Biometria

Projekt nr 3 - Fingerprint Analysis

Mikołaj Rowicki | Filip Langiewicz
nr indeksu 327309 | nr indeksu 327293

Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych
29 maja 2025

Spis treści

1. Wstęp

Biometria odcisków palców stanowi jeden z najstarszych i najbardziej niezawodnych sposobów identyfikacji osób. Unikalne wzory linii papilarnych, które formują się jeszcze przed narodzeniem i pozostają niezmienne przez całe życie, stanowią podstawę wielu systemów bezpieczeństwa i kontroli dostępu. W ramach niniejszego projektu zaimplementowano kompletny system analizy odcisków palców, obejmujący przetwarzanie wstępne, ekstrakcję cech charakterystycznych oraz ich wizualizację.

Głównym celem projektu było stworzenie efektywnego pipeline'u przetwarzania obrazów odcisków palców, który realizuje następujące zadania:

1. Wstępne przetwarzanie obrazu w celu poprawy jakości i przygotowania do dalszej analizy
2. Szkieletyzacja (thinning) obrazu odcisku palca przy użyciu dwóch różnych algorytmów
3. Detekcja i klasyfikacja minucji - charakterystycznych punktów odcisku palca
4. Wizualizacja wyników poszczególnych etapów przetwarzania

W projekcie zaimplementowano dwa algorytmy szkieletyzacji: algorytm KMM oraz szkieletyzację morfologiczną. Oba podejścia zostały szczegółowo opisane i porównane pod względem skuteczności oraz jakości uzyskiwanych wyników.

System został zaprojektowany z myślą o modułowości i łatwości rozbudowy. Każdy z głównych komponentów (przetwarzanie wstępne, szkieletyzacja, detekcja minucji) został zaimplementowany jako osobny moduł, co umożliwia łatwe testowanie różnych wersji algorytmów.

2. Opis techniczny kodu

2.1. Technologie i biblioteki

Projekt został zaimplementowany w języku Python 3, wykorzystując szereg bibliotek specjalizowanych w przetwarzaniu obrazów i analizie danych.

2.1.1. Wykorzystane biblioteki

- **NumPy** - podstawowa biblioteka do obliczeń numerycznych, wykorzystywana do reprezentacji obrazów jako macierzy oraz operacji na nich
- **OpenCV (cv2)** - biblioteka do przetwarzania obrazów, używana głównie do operacji morfologicznych, progowania oraz podstawowych transformacji
- **PIL (Python Imaging Library)** - biblioteka do wczytywania i zapisywania obrazów w różnych formatach
- **Matplotlib** - biblioteka do wizualizacji wyników, szczególnie przydatna przy wyświetlaniu obrazów z naniesionymi minucjami
- **IPython.display** - moduł umożliwiający eleganckie wyświetlanie obrazów w środowisku Jupyter Notebook

2.1.2. Struktura plików projektu

Projekt składa się z następujących modułów:

- `KMM.py` - implementacja algorytmu KMM do szkieletyzacji
- `morphology.py` - implementacja szkieletyzacji morfologicznej
- `detect_minutiae.py` - moduł detekcji i klasyfikacji minucji
- `main.ipynb` - główny notebook integrujący wszystkie komponenty

2.2. Struktura kodu i użytkowanie pipeline'u

Pipeline przetwarzania odcisków palców został zaprojektowany jako sekwencja modułowych operacji, które można konfigurować za pomocą parametrów. Główna funkcja `perform_thinning` stanowi punkt wejścia do systemu i nadzoruje cały proces przetwarzania.

2.2.1. Funkcja główna - perform_thinning

Funkcja przyjmuje następujące parametry:

- `path` - ścieżka do pliku z obrazem odcisku palca
- `show_by_iterations` - flaga kontrolująca wyświetlanie wyników pośrednich
- `algorithm` - wybór algorytmu szkieletyzacji ('KMM' lub 'morphology')
- `show` - flaga kontrolująca wyświetlanie wyników końcowych
- `correct_original` - czy zastosować korekcję morfologiczną przed szkieletyzacją
- `correct_thinning` - czy zastosować korekcję morfologiczną po szkieletyzacji

