Biometria Projekt 3 - Porównanie algorytmów ścieniania w odniesieniu do odcisków palców

Karolina Dunal i Zofia Kamińska

28 maja 2025

Spis treści

1 Cel projektu **2**

2 Opis zaimplementowanych metod **2**
2.1 Struktura klasy `FingerprintProcessor` 2
2.2 Ogólny przebieg przetwarzania obrazu 2

3 Szczegółowy opis implementacji poszczególnych etapów **3**
3.1 Normalizacja obrazu 3
3.2 Segmentacja obrazu 4
3.3 Mapy kierunków i częstotliwości 5
3.3.1 Mapa kierunków 5
3.3.2 Mapa częstotliwości 6
3.4 Filtracja Gaborowska 6
3.4.1 Idea działania 6
3.4.2 Etapy implementacji 6
3.5 Szkieletyzacja 7
3.5.1 Szkieletyzacja morfologiczne 7
3.5.2 Szkieletyzacja KMM 8
3.6 Ekstrakcja cech: Minuncje 10

4 Przykłady i wnioski **11**
4.1 Porównanie metod szkieletyzacji 11
4.2 Porównanie minucji 11
4.3 Liczba wykrytych minucji 12

5 Detekcja minucji z kierunkami **13**
5.1 Wnioski i podsumowanie 13

6 Źródła **14**

1 Cel projektu

Celem projektu jest porównanie skuteczności dwóch algorytmów ścieniania (ang. thinning) w kontekście przetwarzania obrazów odcisków palców. W ramach pracy zaimplementowane zostały dwa podejścia:

- algorytm szkieletyzacji morfologicznej, omówiony na wykładzie,
- algorytm KMM.

Dodatkowo, w celu poprawy jakości przetworzonych obrazów, zostały zastosowane odpowiednie operacje morfologiczne, mające na celu m.in. uzupełnienie przerwanych linii papilarnych oraz poprawę ciągłości ścieżek.
Na podstawie tak przygotowanych danych dokonana została lokalizacja wybranych minucji, czyli charakterystycznych punktów linii papilarnych, takich jak:

- zakończenia linii,
- rozwidlenia linii (bifurkacje).

2 Opis zaimplementowanych metod

Główna logika przetwarzania obrazów odcisków palców została zaimplementowana w klasie `FingerprintProcessor`, której zadaniem jest przeprowadzenie pełnego łańcucha operacji: od wczytania obrazu po ekstrakcję minucji. Każdy etap przetwarzania odpowiada jednej lub kilku metodom klasy, a ich kolejność odzwierciedla standardowy pipeline stosowany w analizie biometrycznej linii papilarnych.

2.1 Struktura klasy `FingerprintProcessor`

Klasa zawiera zestaw atrybutów przechowujących dane z kolejnych etapów przetwarzania:

- `raw_image` – oryginalny obraz wczytany z pliku,
- `normalized_image` – obraz po normalizacji jasności i kontrastu,
- `segmented_image` – obraz po segmentacji, zawierający tylko obszar odcisku,
- `roi_mask` – maska regionu zainteresowania (ROI),
- `orientations` – mapa kierunkowości linii papilarnych,
- `freq_map` – mapa częstotliwości grzbietów,
- `filtered_image` – obraz po filtracji Gaborowskiej,
- `skeleton` – obraz po szkieletyzacji (ścienianiu).

2.2 Ogólny przebieg przetwarzania obrazu

Proces analizy obrazu odbywa się w następujących krokach:

1. **Wczytanie obrazu:** metoda `load_image` ładuje obraz w skali szarości i inicjalizuje kolejne etapy przetwarzania.
2. **Normalizacja:** metoda `normalize_fingerprint` dostosowuje jasność i kontrast obrazu do zadanych wartości średniej i wariancji.
3. **Segmentacja:** `fingerprint_segmentation` izoluje obszar linii papilarnych od tła oraz generuje zaktualizowany obraz do dalszego przetwarzania.
4. **Mapa kierunkowości:** `create_directional_map` oblicza lokalne kierunki grzbietów w blokach 16x16 pikseli, a następnie wizualizuje je na obrazie.
5. **Mapa częstotliwości:** `frequency_map` estymuje lokalną częstotliwość linii papilarnych oraz oblicza wartość mediany tych częstotliwości.

