.

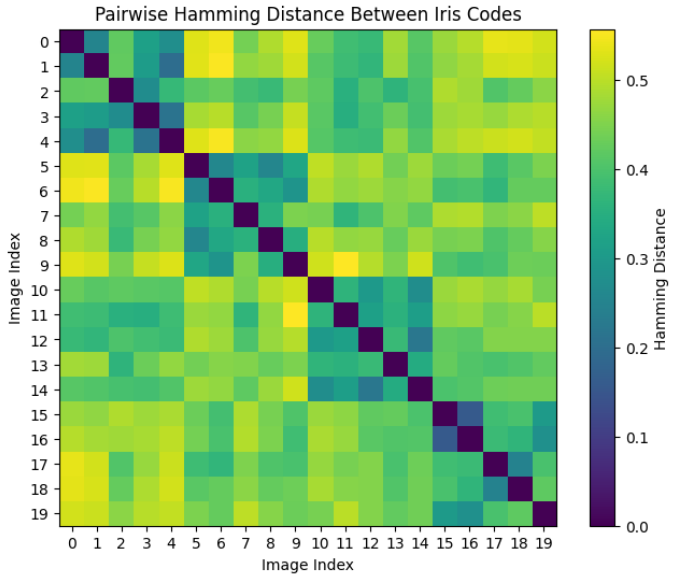


Rysunek 4: Rozwinięcia (góra) oraz odpowiadające im kody binarne tęczówek (dół) dla czterech obrazów o indeksach 0, 1, 5 i 10 (od lewej).

Warto zaznaczyć, że mimo starannej segmentacji, w niektórych przypadkach mogą pozostać artefakty – takie jak rzęsy czy fragmenty powiek – które mogą zakłócać analizę i wpływać na ostateczny kod tęczówki.

4.2.2 Porównanie odległości Hamminga

Poniżej przedstawiono macierz odległości Hamminga obliczonych pomiędzy wszystkimi parami kodów tęczówki dla zestawu obrazów opisanych na samym początku tej sekcji.



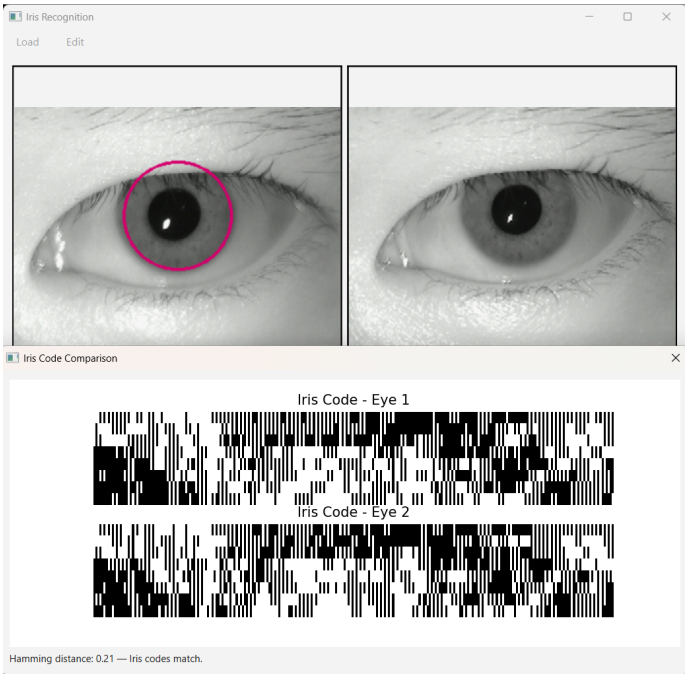
Rysunek 5: Macierz odległości Hamminga pomiędzy kodami tęczówek.