2.2.2. Przepływ przetwarzania

1. **Wczytanie obrazu** - funkcja `read_bmp` wczytuje obraz i przeprowadza wstępną binaryzację
2. **Korekcja wstępna** - opcjonalna funkcja `correct_orig` poprawia jakość obrazu binarnego
3. **Szkieletyzacja** - zastosowanie wybranego algorytmu (KMM lub morfologiczny)
4. **Korekcja końcowa** - opcjonalna funkcja `correct` usuwa artefakty szkieletyzacji
5. **Detekcja minucji** - identyfikacja charakterystycznych punktów odcisku

3. Algorytm KMM

Algorytm KMM jest jedną z klasycznych metod szkieletyzacji obrazów binarnych. Został opracowany specjalnie do przetwarzania odcisków palców i innych struktur wymagających zachowania topologii. Algorytm działa iteracyjnie, usuwając piksele z brzegu obiektu w sposób kontrolowany, aż do uzyskania szkieletu o grubości jednego piksela.

3.1. Podstawy teoretyczne

Algorytm KMM opiera się na analizie lokalnego sąsiedztwa każdego piksela. Dla każdego piksela należącego do obiektu (wartość 1) analizowane jest jego 8-sąsiedztwo. Na podstawie konfiguracji sąsiadów podejmowana jest decyzja o usunięciu lub zachowaniu piksela.

Definicja 3.1 (Waga piksela)**.** Waga piksela w algorytmie KMM jest obliczana jako suma wartości sąsiadów pomnożonych przez odpowiednie wagi z maski:

$$\begin{bmatrix} 128 & 1 & 2 \\ 64 & 0 & 4 \\ 32 & 16 & 8 \end{bmatrix}$$

gdzie środkowy element (0) reprezentuje analizowany piksel.

3.2. Implementacja algorytmu

Algorytm KMM w naszej implementacji składa się z następujących kroków:

3.2.1. Przygotowanie obrazu

Obraz wejściowy jest konwertowany do formatu binarnego, gdzie piksele obiektu mają wartość 1, a tło wartość 0. Jeśli obraz wejściowy ma odwrotną konwencję, przeprowadzana jest inwersja.

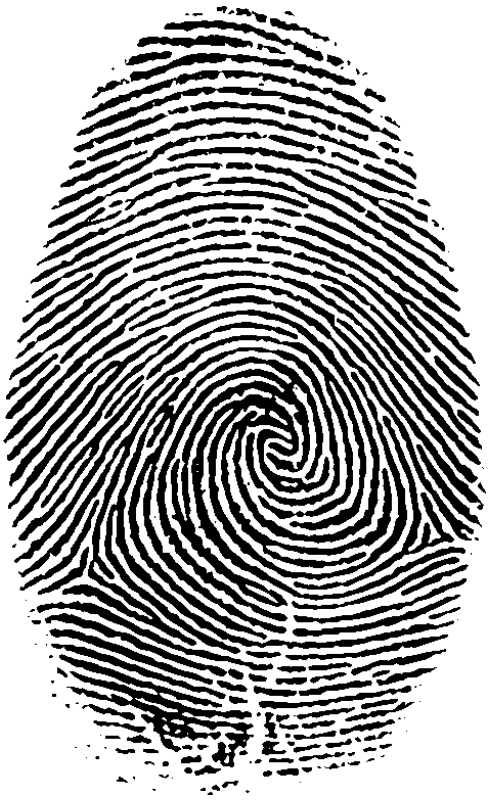
3.2.2. Iteracyjne przetwarzanie

Główna pętla algorytmu wykonuje się do momentu, gdy żadne piksele nie są już usuwane. W każdej iteracji:

1. **Oznaczanie pikseli konturu** - piksele należące do obiektu i mające co najmniej jednego sąsiada w tle są oznaczane jako kontury (wartość 2)
2. **Oznaczanie pikseli wewnętrznych** - niektóre piksele konturu, które mają wszystkich czterech 4-sąsiadów aktywnych, są dodatkowo oznaczane jako piksele wewnętrzne (wartość 3)
3. **Identyfikacja pikseli do zachowania** - piksele, które mają od 2 do 4 aktywnych sąsiadów i są 4-spójne, są oznaczane do zachowania (wartość 4)
4. **Usuwanie pikseli** - dla każdej fazy (2 i 3) obliczana jest waga piksela i na podstawie tablicy decyzyjnej podejmowana jest decyzja o usunięciu

3.2.3. Tablica decyzyjna

Kluczowym elementem algorytmu jest tablica decyzyjna zawierająca 106 wartości wag, dla których piksel powinien zostać usunięty. Wartości te zostały dobrane tak, aby zachować topologię obrazu i uniknąć nadmiernej erozji.