6. **Filtracja Gaborowska:** `apply_gabor_filter` wzmacnia struktury grzbietowe, dopasowując parametry filtrów do lokalnej orientacji i częstotliwości.
7. **Ścienianie (Szkieletyzacja):** zastosowano dwie metody:
   - `morphological_skeletonization` – bazująca na operacjach morfologicznych,
   - `skeletonize_with_kmm` – implementująca algorytm KMM.

   Obie metody generują szkielety linii papilarnych.
8. **Ekstrakcja minucji:** `detect_minutiae` wykrywa zakończenia i rozwidlenia grzbietów na obrazie szkieletowym, a następnie wizualizuje je.

3 Szczegółowy opis implementacji poszczególnych etapów

W tej sekcji przedstawiono szczegółowy opis zaimplementowanych metod przetwarzania obrazu odcisku palca. Każdy podrozdział odpowiada konkretnemu krokowi w pipeline.

Rysunek 1: Przykład obrazu przed jakimikolwiek zmianami.

3.1 Normalizacja obrazu

Celem normalizacji jest ujednolicenie jasności oraz kontrastu obrazu poprzez przekształcenie wartości pikseli tak, aby wynikowy obraz posiadał zadaną średnią oraz wariancję. Ma to na celu zniwelowanie wpływu zmiennych warunków oświetleniowych oraz poprawę działania kolejnych kroków algorytmu.
Normalizacja została zaimplementowana w module `normalization.py` i odbywa się w dwóch etapach:

1. Obraz wejściowy konwertowany jest do skali szarości (jeśli nie jest już jednowymiarowy),
2. Każdy piksel poddawany jest przekształceniu według wzoru zależnego od globalnej średniej i wariancji obrazu oraz zadanych wartości docelowych.

Normalizacja została zaimplementowana w fukcji `normalize_image(image, desired_mean, desired_var)`. Dla każdego piksela (x, y) obliczany jest nowy poziom jasności:

$$I_{norm}(x,y) = \begin{cases} \mu_d + \sqrt{\frac{\sigma_d^2 \cdot (I(x,y)-\mu)^2}{\sigma^2}}, & \text{jeśli } I(x,y) > \mu \\ \mu_d - \sqrt{\frac{\sigma_d^2 \cdot (I(x,y)-\mu)^2}{\sigma^2}}, & \text{w przeciwnym razie} \end{cases}$$

gdzie:

- μ_d, σ_d^2 – zadana średnia i wariancja (np. 100 i 100),
- μ, σ^2 – średnia i wariancja oryginalnego obrazu,
- $I(x, y)$ – wartość piksela w obrazie wejściowym.

Podejście to różni się od prostego skalowania histogramu tym, że każdemu pikselowi przypisywana jest nowa wartość niezależnie, co może skutkować lepszym wyrównaniem kontrastu w lokalnych obszarach.

Rysunek 2: Przykład obrazu po zastosowaniu normalizacji (zmiana mało widoczna z uwagi na początkowo dobry kontrast).

3.2 Segmentacja obrazu

Celem segmentacji jest wyodrębnienie regionu zawierającego rzeczywisty obraz linii papilarnych z tła, które może zawierać szumy, marginesy lub inne nieistotne elementy. Proces ten znacząco redukuje liczbę analizowanych danych.
Segmentacja została zaimplementowana w funkcji `perform_segmentation(img, block_size=16, threshold_ratio=0.35)`. Główne kroki algorytmu to:

1. **Podział obrazu na bloki**: Obraz dzielony jest na bloki o rozmiarze 16×16 pikseli (domyślnie), co pozwala na analizę lokalnych właściwości.
2. **Obliczanie lokalnej wariancji**: Dla każdego bloku obliczana jest odchylenie standardowe pikseli. Bloki o zbyt małym zróżnicowaniu (np. obszary jednolite) uznawane są za tło.
3. **Utworzenie maski ROI**: Na podstawie lokalnych wariancji tworzona jest binarna maska obszaru zainteresowania (ROI) — piksele tła ustawione są na 0, a właściwego obszaru odcisku na 1.
4. **Wygładzenie maski**: W celu usunięcia drobnych artefaktów i spójnego zamknięcia obszarów, maska ROI jest przetwarzana morfologicznie (operacje `open` i `close` z użyciem eliptycznego elementu strukturalnego).
5. **Zastosowanie maski**: Oryginalny obraz jest mnożony przez maskę ROI, co skutkuje usunięciem tła (piksele poza ROI przyjmują wartość 0).
6. **Normalizacja ROI**: Obraz jest dodatkowo normalizowany (do średniej 0 i odchylenia standardowego 1), lecz tylko w obszarze ROI. Tło również zostaje ujednolicone przez przeskalowanie względem jego własnych statystyk.

Zwracane wyniki:

- `segmented` – obraz po zastosowaniu maski ROI (tło usunięte),
- `normalized` – obraz znormalizowany, uwzględniający różnicę między tłem a liniami papilarnymi,
- `roi_mask` – binarna maska obszaru zawierającego odcisk.

Rysunek 3: Przykład obrazu po zastosowaniu segmentacji.

3.3 Mapy kierunków i częstotliwości

Po segmentacji obrazu kolejnym etapem jest wyznaczenie lokalnej orientacji oraz częstotliwości grzbietów linii papilarnych. Te dwie cechy są niezbędne do późniejszego filtrowania Gaborowskiego oraz ekstrakcji cech. Zostały one zaimplementowane w pliku `mapping.py`.

3.3.1 Mapa kierunków

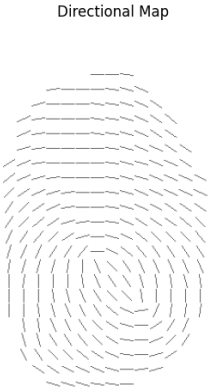
Lokalna orientacja grzbietów wyznaczana jest metodą opartą na gradientach Sobela. Obraz dzielony jest na bloki (domyślnie 16×16), a dla każdego z nich obliczany jest kąt kierunku linii papilarnych w oparciu o uśrednione wartości gradientów G_x i G_y.

- **Gradienty Sobela** – na obrazie stosuje się maski Sobela do wyznaczenia pochodnych w kierunku poziomym (G_x) i pionowym (G_y).
- **Obliczenie kąta** – dla każdego bloku wyznaczany jest kąt θ za pomocą wzoru:

$$\theta = \frac{1}{2} \arctan\left(\frac{2\sum G_x G_y}{\sum G_x^2 - \sum G_y^2}\right) + \frac{\pi}{2}$$

- **Wygładzanie kierunków (opcjonalne)** – na koniec możliwe jest zastosowanie wygładzania mapy kątów przy użyciu filtru Gaussowskiego i wektorowego uśredniania (operacje na $\cos(2\theta)$ i $\sin(2\theta)$).

Wizualizacja mapy orientacji odbywa się poprzez rysowanie linii w blokach na osobnym obrazie RGB. Linie są rysowane jedynie w miejscach, które należą do maski ROI.



Rysunek 4: Przykład mapy kierunków

3.3.2 Mapa częstotliwości

Lokalna częstotliwość odpowiada za szacowanie gęstości linii papilarnych w poszczególnych blokach obrazu. Została zaimplementowana metodą projekcji i detekcji szczytów.