Jak można zaobserwować, istnieje tendencja do niższych wartości odległości Hamminga pomiędzy obrazami pochodzącymi od tej samej osoby (z tego samego oka). Nie jest to jednak reguła absolutna. Występują przypadki, w których odległości pomiędzy kodami tej samej osoby są wyższe niż oczekiwano. Może to wynikać z różnych czynników, takich jak jakość obrazu, zmienność ujęcia czy niedoskonałości w detekcji granic źrenicy i tęczówki.
Na podstawie przeprowadzonych testów i obserwacji przyjęto próg odległości Hamminga równy 0,26 jako granicę decyzyjną, powyżej której dwa obrazy uznawane są za pochodzące od różnych osób. Próg ten został dobrany empirycznie na podstawie analizy wielu przypadków i stanowi kompromis pomiędzy dokładnością a tolerancją błędu. Warto zaznaczyć, że próg ten należy traktować jako orientacyjny. W praktyce, ze względu na znaczną zmienność w wyglądzie i jakości obrazów, nawet kody pochodzące od tego samego oka mogą niekiedy nie spełniać tego warunku, mimo wizualnego podobieństwa.
Na pełnym zbiorze danych *MMU Iris Dataset* przeprowadzono analizę metryk klasyfikacyjnych, takich jak dokładność (accuracy), precyzja (precision), czułość (recall) oraz miara F1. Uzyskane wyniki przedstawiają się następująco:

- **Accuracy:** 98.12%
- **Precision:** 94.20%
- **Recall:** 6.42%
- **F1-score:** 12.02%

Wysoka dokładność oraz precyzja wskazują, że algorytm unika błędnych dopasowań, co świadczy o jego odporności na fałszywe pozytywy. Z kolei relatywnie niskie wartości czułości oraz miary F1 są rezultatem przyjęcia rygorystycznego progu klasyfikacyjnego. Ma to na celu zwiększenie odporności systemu na błędną klasyfikację, więc nawet niewielkie zmiany w położeniu źrenicy, oświetleniu czy jakości obrazu mogą wpłynąć na zaklasyfikowanie poprawnych dopasowań jako negatywnych. Pomimo spadku tych metryk, taki kompromis pozwala na zwiększenie niezawodności.

Poniżej przedstawiono przykład działania interfejsu użytkownika po kliknięciu przycisku *Compare Iris Codes*. Po lewej stronie widoczne jest aktualnie analizowane oko, a po prawej obraz wybrany do porównania. Dodatkowo wyświetlane jest okno z zestawionymi kodami tęczówek oraz informacją, czy zostały one rozpoznane jako pochodzące od tego samego oka.



Rysunek 6: Przykład działania interfejsu użytkownika: porównanie dwóch kodów tęczówek (zdjęcia indeksowane 17 (z lewej) i 18 (z prawej)). Wynik opiera się na porównaniu wartości odległości Hamminga względem ustalonego progu 0,26.

5 Podsumowanie

Przy realizacji projektu można było zauważyć, że pewne cechy obrazów znacząco utrudniają skuteczną segmentację oka. Refleksy świetlne, bardzo jasny kolor tęczówek, obecność rzęs i brwi, często nachodzących na tęczówkę, wpływają negatywnie na wyniki przetwarzania i detekcji. Problemy pojawiają się również podczas rozwijania tęczówki. W wielu przypadkach jej część pozostaje zasłonięta przez powiekę, co zniekształca wynik. Różne pozycje oka względem kamery oraz jego ułożenie również wprowadzają trudności w standaryzacji analizy. Pomimo napotkanych trudności związanych z różnorodnością obrazów i ich cechami, metody zastosowane w projekcie pozwoliły na skuteczną detekcję źrenicy oraz tęczówki, osiągając wyniki zadowalające pod względem jakości oraz precyzji analizy.

6 Źródła

- MMU Iris Database, `https://www.kaggle.com/datasets/naureenmohammad/mmu-iris-dataset`
- dr inż. J. Rafałko, *Wykłady z Biometrii*, Politechnika Warszawska, 2025
- OpenAI, ChatGPT (GPT-4), `https://openai.com/chatgpt`