Wykorzystanie algorytmów ścieniania do automatycznej detekcji minucji na obrazach odcisków palców

Mateusz Iwaniuk, Hubert Kowalski

28 maja 2025

Streszczenie

Sprawozdanie przedstawia efekty implementacji algorytmów przetwarzania obrazów automatycznego rozpoznawania minucji na zdjęciach odcisków palców. Projekt koncentruje się na zastosowania algorytmu szkieletyzacji morfologicznej i algorytmu KMM do wstępnego przetwarzania obrazów. Następnie wykonywana jest detekcja minucji takich jak zakończenia czy bifurkacje. Sprawozdanie szczegółowo opisuje zastosowane metody przetwarzania obrazu, wykorzystane algorytmy oraz napotkane wyzwania i rozwiązania.

Spis treści

1 Wprowadzenie

1.1 Cel projektu

Celem projektu było porównanie efektywności algorytmów ścieniania w kontekście przetwarzania obrazów odcisków palców. Główne założenia obejmowały:

- Implementację oraz analizę dwóch metod ścieniania:
  - Szkieletyzacji morfologicznej (w oparciu o techniki prezentowane podczas wykładów)
  - Algorytmu KMM
- Wykorzystanie operacji morfologicznych i innych technik przetwarzania obrazu do usuwania nieciągłości w ścienionych liniach papilarnych
- Automatyczne wyznaczanie lokalizacji minucji, w tym:
  - Rdzenia
  - Zakończeń
  - Bifurkacji
  - Delty

Podstawę projektu stanowił modułowy pipeline przetwarzania obrazu zrealizowany w języku Python z wykorzystaniem bibliotek *cv2*, *numpy* oraz *matplotlib*. Struktura kodu w folderze *src* odzwierciedla podział na etapy przetwarzania: wstępną obróbkę obrazu, ścienianie, rekonstrukcję linii oraz detekcję minucji.
Podczas prac bazowano na metodologiach opisanych w artykule Saeed et al. [2002], który dostarczył teoretycznych podstaw dla implementowanych algorytmów ścieniania. Projekt stanowi zarówno weryfikację wniosków z literatury przedmiotu, jak i próbę ich praktycznego rozszerzenia i porównania w kontekście biometrycznej analizy odcisków palców.

1.2 Znaczenie biometrii odcisków palca

Biometria odcisku palca, oparta na liniach papilarnych, jest kluczową technologią identyfikacyjną XXI wieku. Jej przewaga nad tradycyjnymi metodami

wynika z niezmienności wzorców przez całe życie oraz trudności w podrobieniu Maltoni et al. [2009]. Podstawę rozpoznawania stanowią **minucje**– mikroskopijne cechy geometryczne, takie jak zakończenia, bifurkacje czy skrzyżowania linii, których układ jest unikalny nawet dla bliźniąt jednojajowych.

Proces identyfikacji obejmuje trzy etapy:

- Akwizycję obrazu (czytniki optyczne/pojemnościowe),
- Ekstrakcję minucji z wykorzystaniem algorytmów AI i przetwarzania obrazu,
- Porównanie z bazą wzorców poprzez analizę statystyczną Chen and Deng [2019].

Mimo wysokiej dokładności (FAR $< 0.001\%$ w systemach komercyjnych Group [2021]), wyzwania obejmują:

- Wrażliwość na jakość sensorów,
- Zmiany fizjologiczne skóry,
- Ryzyko naruszenia prywatności danych biometrycznych

Zastosowania sięgają od kontroli dostępu do śledztw kryminalistycznych, a rozwój technologii opartych o głębokie sieci neuronowe stale poszerza ich możliwości Nguyen et al. [2021].

1.3 Ogólny zarys systemu

System rozpoznawania odcisków palców wykorzystuje złożony algorytm przetwarzania obrazu, składający się z kilku kluczowych etapów przetwarzania. Proces ten przeprowadza transformację od oryginalnego, nieprzetworzonego obrazu odcisku palca do finalnego wyniku zawierającego zlokalizowane minucje.
Algorytm, w najbardziej rozbudowanej wersji, realizuje następujące kroki:

1. **Wstępne przetwarzanie obrazu:**
    - Usuwanie szumu przy zastosowaniu filtra medianowego i rozmycia gaussowskiego
    - Binaryzacja obrazu
    - Operacja dylatacji w celu połączenia przerwanych linii papilarnych

2. **Ścienianie obrazu:**

    - Zastosowanie algorytmu ścieniania KMM lub alternatywnie szkieletyzacji morfologicznej
    - Redukcja linii papilarnych do szerokości jednego piksela przy zachowaniu ich ciągłości i topologii

3. **Przetwarzanie końcowe:**

    - Operacja domknięcia morfologicznego w celu połączenia bliskich zakończeń linii
    - Mostkowanie przerw (bridging) do łączenia przerwanych segmentów

4. **Detekcja minucji:**

    - Analiza lokalnego sąsiedztwa dla każdego piksela
    - Identyfikacja dwóch spośród czterech typów minucji: zakończenia i bifurkacji (rozwidlenia)
    - Eliminacja fałszywych minucji przy wykorzystaniu reguł geometrycznych

Zastosowane podejście bazuje na klasycznych metodach przetwarzania obrazów odcisków palców opisanych w literaturze. W szczególności detekcja minucji opiera się na metodologii zaprezentowanej przez Maiego i Jaina Mai and Jain [2009]. Nasze rozwiązanie wprowadza dodatkowe techniki poprawy jakości przez łączenie przerwanych linii papilarnych, co zwiększa efektywność wykrywania minucji w obrazach o niższej jakości.

2 Opis aplikacji

2.1 Środowisko implementacji

Projekt został zrealizowany w języku *Python* w wersji **3.10.13**. Ostateczną prezentację wyników i możliwości systemu zaprezentowano z wykorzystaniem interaktywnego środowiska Jupyter Notebook (plik `app.ipynb`). Kluczowe elementy systemu zostały zaimplementowane w formie niezależnych modułów w folderze `src`, w tym skrypty implementujące ścienianie i automatyczną detekcję minucji. Szczegóły dotyczące środowiska znajdują się na repozytorium *github* projektu: Biometric-Fingerprint-Recognition.