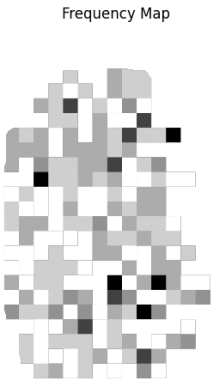
- **Obrót bloku** – dla każdego bloku obraz jest obracany tak, aby linie grzbietów były ustawione pionowo. Obrót wykonywany jest o kąt równy lokalnej orientacji plus 90°.
- **Projekcja i detekcja pików** – po obróceniu wykonuje się projekcję intensywności wzdłuż kolumn i identyfikuje lokalne maksima (szczyty), odpowiadające liniom grzbietów.
- **Obliczenie długości fali** – odległość między kolejnymi szczytami służy do wyznaczenia długości fali, a następnie częstotliwości:

  $$f = \frac{1}{\lambda}$$

  tylko jeśli λ mieści się w dopuszczalnym zakresie (np. od 3 do 20 pikseli).

Zwracane są:

- `frequency_map` – dwuwymiarowa mapa częstotliwości grzbietów,
- `median_frequency` – globalna wartość mediany częstotliwości w obrazie.



Rysunek 5: Przykład mapy częstotliwości

Zarówno mapa kierunków, jak i częstotliwości są niezbędne do adaptacyjnego filtrowania obrazu przy użyciu filtrów Gabora.

3.4 Filtracja Gaborowska

Filtracja Gaborowska jest stosowana w celu wzmocnienia struktury grzbietów linii papilarnych. Została zaimplementowana w pliku `filtering.py`.

3.4.1 Idea działania

Filtry Gabora są używane jako pasmowo-przepustowe filtry kierunkowe, które dostosowują się do lokalnej orientacji oraz częstotliwości grzbietów. W praktyce filtr o odpowiednim kącie i częstotliwości jest nakładany na blok obrazu, co skutkuje podkreśleniem struktur zgodnych z lokalną orientacją i częstotliwością.

3.4.2 Etapy implementacji

1. **Generacja bazowego filtru Gaborowskiego**: Na podstawie częstotliwości lokalnej lub — w przypadku jej niepewności, np. gdy nie mieści się w przyjętym zakresie lub nie może zostać wiarygodnie oszacowana — częstotliwości globalnej obliczany jest filtr Gabora w postaci:

   $$G(x, y) = e^{-\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)} \cdot \cos(2\pi f x)$$

   gdzie f to częstotliwość, a σ_x, σ_y są parametrami rozmycia zależnymi od częstotliwości.

2. **Rotacja filtrów**: Filtr bazowy jest obracany co $\Delta = 5^\circ$ w zakresie $[0^\circ, 180^\circ)$, co daje 36 filtrów o różnych orientacjach. Dzięki temu można szybko dopasować filtr do lokalnej orientacji bloku.

3. **Filtracja bloków**: Obraz jest przetwarzany blokowo. Dla każdego bloku:

   - Sprawdzane jest, czy blok należy do obszaru ROI,
   - Obliczany jest indeks filtru odpowiadającego lokalnej orientacji,
   - Wybierany jest odpowiedni filtr Gabora,
   - Stosowana jest operacja splotu z wykorzystaniem funkcji `cv.filter2D`.

4. **Normalizacja i binarizacja**: Po zakończeniu cały obraz jest normalizowany do zakresu $[0, 255]$ i binaryzowany metodą Otsu. Aby uzyskać standardowy zapis (linie jako czarne, tło jako białe), wykonywana jest operacja negacji bitowej.

Końcowym wynikiem jest binarny obraz, w którym linie papilarne są wyraźnie zaznaczone.

Rysunek 6: Przykład obrazu po zastosowaniu filtrowania.

3.5 Szkieletyzacja

Celem szkieletyzacji jest przekształcenie binarnego obrazu z grzbietami linii papilarnych w ich jedno-pikselową reprezentację – szkielet. Taka forma danych pozwala na precyzyjne wyodrębnienie cech, takich jak zakończenia linii czy rozwidlenia. W pliku `skeletonization.py` zaimplementowano kilka podejść do tego procesu.