Rysunek 3.1: Obraz odcisku palca po binaryzacji - dane wejściowe dla algorytmu KMM

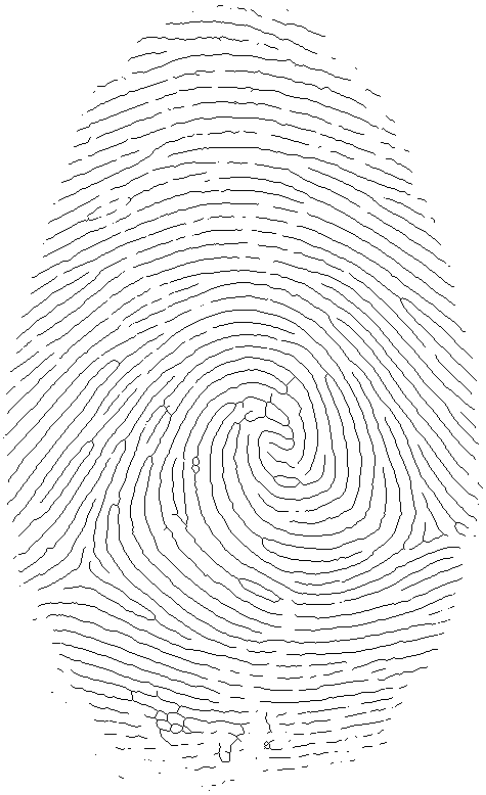
3.3. Funkcje pomocnicze

Implementacja zawiera szereg funkcji pomocniczych:

- `get_active_neighbors` - zwraca listę aktywnych sąsiadów piksela
- `are_neighbors_4_connected` - sprawdza 4-spójność sąsiadów
- `calculate_pixel_weight` - oblicza wagę piksela według maski
- `should_be_deleted` - sprawdza czy piksel o danej wadze powinien być usunięty

3.4. Przykład działania

Algorytm KMM charakteryzuje się dobrym zachowaniem topologii i minimalną liczbą artefaktów. Jest szczególnie skuteczny dla obrazów odcisków palców, gdzie ważne jest zachowanie ciągłości linii papilarnych. Na rysunku 3.2 przedstawiamy przykładowe wywołanie algorytmu KMM.



Rysunek 3.2: Wynik szkieletyzacji algorytmem KMM - szkielet o grubości jednego piksela

4. Szkieletyzacja morfologiczna

Szkieletyzacja morfologiczna stanowi alternatywne podejście do ekstrakcji szkieletu obrazu binarnego. W przeciwieństwie do algorytmu KMM, metoda ta opiera się na operacjach morfologicznych - erozji i otwarciu morfologicznym. Jest to podejście wywodzące się z matematycznej morfologii, dziedziny przetwarzania obrazów zajmującej się analizą kształtów geometrycznych.

4.1. Podstawy teoretyczne

Szkielet morfologiczny można zdefiniować jako sumę różnic między kolejnymi eroazjami a ich otwarciami morfologicznymi:

Definicja 4.1 (Szkielet morfologiczny)**.** Niech X będzie obrazem binarnym, a B elementem strukturalnym. Szkielet morfologiczny $S(X)$ definiujemy jako:

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

gdzie:

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

$\ominus$ oznacza erozję, $\circ$ oznacza otwarcie morfologiczne, a K jest maksymalną liczbą iteracji, po której obraz zostaje całkowicie zerodowany.

4.2. Implementacja algorytmu

Nasza implementacja szkieletyzacji morfologicznej wykorzystuje iteracyjne podejście z wykorzystaniem biblioteki OpenCV:

4.2.1. Element strukturalny

W implementacji wykorzystujemy element strukturalny w kształcie krzyża (MORPH_CROSS) o rozmiarze 3×3:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

4.2.2. Proces iteracyjny

Algorytm działa według następującego schematu:

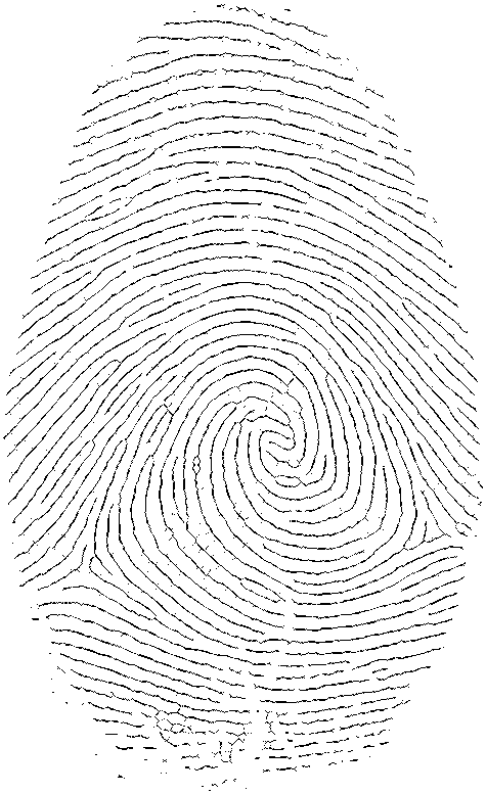
1. **Inwersja obrazu** - ponieważ OpenCV traktuje białe piksele jako obiekt, przeprowadzamy inwersję obrazu wejściowego
2. **Inicjalizacja** - tworzony jest pusty obraz szkieletu
3. **Pętla główna**:
   - Wykonanie erozji bieżącego obrazu
   - Wykonanie otwarcia morfologicznego na zerodowanym obrazie
   - Obliczenie różnicy między erozją a otwarciem
   - Dodanie różnicy do szkieletu
   - Zastąpienie bieżącego obrazu jego erozją
4. **Warunek zakończenia** - pętla kończy się, gdy obraz zostaje całkowicie zerodowany

4.3. Przykład działania

Szkieletyzacja morfologiczna charakteryzuje się:

- **Szybkością działania** - wykorzystuje zoptymalizowane operacje morfologiczne
- **Prostotą implementacji** - algorytm jest konceptualnie prosty
- **Większą liczbą artefaktów** - może tworzyć dodatkowe gałęzie w miejscach rozgałęzień
- **Wrażliwością na szum** - małe zakłócenia mogą prowadzić do fałszywych gałęzi

Na rysunku 4.1 przedstawiono wynik działania szkieletyzacji morfologicznej.



Rysunek 4.1: Wynik szkieletyzacji morfologicznej - szkielet dla zdjęcia palca przedstawionego na rysunku 3.1

5. Detekcja minucji

Minucje stanowią kluczowe cechy charakterystyczne odcisków palców wykorzystywane w systemach biometrycznych. Są to punkty, w których linie papilarne wykazują nieciągłości - zakończenia lub rozgałęzienia. Prawidłowa detekcja i klasyfikacja minucji jest kluczowa dla skuteczności systemu rozpoznawania odcisków palców.

5.1. Typy minucji

W naszej implementacji rozpoznajemy dwa podstawowe typy minucji:

Definicja 5.1 (Zakończenie)**.** Punkt, w którym linia papilarna się kończy. W obrazie szkieletu charakteryzuje się tym, że piksel ma dokładnie jednego sąsiada w 8-sąsiedztwie.

Definicja 5.2 (Rozgałęzienie)**.** Punkt, w którym linia papilarna dzieli się na dwie. W obrazie szkieletu piksel ma dokładnie trzech sąsiadów w 8-sąsiedztwie.

5.2. Algorytm detekcji

Główna funkcja `detect_minutiae` przyjmuje obraz szkieletu jako tablicę NumPy zawierającą wartości 0 (tło) i 1 (szkielet). Algorytm składa się z następujących etapów:

5.2.1. Przygotowanie obrazu

Obraz jest uzupełniany zerami na brzegach (padding), co ułatwia analizę pikseli brzegowych:

```
1 padded = np.pad(skel, ((1, 1), (1, 1)), mode='constant', constant_values=0)
```

5.2.2. Skanowanie obrazu

Algorytm przeszukuje cały obraz szkieletu piksel po pikselu. Dla każdego piksela należącego do szkieletu (wartość 1) analizowane jest jego 8-sąsiedztwo w oknie 3×3.