2.2 Architektura systemu

Architektura systemu została zaprojektowana modułowo, co umożliwia łatwą wymianę poszczególnych komponentów i porównywanie różnych implementacji algorytmów ścieniania. Poniżej przedstawiono strukturę systemu w formie tekstowej z krótkim opisem najważniejszych komponentów.

```
Biometric-Fingerprint-Recognition/

 src/                          # Kod źródłowy aplikacji
    stitching/                 # Moduły do ścieniania obrazu
       kmm.py                  # Implementacja algorytmu ścieniania KMM
       morphological_skeletonization.py  # Ścienianie morfologiczne

    improvement/               # Moduły do ulepszania obrazu
       connect_broken_fingerprint_ridges.py  # Łączenie przerwanych linii

    detect_minutiae/           # Moduły do wykrywania minucji
        detect_minutiae.py     # Detekcja minucji

 data/                        # Obrazy testowe odcisków palców

 app.ipynb                    # Główny interfejs aplikacji w formie notebooka

 test_kmm.py                  # Skrypt testowy dla algorytmu KMM
```

Rysunek 1: Struktura głównych komponentów systemu do rozpoznawania minucji w odciskach palców

2.3 Wykorzystane biblioteki

Biblioteka	Wykorzystanie
NumPy	Operacje na macierzach i wydajne przetwarzanie danych obrazowych. Wykorzystana do implementacji algorytmów ścieniania, szczególnie KMM, gdzie wymagane są złożone operacje na tablicach dwuwymiarowych reprezentujących obraz.
PIL (Python Imaging Library)	Wczytywanie, zapisywanie oraz podstawowe operacje na obrazach. Używana głównie do otwierania plików graficznych z odciskami palców i zapisywania wyników przetwarzania.
OpenCV (cv2)	Zaawansowane przetwarzanie obrazów, w tym operacje morfologiczne (dylatacja, erozja, domknięcie), filtrowanie obrazów (rozmycie gaussowskie, filtr medianowy) oraz detekcja krawędzi. Biblioteka odgrywa kluczową rolę w poprawianiu jakości obrazów przed i po procesie ścieniania.

Tabela 1: Biblioteki wykorzystane w projekcie wraz z ich zastosowaniem

3 Ścienianie metodą szkieletyzacji morfologicznej

Teoretyczne podstawy szkieletyzacji morfologicznej

Szkieletyzacja morfologiczna to technika przetwarzania obrazu, która umożliwia redukcję kształtu obiektu (w naszym przypadku linii papilarnych) do linii o grubości jednego piksela. Jest to kluczowy etap w procesie rozpoznawania odcisków palców, ponieważ pozwala na zachowanie topologicznych właściwości wzoru przy jednoczesnej eliminacji nadmiarowych informacji o szerokości linii.
Matematyczna definicja szkieletu morfologicznego obiektu X opiera się na następującej formule:

$$S(X) = \bigcup_{k=0}^{K} S_k(X)$$

gdzie komponent $S_k(X)$ dla każdej iteracji k jest definiowany jako:

$$S_k(X) = (X \ominus kB) \setminus ((X \ominus kB) \circ B)$$

W powyższych wzorach:

- X reprezentuje obraz binarny odcisku palca
- B oznacza element strukturalny (najczęściej krzyż 3×3)
- $\ominus$ to operacja erozji morfologicznej
- $\circ$ to operacja otwarcia morfologicznego (erozja, a następnie dylatacja)
- $\setminus$ oznacza różnicę zbiorów
- K jest maksymalną liczbą iteracji, przy której $(X \ominus kB) \neq \emptyset$

Implementacja algorytmu

Algorytm szkieletyzacji morfologicznej zastosowany w naszym projekcie składa się z następujących kroków:

Preprocessing obrazu wejściowego:

- Konwersja obrazów kolorowych do skali szarości
- Normalizacja wartości pikseli do standardowego zakresu $[0, 255]$
- Binaryzacja z zastosowaniem odpowiedniego progu (domyślnie 127)

3.0.1 Definiowanie elementu strukturalnego:

- W naszej implementacji użyto elementu strukturalnego typu krzyż 3×3 (MORPH_CROSS)
- Alternatywnie dostępny jest również element kwadratowy 3×3 (MORPH_RECT)

3.0.2 Iteracyjny proces szkieletyzacji:

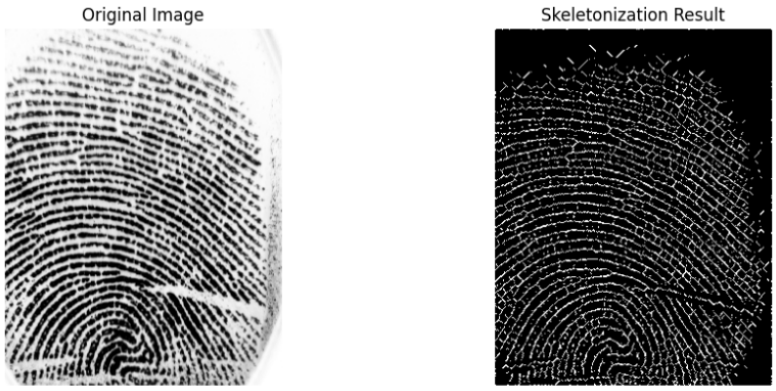
- Inicjalizacja pustego obrazu szkieletu
- W każdej iteracji k wykonywane są operacje:
  - Obliczenie erozji ($X \ominus kB$)
  - Zastosowanie operacji otwarcia na wynikowej erozji
  - Obliczenie różnicy między erozją a wynikiem otwarcia
  - Dodanie obliczonej różnicy do szkieletu

3.0.3 Warunki zakończenia:

- Algorytm kończy działanie, gdy obraz po erozji jest pusty (nie zawiera pikseli obiektu)
- Dodatkowe zabezpieczenie stanowi ograniczenie maksymalnej liczby iteracji
- Implementacja zawiera także optymalizację kończącą proces, gdy kolejne erozje nie zmieniają obrazu

3.1 Komentarz twórców

Działanie algorytmu można opisać cytatem z popularnego tekstu kultury "Nie jest dobrze, ale nie jest też źle. Można by rzec, że jest średnio".
Przy optymalnie dobranych parametrach i odpowiednim preprocessingu szkieletyzacja morfologiczna stanowi jednak efektywne narzędzie do przygotowania obrazu odcisku palca do późniejszej analizy minucji.