3.5.1 Szkieletyzacja morfologiczne

Funkcja `morphological_skeleton` przeprowadza morfologiczną szkieletację binarnego obrazu, wykorzystując iteracyjne operacje erozji i otwarcia. Działa zgodnie z klasycznym podejściem poznanym na wykładzie.

- W przypadku podania maski, szkieletacja wykonywana jest tylko dla wskazanego przez maskę obszaru.
- Iteracyjnie wykonywane są operacje:
  1. Erozja bieżącego obrazu.
  2. Otwarcie tego samego obrazu.
  3. Odjęcie otwarcia od erozji i dodanie wyniku do szkieletu.
- Proces powtarzany jest do momentu, w którym pełna erozja wyzeruje obraz.
- W celu dodatkowego wyszczuplenia szkieletu, na końcu wykonywane jest otwarcie niewielkim elementem strukturalnym (`2x2` typu `MORPH_CROSS`), a jego wynik odejmowany jest od szkieletu powstałego po szkieletyzacji opisanej w poprzednim punkcie.

- Wynik jest negowany, aby zachować konwencję: tło białe, grzbiety czarne.

Niestety, metoda ta nie gwarantuje uzyskania szkieletu jednopikselowej grubości (co możemy zauważyć na Rysunku 7), co może prowadzić do błędów w dalszej analizie topologicznej (np. wykrywaniu bifurkacji). W celu ograniczenia grubości szkieletu i usunięcia artefaktów zastosowano dodatkowy krok polegający na odjęciu od uzyskanego szkieletu jego wersji po dodatkowej operacji otwarcia, jak opisano wyżej.

Rysunek 7: Przykład obrazu po szkieletyzacji morfologicznej.

3.5.2 Szkieletyzacja KMM

Zaawansowana metoda szkieletyzacji oparta na algorytmie opisanym w pracy *Algorytm do Ścieniania Obrazów: Implementacja i Zastosowania* (szczegóły pracy w skecji Źródła) została zaimplementowana jako `kmm_skeletonize`. Bazuje ona na funkcji `run_single_pass` i działa według następujących kroków:

1. **Normalizacja binarna:** Obraz przekształcany jest do postaci zawierającej jedynie wartości 0 i 1.
2. **Oznaczanie sąsiedztwa:** Każdy piksel typu 1 otrzymuje etykietę w zależności od położenia względem tła (np. piksel sąsiadujący z tłem ortogonalnie lub diagonalnie).
3. **Filtrowanie 4-sąsiedztwa:** Piksele oznaczone są tylko wtedy usuwane, jeśli ich aktywne sąsiedztwo tworzy spójną strukturę w 4-sąsiedztwie.
4. **Obliczenie wagi sąsiedztwa:** Na podstawie macierzy wag (analogia do binarnego kodowania otoczenia pikseli), obliczana jest liczba decydująca o możliwości usunięcia danego piksela.
5. **Iteracyjne usuwanie:** Dwa przebiegi (dla etykiet 2 i 3) pozwalają na bezpieczne usuwanie tylko tych pikseli, które nie zniszczą ciągłości szkieletu.

Oryginalny obraz binarny.

Piksele oznaczone jako 2 (żółte) i 3 (czerwone).

Piksele oznaczone do usunięcia (zielone, wartość 4).

Obraz po usunięciu pikseli o wartości 4.

Po usunięciu zbędnych pikseli o wartości 2.

Obraz po usunięciu zbędnych pikseli o wartości 3 (rezultat).

Rysunek 8: Kolejne etapy szkieletyzacji metodą KMM: oznaczanie, klasyfikacja i usuwanie pikseli.

Rysunek 9: Przykład obrazu po szkieletyzacji metodą KMM.

3.6 Ekstrakcja cech: Minuncje

Po uzyskaniu szkieletu linii papilarnych przeprowadzana jest analiza cech lokalnych zwanych *minutiae*. Dwa typy minuncji, które będą lokalizowane przez algorytm to:

- **Zakończenia grzbietów** – miejsca, gdzie linia się kończy,
- **Rozwidlenia grzbietów (bifurkacje)** – miejsca, gdzie jedna linia dzieli się na co najmniej dwie.