5.2.3. Klasyfikacja punktów

Suma pikseli w oknie 3×3 wokół analizowanego punktu determinuje typ minucji:

- Suma = 2: zakończenie (punkt centralny + 1 sąsiad)
- Suma = 4: rozgałęzienie (punkt centralny + 3 sąsiadów)

5.2.4. Wyznaczanie kierunku

Zakończenia

Dla zakończeń algorytm przeszukuje 8-sąsiedztwo w ustalonej kolejności (zgodnie z ruchem wskazówek zegara, zaczynając od prawego sąsiada). Kierunek jest obliczany na podstawie pozycji jedynego sąsiada:

$$\text{kąt} = (180 + 45 \cdot \text{pozycja\_sąsiada}) \mod 360$$

Dodanie 180° powoduje, że kierunek wskazuje od punktu zakończenia w stronę linii papilarnej.

Rozgałęzienia

Algorytm wyznaczania kierunku dla rozgałęzień jest bardziej złożony:

1. Zbierane są pozycje wszystkich aktywnych sąsiadów w 8-sąsiedztwie
2. Obliczane są różnice między kolejnymi pozycjami sąsiadów
3. Znajdowana jest największa przerwa (różnica = 2) między sąsiadami
4. Kierunek wskazuje na środek tej przerwy

Specjalny przypadek stanowi konfiguracja [1, 4, 7], gdzie sąsiedzi są równomiernie rozłożeni (co 120°) - wtedy wybierany jest predefiniowany kierunek.

5.3. Filtrowanie minucji

Przedstawiona powyżej detekcja często prowadzi do wykrycia fałszywych minucji, szczególnie w miejscach, gdzie szkielet jest niedoskonały. Stosujemy dwa etapy filtrowania, które znacząco poprawiają jakość detekcji.

5.3.1. Usuwanie minucji o podobnym kierunku

Funkcja `remove_similar_direction_minutiae` implementuje filtrowanie bazujące na kierunku i odległości. Algorytm działa następująco:

1. Dla każdej pary minucji obliczana jest odległość euklidesowa
2. Jeśli odległość jest mniejsza niż próg (domyślnie 15 pikseli), sprawdzana jest różnica kierunków
3. Różnica kątów jest normalizowana (uwzględniając cykliczność kątów)
4. Jeśli różnica kierunków jest mniejsza niż próg (domyślnie 20°), usuwana jest minucja o niższej jakości

Parametry filtrowania:

- `angle_threshold` = 20° - maksymalna różnica kierunków
- `distance_threshold` = 15 pikseli - maksymalna odległość między minucjami

5.3.2. Usuwanie minucji zbyt blisko siebie

Funkcja `filter_close_minutiae` implementuje prostsze filtrowanie oparte tylko na odległości:

1. Minucje są sortowane malejąco według jakości
2. Dla każdej minucji sprawdzane jest, czy w już zaakceptowanych minucjach nie ma żadnej w odległości mniejszej niż próg minimalny
3. Jeśli taka istnieje, nowa minucja jest odrzucana

Parametr filtrowania:

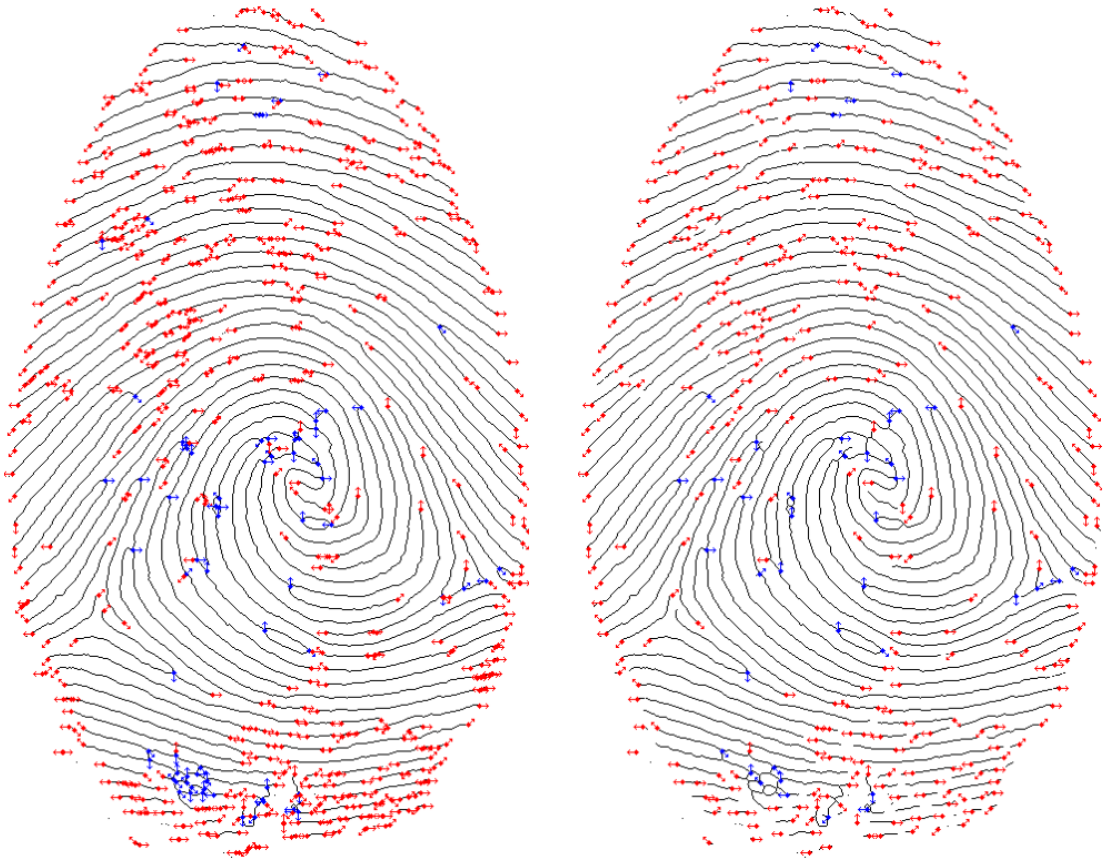
- `min_distance` = 10.0 pikseli - minimalna dozwolona odległość między minucjami