Rysunek 2: Efekt algorytmu szkieletyzacji morfologicznej.

4 Ścienianie metodą KMM

4.1 Teoretyczne podstawy ścieniania metodą KMM

Algorytm KMM (Khalid-Marcin-Mariusz) opiera się na wielofazowej iteracyjnej procedurze tworzenia szkieletu obrazu binarnego o szerokości jednego piksela. Jego implementacja została oparta o prace naukową autorów tego rozwiązania.
Kluczowe definicje i operacje matematyczne:

1. **Obraz binarny**: Macierz $B \in \{0,1\}^{M \times N}$, gdzie 1 oznacza pikselowy (obiekt), a 0 tło.

2. **Piksele konturowe**: Piksel (i,j) jest krawędziowy (etykieta 2), jeśli sąsiaduje z tłem ($B_{i,j} = 1 \wedge \exists_{(k,l)\in N_8(i,j)} B_{k,l} = 0$). Piksel jest narożny (etykieta 3), jeśli przylega do tła po przekątnej.

3. **Macierz wag sąsiadów**:

$$W = \begin{bmatrix} 128 & 1 & 2 \\ 64 & X & 4 \\ 32 & 16 & 8 \end{bmatrix}$$

   gdzie X to środek (wartość 0 dla obliczeń).

4. **Suma wag sąsiadów**: Dla sąsiedztwa $N_8(i,j)$, suma wag $S = \sum_{(k,l)\in N_8(i,j)} W_{k,l} \cdot B_{k,l}$.

5. **Tablica usunięć**: Zbiór 128 wartości (np. $\{3, 5, 7, 12, \ldots, 255\}$), dla których piksel zostaje usunięty.

4.2 Implementacja algorytmu KMM

Implementacja algorytmu została zawarta w module `src/stitching/kmm.py`. Obejmuje ona trzy podstawowe etapy:

4.2.1 Preprocessing

- **Usuwianie szumu**: Zastosowanie filtrów medianowego i Gaussowskiego.
- **Dylatacja**: Połączenie zerwanych fragmentów za pomocą jądra kwadratowego (domyślnie 3×3).

4.2.2 Główny algorytm

1. **Inicjalizacja binarna**: Konwersja obrazu do macierzy B.
2. **Etykietowanie konturu**:

   - Znajdź piksele krawędziowe (2) i narożne (3).
3. **Analiza sąsiedztwa**:

   - Dla każdego piksela policz liczbę jego sąsiadów. Jeśli liczba ta mieści się w przedziale $(2 - 4)$, oznacz ten piksel jako 4.
4. **Usuwanie pikseli**:

   - Oblicz sumę wag S dla sąsiadów danego piksela, na podstawie macierzy wag sąsiadów przedstawionej wyżej.

   - Usuń piksel, jeśli $S \in$ Tablica usunięć.
5. **Przywracanie ciągłości**:

   - Zachowaj te spośród pikseli 2/3, które zapobiegają przerwom w szkielecie.

4.2.3 Postprocessing

- **Zamykanie morfologiczne**: Łączenie końców fragmentów.
- **Mostkowanie przerw**: Dylatacja + erozja z jądrem 3×3.

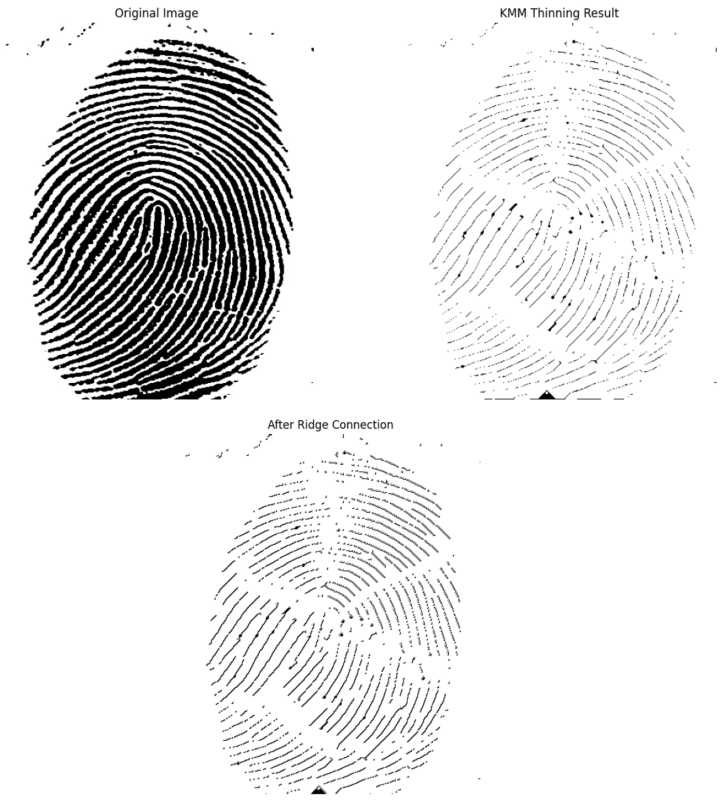
4.3 Efekty działania

Algorytm KMM wykazuje ograniczoną skuteczność w porównaniu do tradycyjnego ścieniania morfologicznego. Na rysunku 3 przedstawiono przykładowy wynik, gdzie widoczne są:

- Przerwane linie w obszarach o złożonej topologii.
- Niepełne usunięcie nadmiarowych pikseli.

Zastosowanie preprocessingu i postprocessingu poprawia efekty, ale nie eliminuje całkowicie defektów. Potencjalne ulepszenia obejmują:

- Modyfikację tablicy wag W.
- Adaptacyjne dostosowanie tablicy usunięć.
- Hybrydyzację z metodami gradientowymi.
- Implementację bardziej zaawansowanego algorytmu K3M.