Detekcja minuncji zaimplementowana została w pliku `minutiae.py`, w funkcji `detect_minutiae`. Przetwarza ona obraz szkieletowy o wartościach 0 i 255, który jest binarnie normalizowany do wartości `0` i `1`. Następnie:

1. Dla każdego białego piksela w obrazie analizowane jest jego 8-sąsiedztwo (macierz 3×3),
2. Liczba aktywnych (białych) pikseli wokół danego punktu decyduje o klasyfikacji:
    - Jeśli tylko jeden sąsiad – zakończenie linii,
    - Jeśli trzech lub więcej sąsiadów – rozwidlenie.

Wykryte współrzędne punktów przechowywane są jako dwie osobne listy: `endings` oraz `bifurcations`. Dla potrzeb wizualizacji zastosowano funkcję `visualize_minutiae`, która:

- Konwertuje obraz szkieletowy do przestrzeni BGR,
- Nakłada czerwone punkty w miejscach zakończeń,
- Nakłada zielone punkty w miejscach rozwidleń.

Rysunek 10: Wizualizacja minuncji: zakończenia (żółte), rozwidlenia (różowe)

Cały opisany powyżej algorytm przeprowadzający wszystkie kroki można wywołać przy użyciu funkcji `process_fingerprint`.

4 Przykłady i wnioski

W celu oceny jakości szkieletyzacji i wykrywania cech biometrycznych porównano metodę morfologiczną oraz algorytm KMM na kilku obrazach odcisków tego samego palca, zarejestrowanych w różnych warunkach.

4.1 Porównanie metod szkieletyzacji

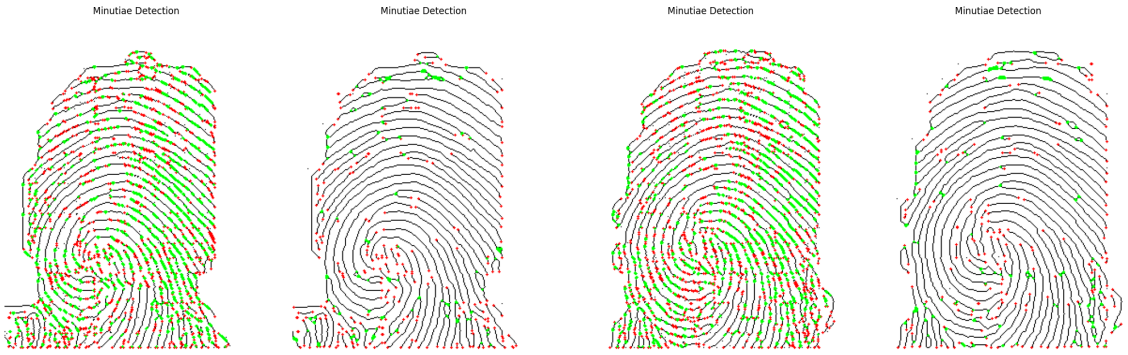
Na rysunku 11 przedstawiono porównanie wyników szkieletyzacji przy użyciu metod morfologicznych i algorytmu KMM. Metoda KMM lepiej odwzorowuje kształt linii papilarnych, zachowując ciągłość grzbietów przy jednoczesnym eliminowaniu szumów. Metoda morfologiczna wykonuje się znacznie szybciej, ale wynikowy obraz ma w sobie więcej szumów i nie wszędzie zachowana jest grubość jednego piksela.

(a) Obraz oryginalny po filtrowaniu. (b) Szkielet – morf. (c) Szkielet – KMM.