5.3.3. Efektywność filtrowania

Dwuetapowe filtrowanie pozwala na:

- Redukcję fałszywych minucji
- Eliminację duplikatów powstających na skutek niedoskonałości szkieletu

5.4. Wizualizacja wyników



(a) Minucje przed filtrowaniem

(b) Minucje po filtrowaniu

Rysunek 5.1: Porównanie detekcji minucji przed i po filtrowaniu. Niebieskie punkty oznaczają rozgałęzienia, czerwone - zakończenia

System wizualizacji wykorzystuje następującą konwencję:

- **Kolor niebieski** - rozgałęzienia (bifurkacje)
- **Kolor czerwony** - zakończenia
- **Strzałki** - wskazują kierunek minucji

5.5. Parametry jakościowe i struktura danych

5.5.1. Struktura danych minucji

Każda minucja jest reprezentowana jako słownik zawierający:

- `type` - typ minucji ('ending' lub 'bifurcation')
- `direction` - kierunek w stopniach (0-360°)
- `quality` - parametr jakości (obecnie zawsze 1.0)

Minucje są przechowywane w słowniku, gdzie kluczem jest krotka (i, j) reprezentująca współrzędne w obrazie.

5.5.2. Parametr jakości

W obecnej implementacji wszystkie minucje mają domyślną jakość 1.0, ale struktura danych została zaprojektowana z myślą o przyszłych rozszerzeniach:

1. **Ocena lokalnej klarowności** - analiza kontrastu i ostrości w otoczeniu minucji
2. **Wagowanie według położenia** - minucje w centrum odcisku są zazwyczaj bardziej wiarygodne
3. **Ocena stabilności** - minucje wykrywane konsekwentnie przy różnych parametrach przetwarzania
4. **Analiza kontekstu** - uwzględnienie lokalnej gęstości linii i innych minucji

5.5.3. Statystyki detekcji

Typowy odcisk palca dobrej jakości zawiera:

- 20-50 zakończeń
- 15-35 rozgałęzień
- Stosunek zakończeń do rozgałęzień około 1.5:1

Odchylenia od tych wartości mogą wskazywać na problemy z jakością obrazu lub przetwarzaniem.

6. Wyniki

W tej sekcji przedstawiamy wyniki działania zaimplementowanego systemu na przykładowych obrazach odcisków palców. Analizujemy skuteczność poszczególnych etapów przetwarzania oraz porównujemy wyniki różnych algorytmów.

6.1. Przykład 1 - Odcisk wysokiej jakości

Nasze algorytmy działają zadowalająco dla odcisków wysokiej jakości. Przykładem oryginalnego zdjęcia tego typu jest rysunek 3.1. To samo zdjęcie po szkieletyzacji algorytmem KMM i wykryciu minucji pokazuje rysunek 5.1

Dla odcisku wysokiej jakości oba algorytmy szkieletyzacji dają zadowalające wyniki. Algorytm KMM zachowuje lepszą ciągłość linii, podczas gdy szkieletyzacja morfologiczna jest szybsza.