Rysunek 3: Przykład działania algorytmu KMM. (a) Obraz oryginalny, (b) Wynik ścieniania, (c) Wynik po dodatkowych poprawkach poprzez funkcję `connect_broken_fingerprint_ridges`

5 Algorytm poprawy obrazu odcisków palców po ścienianiu

Po procesie ścieniania linii papilarnych, obraz może nadal zawierać pewne nieciągłości lub artefakty. W celu dalszej poprawy jakości obrazu binarnego, stosowany jest algorytm modyfikujący piksele na podstawie ich wartości oraz wartości ich sąsiadów. Algorytm operuje na obrazie, gdzie linie papilarne są reprezentowane przez białe piksele (wartość 255), a tło przez czarne piksele (wartość 0).
Podstawą działania jest wykorzystanie dwóch obrazów pomocniczych, wygenerowanych przy użyciu standardowych operacji morfologicznych: erozji i dylatacji. Obie te operacje wykorzystują wspólny element strukturalny (jądro) o wymiarach 2×2, składający się z samych jedynek:

$$K = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{1}$$

Pierwszy obraz pomocniczy, I_{eroded}, jest wynikiem zastosowania operacji erozji na obrazie wejściowym $I_{thinned}$:

$$I_{eroded} = I_{thinned} \ominus K \tag{2}$$

Drugi obraz pomocniczy, $I_{dilated}$, jest wynikiem zastosowania operacji dylatacji na tym samym obrazie wejściowym $I_{thinned}$:

$$I_{dilated} = I_{thinned} \oplus K \tag{3}$$

Obraz wynikowy, I_{output}, jest następnie konstruowany poprzez warunkowe przypisanie wartości pikseli z obrazów I_{eroded} lub $I_{dilated}$, w zależności od wartości odpowiadającego im piksela w oryginalnym obrazie $I_{thinned}$. Logika ta jest następująca:

- Jeśli piksel (x, y) w obrazie $I_{thinned}$ jest biały (ma wartość 255), to piksel (x, y) w obrazie wynikowym I_{output} przyjmuje wartość piksela (x, y) z obrazu I_{eroded}. W praktyce oznacza to, że biały piksel, który posiada przynajmniej jednego czarnego sąsiada (w ramach jądra 2×2), zostanie zamieniony na czarny.

- Jeśli piksel (x, y) w obrazie $I_{thinned}$ jest czarny (ma wartość 0), to piksel (x, y) w obrazie wynikowym I_{output} przyjmuje wartość piksela (x, y) z obrazu $I_{dilated}$. Oznacza to, że czarny piksel, który posiada przynajmniej jednego białego sąsiada (w ramach jądra 2×2), zostanie zamieniony na biały.

Formalnie, operację tę można zapisać jako:

$$I_{output}(x,y) = \begin{cases} I_{eroded}(x,y) & \text{jeśli } I_{thinned}(x,y) = 255 \\ I_{dilated}(x,y) & \text{jeśli } I_{thinned}(x,y) = 0 \end{cases} \tag{4}$$

W implementacji z wykorzystaniem biblioteki NumPy, odpowiada to operacji `np.where(thinned_image == 255, eroded_image, dilated_image)`. Efektem tego algorytmu jest modyfikacja obrazu, która może prowadzić do usunięcia niektórych izolowanych białych pikseli (szumu) oraz wypełnienia małych przerw w liniach papilarnych, jednocześnie potencjalnie wpływając na grubośćsamych linii.

6 Opis wyszukiwania automatycznego zakończeń i bifurkacji i zdjęcia efektów

Identyfikacja unikalnych cech w obrazie odcisku palca, znanych jako minucje, jest kluczowym etapem w systemach rozpoznawania biometrycznego. Najczęściej analizowanymi typami minucji są zakończenia (punkty, w których linia papilarna nagle się kończy) oraz bifurkacje (punkty, w których linia papilarna rozdziela się na dwie). W naszym systemie proces ten realizowany jest dwuetapowo: najpierw detekcja minucji, a następnie ich wizualizacja na obrazie.

6.1 Detekcja minucji metodą Liczby Skrzyżowań (Crossing Number)

Do wykrywania zakończeń i bifurkacji wykorzystywana jest metoda oparta na koncepcji Liczby Skrzyżowań (Crossing Number - CN). Metoda ta analizuje lokalne sąsiedztwo każdego piksela należącego do linii papilarnej na obrazie po procesie ścieniania. Obraz wejściowy dla tej operacji to obraz binarny, gdzie linie papilarne są reprezentowane przez białe piksele (wartość 255), a tło przez czarne piksele (wartość 0). Na potrzeby obliczeń CN, obraz jest przekształcany tak, aby linie miały wartość 1, a tło 0.

6.1.1 Obliczanie Liczby Skrzyżowań

Liczba Skrzyżowań dla centralnego piksela P w oknie 3×3 jest obliczana poprzez analizę wartości jego ośmiu sąsiadów $(P_1, P_2, \ldots, P_8)$ ułożonych w kolejności zgodnej z ruchem wskazówek zegara lub przeciwnie do niego:

$$\text{CN} = \frac{1}{2} \sum_{i=1}^{8} |P_i - P_{(i \pmod 8)+1}| \tag{5}$$

gdzie P_k to wartość binarna k-tego sąsiada (0 dla tła, 1 dla linii). Wartość CN określa typ minucji:

- CN = 1: wskazuje na zakończenie linii papilarnej.
- CN = 3: wskazuje na bifurkację (rozwidlenie) linii papilarnej.
- Inne wartości CN (np. 0, 2, 4) mogą wskazywać na inne struktury, takie jak piksel izolowany, fragment prostej linii lub bardziej złożone skrzyżowanie, które nie są klasyfikowane jako standardowe zakończenia lub bifurkacje w tym podejściu.

Funkcja pomocnicza `_calculate_crossing_number` implementuje powyższą logikę.