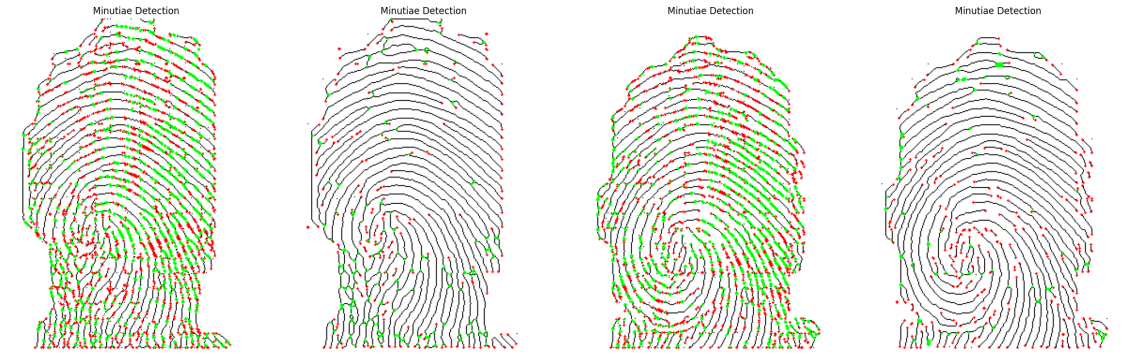
Rysunek 11: Porównanie szkieletów dla przykładowego obrazu.

4.2 Porównanie minucji

Na rysunku 12 pokazano wykryte minucje: zakończenia (czerwone) oraz rozwidlenia (zielone) dla szkieletyzacji wykonanej obiema metodami.



(a) Morf. – obraz 1 (b) KMM – obraz 1 (c) Morf. – obraz 2 (d) KMM – obraz 2



(e) Morf. – obraz 3 (f) KMM – obraz 3 (g) Morf. – obraz 4 (h) KMM – obraz 4

Rysunek 12: Wykryte minucje na różnych obrazach tego samego palca: zakończenia (czerwone) i rozwidlenia (zielone).

4.3 Liczba wykrytych minucji

W tabeli 1 przedstawiono liczby wykrytych zakończeń i rozwidleń dla czterech obrazów odcisków tego samego palca.

Obraz	Metoda	Zakończenia (czerwone)	Rozwidlenia (zielone)	Suma
1	Morfologiczna	1013	2149	3162
1	KMM	259	242	501
2	Morfologiczna	1125	2239	3364
2	KMM	200	277	477
3	Morfologiczna	1214	2146	3360
3	KMM	197	266	463
4	Morfologiczna	1122	2066	3188
4	KMM	268	202	470

Tabela 1: Porównanie liczby wykrytych minucji.

Różnice w liczbie wykrytych minucji mogą wynikać ze zmian w mocy docisku oraz położeniu palca na skanerze, co wpływa na to, która dokładnie część powierzchni opuszka została zarejestrowana. Zbyt silny docisk może prowadzić do deformacji obrazu i wycięcia fragmentów odcisku, natomiast zbyt słaby — do niepełnego odwzorowania linii papilarnych. Dodatkowo, podczas segmentacji czasem zdarza się, że do obszaru analizy zostaje błędnie włączone tło, co również może wpływać na jakość wyników. W praktycznych systemach identyfikacyjnych, by uzyskać wysoką powtarzalność, należałoby ograniczyć analizę do mniejszego, reprezentatywnego obszaru (np. centralnej części opuszka), zdefiniowanego w sposób jednoznaczny. Algorytm KMM zapewnia dużo bardziej stabilne wyniki, dlatego to jego rezultaty powinny być preferowane przy dalszym przetwarzaniu i dopasowywaniu. Morfologiczna szkieletyzacja generuje stosunkowo poszarpany, nieregularny szkielet (często też miejscami grubszy niż jeden piksel) co znacznie pogarsza wykrywanie minucji, co widać na uzyskanych wynikach.

5 Detekcja minucji z kierunkami

W tej sekcji dodatkowo, dla każdej wykrytej minucji oszacowano jej kierunek.