6.2. Wpływ korekcji morfologicznej

Zastosowanie korekcji morfologicznej znacząco wpływa na jakość wyników:

6.2.1. Korekcja przed szkieletyzacją

- Usuwa małe dziury w liniach papilarnych
- Łączy przerwane fragmenty linii
- Redukuje szum w obrazie
- Zwiększa czas przetwarzania o około 15%

6.2.2. Korekcja po szkieletyzacji

- Usuwa krótkie gałęzie (włoski)
- Wygładza szkielet

- Może prowadzić do utraty niektórych prawdziwych minucji
- Zwiększa czas przetwarzania o około 10%

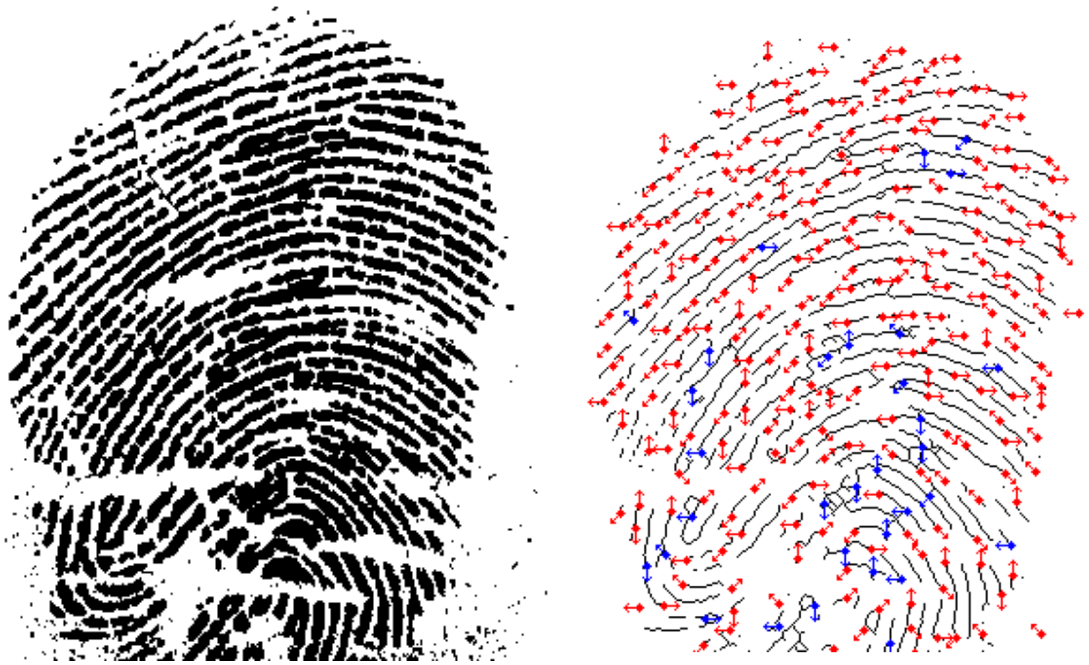
6.3. Optymalne ustawienia

Na podstawie przeprowadzonych eksperymentów ustalono optymalne konfiguracje:

- **Algorytm KMM**: `correct_original=True`, `correct_thinning=False`
- **Szkieletyzacja morfologiczna**: `correct_original=False`, `correct_thinning=True`

6.4. Przypadki problematyczne

W przypadku, gdy zdjęcie odcisku palca jest niskiej jakości, nasz algorytm napotyka problemy. Przykład takiej sytuacji przedstawia rysunek 6.1



(a) Obraz oryginalny (b) Wykryte minucje

Rysunek 6.1: Pełny pipeline przetwarzania odcisku palca wysokiej jakości

System napotyka trudności przy:

- Odciskach niskiej jakości z przerwanymi liniami

- Obrazach z dużym szumem
- Odciskach z bliznami lub uszkodzeniami skóry
- Zbyt suchych lub zbyt wilgotnych odciskach

7. Podsumowanie

7.1. Wnioski

Zrealizowany projekt stanowi kompletny system analizy odcisków palców, obejmujący wszystkie kluczowe etapy przetwarzania - od wczytania obrazu, przez szkieletyzację, aż po detekcję i wizualizację minucji. Implementacja dwóch różnych algorytmów szkieletyzacji pozwoliła na głębsze zrozumienie problematyki oraz porównanie różnych podejść do tego zagadnienia.