6.1.2 Proces detekcji

Główna funkcja `detect_minutiae` wykonuje następujące kroki:

1. Sprawdza poprawność obrazu wejściowego.
2. Konwertuje obraz wejściowy (linie=255, tło=0) na format wymagany przez algorytm CN (linie=1, tło=0).
3. Iteruje przez wszystkie piksele obrazu, z pominięciem zdefiniowanego marginesu brzegowego (`border_margin`), aby uniknąć błędów analizy na krawędziach.
4. Dla każdego piksela należącego do linii papilarnej (wartość 1):
   - Pobiera okno 3×3 otaczające analizowany piksel.
   - Oblicza wartość CN dla tego okna za pomocą funkcji `_calculate_crossing_number`.
   - Jeśli CN = 1, piksel jest klasyfikowany jako zakończenie.
   - Jeśli CN = 3, piksel jest klasyfikowany jako bifurkacja.
5. Gromadzi współrzędne wykrytych zakończeń i bifurkacji.
6. Opcjonalnie, na podstawie parametru `select_one_of_each_type`, funkcja może zwrócić wszystkie wykryte minucje lub tylko po jednym przykładzie każdego typu (pierwsze znalezione zakończenie i pierwszą znalezioną bifurkację).
7. Zwraca słownik zawierający listy współrzędnych (r, c) dla typów minucji: `"terminations"` i `"bifurcations"`. Należy zaznaczyć, że metoda CN w tej implementacji nie jest przeznaczona do wykrywania rdzeni (cores) ani delt (deltas), więc listy dla tych typów minucji są zawsze puste.

6.2 Detekcja minucjii rdzenia i delty metodą Poincaré

Metoda Poincaré stanowi matematycznie uzasadnioną technikę detekcji punktów osobliwych (singular points) w obrazach linii papilarnych, takich jak rdzeń (core) i delta. W przeciwieństwie do metody liczby skrzyżowań (Crossing Number), która jest skuteczna dla detekcji zakończeń i bifurkacji, metoda Poincaré wykorzystuje analizę pola orientacji do identyfikacji struktur globalnych charakteryzujących wzór linii papilarnych.

6.2.1 Podstawy matematyczne

Podstawą metody Poincaré jest teoria matematyczna indeksu Poincaré, który opisuje zachowanie pola wektorowego wokół punktów osobliwych. W kontekście analizy linii papilarnych, pole orientacji jest traktowane jako pole wektorowe, gdzie każdy wektor reprezentuje lokalną orientację linii papiliarnych. Dla danego punktu $P(i, j)$ w polu orientacji, indeks Poincaré jest definiowany jako:

$$P_{index} = \frac{1}{2\pi} \sum_{k=0}^{7} \delta_k$$

gdzie δ_k reprezentuje różnicę kątową między sąsiednimi orientacjami wokół punktu P:

$$\delta_k = \theta_{k+1} - \theta_k$$

Ze względu na 180-stopniową niejednoznaczność orientacji linii papilarnych (linie nie mają określonego kierunku), różnica kątowa jest normalizowana do przedziału $[-90°, 90°)$:

$$\delta_k = ((\theta_{k+1} - \theta_k) + 90°) \bmod 180° - 90°$$

Wartość indeksu Poincaré pozwala na klasyfikację punktów osobliwych:

- $P_{index} = +1$ (lub $\approx +180°$): punkt typu rdzeń (core)
- $P_{index} = -1$ (lub $\approx -180°$): punkt typu delta
- $P_{index} = 0$: punkt regularny (brak osobliwości)

6.2.2 Algorytm detekcji

Implementacja algorytmu detekcji rdzenia i delty składa się z następujących etapów:

Etap 1: Obliczanie pola orientacji Obraz po skeletonizacji jest dzielony na bloki o rozmiarze $N \times N$ pikseli (domyślnie $N = 8$). Dla każdego bloku zawierającego piksele linii papilarnych obliczana jest lokalna orientacja metodą analizy składowych głównych (PCA):

$$\theta = \frac{1}{2} \arctan\left(\frac{2C_{xy}}{C_{xx} - C_{yy}}\right)$$

gdzie:

$$C_{xx} = \sum_k (x_k - \bar{x})^2 \tag{6}$$

$$C_{yy} = \sum_k (y_k - \bar{y})^2 \tag{7}$$

$$C_{xy} = \sum_k (x_k - \bar{x})(y_k - \bar{y}) \tag{8}$$

są elementami macierzy kowariancji współrzędnych pikseli linii w bloku.

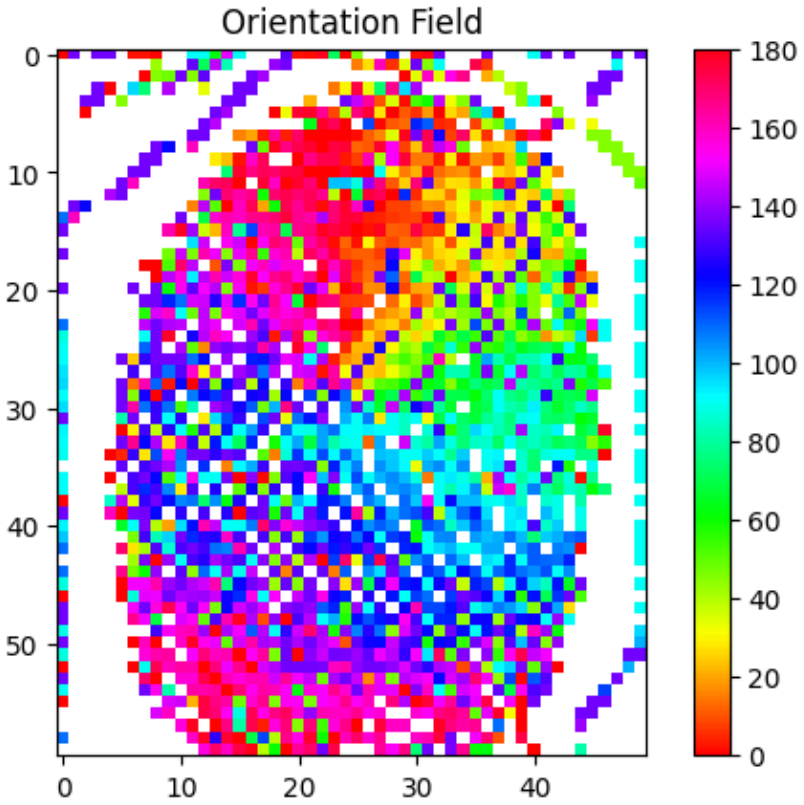
Etap 2: Obliczanie indeksu Poincaré Dla każdego bloku w polu orientacji (z wyłączeniem bloków brzegowych) obliczany jest indeks Poincaré na podstawie 8-elementowego sąsiedztwa. Algorytm implementuje funkcję `compute_poincare_index` która:

1. Pobiera orientacje z 8 sąsiadujących bloków w porządku zgodnym z ruchem wskazówek zegara
2. Oblicza skumulowaną różnicę kątową uwzględniającą 180-stopniową niejednoznaczność
3. Klasyfikuje punkt na podstawie wartości progowych: rdzeń dla sumy w przedziale $(160°, 200°)$, delta dla sumy w przedziale $(-200°, -160°)$

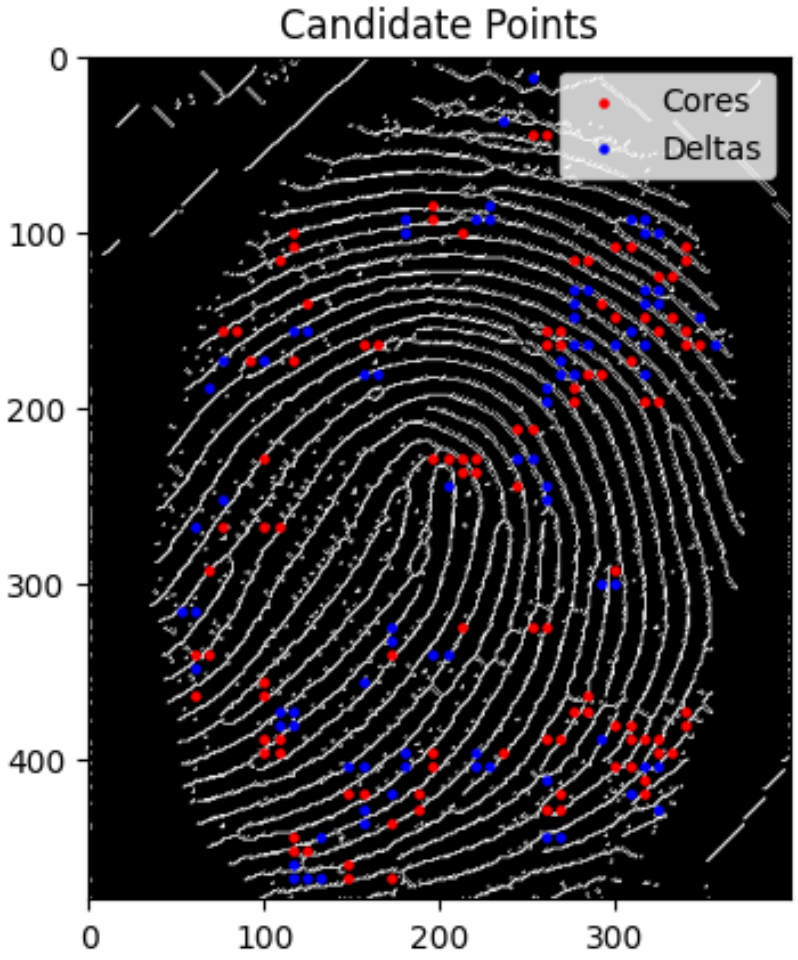
Etap 3: Klasteryzacja kandydatów Zidentyfikowane kandydaty na rdzeń i deltę są grupowane przy użyciu algorytmu DBSCAN (Density-Based Spatial Clustering of Applications with Noise) z parametrem ϵ określającym maksymalną odległość między punktami w klastrze. Z każdego klastra wybierany jest reprezentatywny punkt będący średnią arytmetyczną współrzędnych wszystkich punktów w klastrze.
Funkcja `detect_cores_deltas` implementuje kompletny algorytm i zwraca słownik zawierający współrzędne wykrytego rdzenia i delty. W przypadku gdy w obrazie występuje więcej niż jeden klaster kandydatów danego typu, wybierany jest klaster o największej liczbie punktów.

6.2.3 Wizualizacja wyników

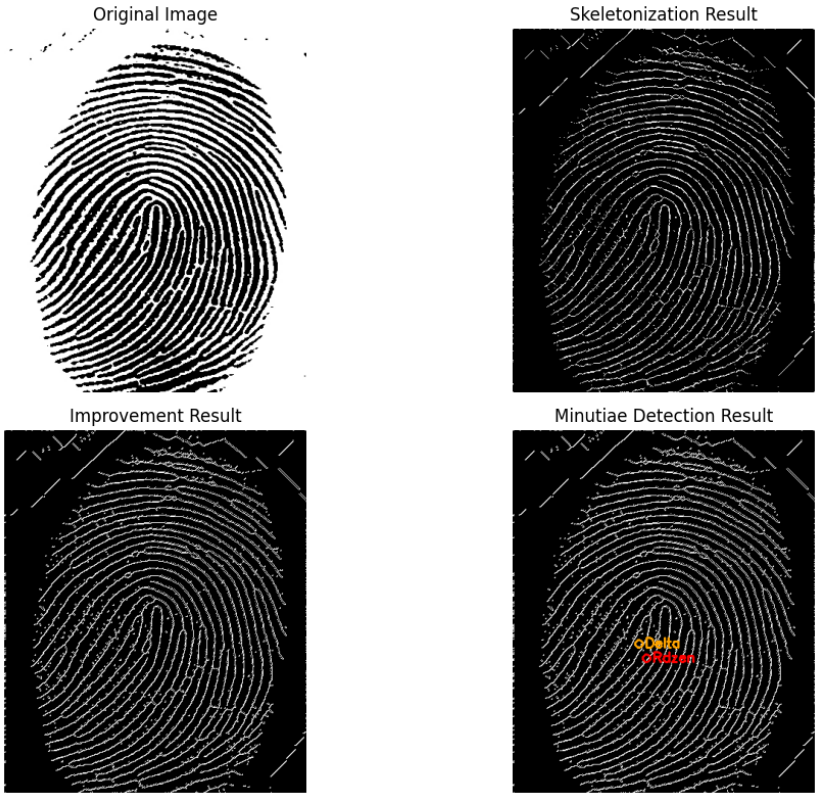
Proces detekcji może być zwizualizowany na trzech poziomach:



Rysunek 4: Mapa orientacji linii papilarnych przedstawiona kolorystycznie. Różne kolory reprezentują różne kąty orientacji lokalnych linii papilarnych obliczone metodą PCA dla bloków 8×8 pikseli.