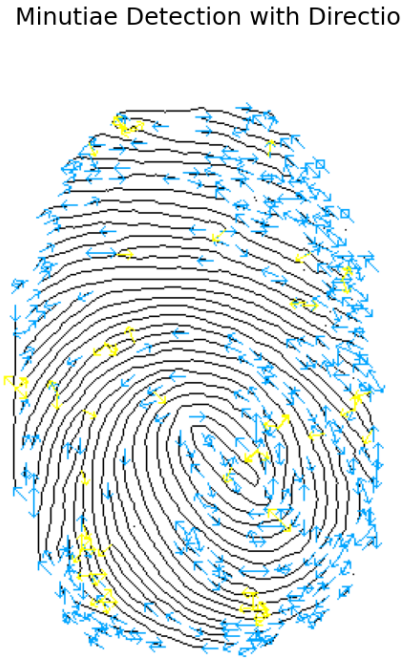
Zasada działania

- Kierunek minucji estymowany jest na podstawie otoczenia 5×5 poprzez analizę rozkładu sąsiednich pikseli i obliczenie średniego wektora kierunku.
- Dla bifurkacji dodatkowo wyznaczany jest kierunek między dwoma najbardziej zbliżonymi rozwidleniami, co lepiej oddaje geometrię rozdzielenia linii.

Wizualizacja

Na obrazie wynikowym minucje wizualizowane są za pomocą kolorowych strzałek:

- zakończenia linii — seledynowe strzałki,
- bifurkacje — żółte strzałki, wskazujące kierunek między rozwidleniami.



Rysunek 13: Wizualizacja wykrytych minucji i ich kierunków

W większości przypadków estymacja kierunku dla zakończeń linii jest trafna i dobrze odzwierciedla rzeczywisty przebieg grzbietów. Dla bifurkacji jakość kierunku jest nieco gorsza, co może wynikać z faktu, że do analizy używane jest stosunkowo małe sąsiedztwo.

Dalsze ulepszenia mogą polegać na zastosowaniu większych okien analizy lub bardziej zaawansowanych metod wektorowych uwzględniających lokalny kontekst struktury grzbietów. Niezależnie od tego, uzyskana wizualizacja stanowi przydatne narzędzie pomocnicze przy analizie minucji w systemach biometrycznych.

5.1 Wnioski i podsumowanie

Algorytm KMM umożliwia uzyskanie szkieletów o znacznie lepszej ciągłości i mniejszym szumie, co przekłada się na bardziej wiarygodne i dokładne wykrywanie minucji. W przeciwieństwie do tego, szkielety generowane metodą morfologiczną często charakteryzują się przerywanymi liniami

i obecnością zbędnych rozwidleń, a także niekoniecznie szkieletem o szerokości jednego piksela, co prowadzi do wykrycia wielu fałszywych punktów charakterystycznych. Minucje uzyskane za pomocą KMM wykazują znacznie większą stabilność między różnymi obrazami tego samego palca, co jest kluczowe z punktu widzenia skuteczności procesu dopasowywania. Pomimo licznych prób poprawy jakości szkieletu generowanego metodą morfologiczną — w tym testów różnych elementów strukturalnych, dodatkowych operacji morfologicznych czy filtrów wygładzających — nie udało się uzyskać zadowalającego rezultatu. Możliwe, że dalsze eksperymenty lub połączenie morfologicznej szkieletyzacji z innymi technikami przetwarzania mogłyby przynieść poprawę, jednak w obecnej implementacji metoda ta okazuje się zbyt podatna na zakłócenia i niestabilność.

6 Źródła

- dr inż. J. Rafałko, *Wykłady z Biometrii*, Politechnika Warszawska, 2025.
- Khalid Saeed, Mariusz Rybnik, Marek Tabędzki i Marcin Adamski, Algorytm do Ścieniania Obrazów: Implementacja i Zastosowania, Zeszyty Naukowe Politechniki Białostockiej 2002 `http://jrafalko.cba.pl/mini/KMM%20pl.pdf`
- OpenAI, ChatGPT (model GPT-4), dostęp online: `https://openai.com/chatgpt`, dostęp: maj 2025.
- Cuevas, *Fingerprint Recognition Repository*, GitHub, `https://github.com/cuevas1208/fingerprint_recognition`, dostęp: maj 2025.