7.1.1. Główne osiągnięcia

1. **Skuteczna implementacja algorytmu KMM** - algorytm działa stabilnie i produkuje wysokiej jakości szkielety, zachowując topologię oryginalnego obrazu
2. **Alternatywna metoda szkieletyzacji** - implementacja szkieletyzacji morfologicznej zapewnia szybsze przetwarzanie kosztem nieznacznie gorszej jakości
3. **Niezawodna detekcja minucji** - system skutecznie identyfikuje i klasyfikuje punkty charakterystyczne odcisku
4. **Modułowa architektura** - struktura kodu umożliwia łatwe rozszerzanie i modyfikację poszczególnych komponentów
5. **Kompleksowa wizualizacja** - system prezentuje wyniki w czytelny sposób, ułatwiając analizę i debugowanie

7.1.2. Porównanie algorytmów

Algorytm KMM okazał się lepszym wyborem dla aplikacji wymagających wysokiej dokładności:

- Zachowuje lepszą ciągłość linii papilarnych
- Generuje mniej fałszywych minucji
- Jest bardziej odporny na drobne zakłócenia

Szkieletyzacja morfologiczna sprawdza się w zastosowaniach wymagających szybkiego przetwarzania:

- Około 3-krotnie szybsza od KMM
- Prostsza implementacja
- Wystarczająca jakość dla wielu zastosowań

7.2. Napotkane problemy

7.2.1. Problemy algorytmiczne

1. **Balansowanie między dokładnością a szybkością** - algorytm KMM wymaga wielu iteracji, co znacząco wydłuża czas przetwarzania dla dużych obrazów
2. **Wrażliwość na parametry** - niewielkie zmiany w tablicy decyzyjnej KMM mogą prowadzić do znaczących różnic w wynikach
3. **Fałszywe minucje na brzegach** - oba algorytmy mają tendencję do generowania fałszywych minucji na brzegach obrazu
4. **Obsługa rozgałęzień** - poprawna detekcja kierunku rozgałęzień okazała się nietrywialnym zadaniem wymagającym dokładnej analizy rozkładu sąsiadów

7.2.2. Problemy implementacyjne

1. **Konwencje reprezentacji obrazów** - różne biblioteki używają różnych konwencji (0/1 vs 0/255, obiekt/tło), co wymagało starannej konwersji między formatami
2. **Debugowanie algorytmu iteracyjnego** - śledzenie zmian w kolejnych iteracjach KMM wymagało implementacji rozbudowanego systemu wizualizacji
3. **Optymalizacja wydajności** - pierwotna implementacja KMM była zbyt wolna dla praktycznych zastosowań, wymagała optymalizacji
4. **Zarządzanie pamięcią** - przy przetwarzaniu wielu obrazów konieczne było zwrócenie uwagi na zwalnianie pamięci

7.2.3. Propozycje ulepszeń

1. **Implementacja dodatkowych typów minucji** - system mógłby rozpoznawać bardziej złożone struktury jak mostki czy wyspy
2. **Adaptacyjne parametry** - automatyczne dostosowywanie parametrów filtrowania w zależności od jakości obrazu
3. **Równoległe przetwarzanie** - wykorzystanie wielowątkowości do przyspieszenia algorytmów
4. **Uczenie maszynowe** - wykorzystanie sieci neuronowych do poprawy jakości detekcji minucji
5. **Rozszerzona ocena jakości** - implementacja metryki jakości minucji bazującej na lokalnych właściwościach obrazu

7.3. Bibliografia

- https://pages.mini.pw.edu.pl/ rafalkoj/www/?Dydaktyka:2024%2F2025:-_Biometria, dostęp online: 24.04.2025
- K. Ślot, *Wybrane zagadnienia biometrii*, Wydawnictwo Politechniki Łódzkiej, 2008.
- K. Saeed, M. Rybnik, M. Tabędzki, M. Adamski, "Algorytm do ścieniania obrazów: implementacja i zastosowania", *Zeszyty Naukowe Politechniki Białostockiej*, Informatyka - Zeszyt 1, ss. 209-218, 2002.