Rysunek 5: Zaznaczone kandydaci na rdzeń (czerwone punkty) i deltę (niebieskie punkty) przed procesem klasteryzacji. Punkty te powstają w wyniku analizy indeksu Poincaré dla każdego bloku w polu orientacji.



Rysunek 6: Finalne wyniki detekcji przedstawiające znaleziony rdzeń i deltę po procesie klasteryzacji. Punkty te reprezentują najważniejsze struktury globalne wzoru linii papilarnych.

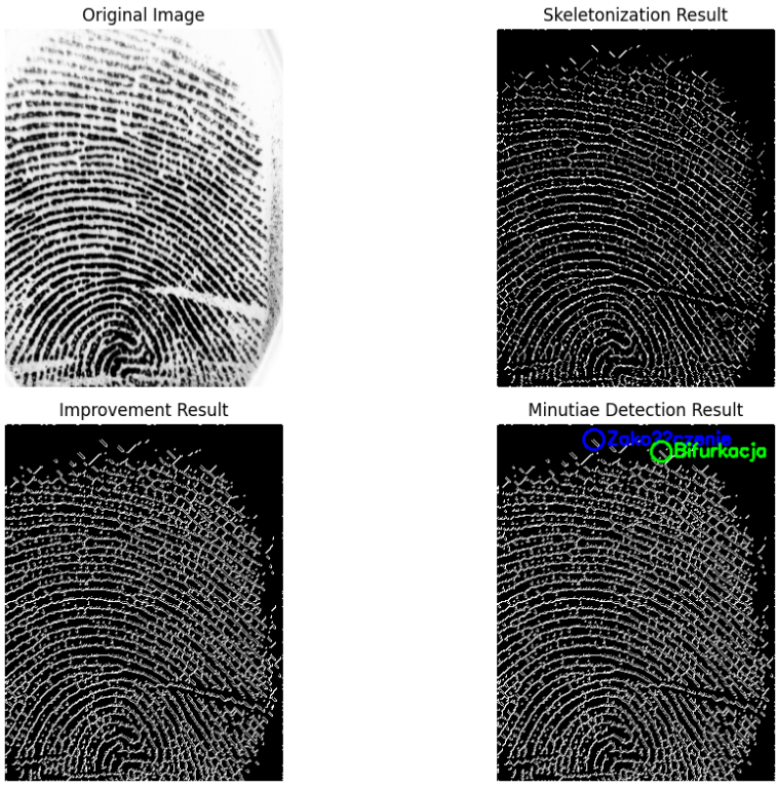
6.3 Wizualizacja wykrytych minucji

Po zakończeniu procesu detekcji, wykryte minucje mogą zostać naniesione na obraz w celu wizualnej weryfikacji. Służy do tego funkcja `draw_minutiae`, która:

1. Przyjmuje jako argumenty obraz, na którym mają być narysowane minucje, oraz słownik z danymi o minucjach (wynik funkcji `detect_minutiae`).
2. Tworzy kopię obrazu wejściowego. Jeśli obraz jest w skali szarości, konwertuje go do formatu BGR, aby umożliwić rysowanie kolorowych oznaczeń.
3. Dla każdej minucji z listy:
   - Rysuje okrąg w miejscu wykrycia minucji. Kolor okręgu zależy od typu minucji (np. czerwony dla zakończeń, zielony dla bifurkacji).
   - Opcjonalnie (jeśli parametr `show_labels` jest `True`), dodaje etykietę tekstową z pełną polską nazwą typu minucji (np. "Zakończenie", "Bifurkacja") obok znacznika.
   - Implementuje logikę pozycjonowania etykiet, aby w miarę możliwości mieściły się one w granicach obrazu i nie zasłaniały nadmiernie innych elementów.
4. Zwraca obraz z naniesionymi oznaczeniami minucji.

6.4 Przykładowe efekty działania

Poniżej przedstawiono przykładowe wyniki działania algorytmów detekcji i wizualizacji minucji.



Rysunek 7: Finalny efekt. Wprawdzie algorytm wyłapał tylko zakończenia obrazu, ale zrobił to prawidłowo.

7 Podsumowanie

7.1 Wnioski

Na podstawie przeprowadzonych eksperymentów i implementacji różnych algorytmów przetwarzania obrazów linii papilarnych można sformułować następujące wnioski:

Szkieletyzacja jako podstawa analizy Szkieletyzacja okazała się kluczowym narzędziem do przetwarzania obrazów linii papilarnych, umożliwiającym redukcję grubości linii do pojedynczych pikseli przy zachowaniu topologii struktury. Dzięki szkieletyzacji możliwa jest dalsza analiza minucji oraz punktów charakterystycznych odcisku palca.

Porównanie metod szkieletyzacji Przeprowadzone testy wykazały, że szkieletyzacja morfologiczna działała lepiej niż algorytm KMM w większo-

ści przypadków, szczególnie dla obrazów o średniej jakości. Algorytm KMM wykazywał dobre rezultaty dla zdjęć wysokiej jakości, jednak w przypadku obrazów zaszumionych lub o niskiej jakości generował więcej artefaktów i przerwanych linii.

Znaczenie przetwarzania końcowego Przetwarzanie końcowe wynikowych obrazów po szkieletyzacji okazało się niezwykle istotne. Funkcje takie jak `connect_broken_fingerprint_ridges` oraz operacje morfologiczne (closing, bridging) znacząco poprawiały jakość szkieletu poprzez łączenie przerwanych linii i eliminację małych przerw. Bez tego etapu detekcja minucji była znacznie mniej skuteczna.

Wyzwania w detekcji minucji Detekcja minucji stanowi kluczowy element systemów uwierzytelniania biometrycznego, jednak okazała się procesem znacznie bardziej skomplikowanym niż pierwotnie zakładano. Główne problemy to wysoka liczba fałszywych detekcji oraz niestabilność wyników w zależności od jakości obrazu wejściowego.

Ograniczenia metody Crossing Number Metoda Crossing Number do detekcji bifurkacji i zakończeń, mimo swojej prostoty implementacyjnej, nie dawała zadowalających efektów. Algorytm generował zbyt dużo fałszywych detekcji (false positives), szczególnie w obszarach zaszumionych lub słabo wyskeletonizowanych. Filtrowanie odległościowe i długości linii papilarnych tylko częściowo rozwiązywało ten problem.

Złożoność metody Poincaré Metoda Poincaré do detekcji delt i rdzeni, choć matematycznie bardziej zaawansowana i teoretycznie uzasadniona, również wykazywała znaczną liczbę fałszywych detekcji. Problemy wynikały głównie z wrażliwości algorytmu na niedokładności w polu orientacji oraz trudności w ustaleniu optymalnych progów klasyfikacji punktów osobliwych.

Niestabilność systemu Ostatecznie uzyskane efekty nie pozwalają na skonstruowanie niezawodnego systemu rozpoznawania odcisków palców ze względu na zbyt wysoką niestabilność wyników. System wymaga znaczących ulepszeń w zakresie filtrowania, normalizacji obrazów oraz algorytmów detekcji, aby mógł być wykorzystany w praktycznych zastosowaniach.

7.2 Ograniczenia

Ograniczenia algorytmiczne

- **Wrażliwość na jakość obrazu**: Wszystkie zaimplementowane algorytmy wykazują silną zależność od jakości obrazu wejściowego, co ogranicza ich zastosowanie w praktycznych scenariuszach
- **Brak adaptacyjności**: Algorytmy nie dostosowują się automatycznie do różnych typów wzorów linii papilarnych (łuk, pętla, whorl)
- **Statyczne parametry**: Większość parametrów (progi, rozmiary kerneli, odległości) jest ustalana statycznie, co nie uwzględnia różnorodności obrazów
- **Pojedyncza rozdzielczość**: Algorytmy nie wykorzystują analizy wieloskalowej, która mogłaby poprawić robustność detekcji

Ograniczenia implementacyjne

- **Sekwencyjne przetwarzanie**: Brak równoległego przetwarzania ogranicza wydajność dla dużych zbiorów danych
- **Ograniczona walidacja**: System nie zawiera mechanizmów walidacji jakości wykrytych minucji
- **Brak normalizacji**: Niekonsekwentne formaty obrazów (0/1 vs 0/255) prowadzą do błędów przetwarzania
- **Nieoptymalne zarządzanie pamięcią**: Tworzenie wielu kopii obrazów zwiększa zużycie pamięci

Ograniczenia metodologiczne

- **Nieuwzględnienie kontekstu**: Algorytmy nie wykorzystują informacji o sąsiadujących minucjach lub globalnej strukturze odcisku
- **Brak uczenia maszynowego**: System bazuje wyłącznie na tradycyjnych metodach przetwarzania obrazu

7.3 Kierunki rozwoju

Ulepszenia algorytmiczne

- **Adaptacyjne przetwarzanie**: Implementacja algorytmów dostosowujących parametry w zależności od lokalnej jakości obrazu i typu wzoru

- **Analiza wieloskalowa**: Wprowadzenie przetwarzania na różnych poziomach rozdzielczości dla lepszej detekcji struktur
- **Zaawansowane filtrowanie**: Implementacja filtrów Gabora i banków filtrów kierunkowych dla lepszej analizy orientacji
- **Hybrydowe podejścia**: Kombinowanie różnych metod detekcji (CN + Poincaré + dodatkowe heurystyki)

Integracja uczenia maszynowego

- **Sieci neuronowe**: Wykorzystanie CNN do detekcji minucji z automatycznym uczeniem cech charakterystycznych
- **Transfer learning**: Adaptacja pretrenowanych modeli do specyfiki detekcji minucji
- **Uczenie nienadzorowane**: Automatyczne wykrywanie wzorców w danych bez ręcznego znakowania

Poprawa jakości obrazu

- **Zaawansowane preprocessing**: Implementacja adaptacyjnej normalizacji histogramu i redukcji szumów
- **Ocena jakości**: Automatyczna ocena jakości obrazu i rejonów nadających się do analizy

Rozszerzenia funkcjonalne

- **Dopasowywanie wzorców**: Implementacja algorytmów porównywania i dopasowywania odcisków palców
- **Multimodalne podejście**: Integracja z innymi metodami biometrycznymi (twarz, tęczówka)

Optymalizacja wydajności

- **Przetwarzanie równoległe**: Implementacja algorytmów GPU dla przyspieszenia obliczeń
- **Optymalizacja algorytmów**: Redukcja złożoności obliczeniowej poprzez aproksymacje i heurystyki

Przedstawione kierunki rozwoju pokazują, że chociaż obecna implementacja ma ograniczenia, stanowi solidną podstawę do dalszych prac nad systemem rozpoznawania odcisków palców. Szczególnie obiecujące wydaje się połączenie tradycyjnych metod przetwarzania obrazu z nowoczesnymi technikami uczenia maszynowego.

Literatura

Jie Chen and Guang Deng. Fingerprint recognition based on deep learning. In *IEEE International Conference on Multimedia & Expo Workshops*, 2019.

International Biometric Group. Biometric performance testing report. Technical report, IBG Consulting, 2021.

Davide Mai and Anil K. Jain. *Handbook of Fingerprint Recognition*. Springer London, London, 2009. ISBN 978-1-84882-253-5.

Davide Maltoni, Dario Maio, Anil K. Jain, and Salil Prabhakar. *Handbook of Fingerprint Recognition*. Springer, 2009.

Thanh Nguyen, Nhat Bui, and Saeid Nahavandi. Deep learning for fingerprint recognition: A survey. *IEEE Transactions on Biometrics*, 3:45–62, 2021.

K. Saeed, M. Rybnik, M. Tabędzki, and M. Adamski. Algorytm ścieniania obrazów: implementacja i zastosowania. *Zeszyty Naukowe Politechniki Białostockiej. Informatyka*, Z.1:209–217, 2002.