Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

KATEDRA AUTOMATYKI

KIERUNEK AUTOMATYKA I ROBOTYKA

PRACA MAGISTERSKA

ŁUKASZ PINDEL

METODY IDENTYFIKACJI OSÓB NA PODSTAWIE LINII PAPILARNYCH

PROMOTOR:
dr inż. Paweł Rotter

Kraków 2011

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W PRACY.

. . . . . .

PODPIS

AGH

University of Science and Technology in Krakow

Faculty of Electrical Engineering, Automatics, Computer Science and Electronics

DEPARTMENT OF AUTOMATICS

DISCIPLINE AUTOMATICS AND ROBOTICS

MASTER OF SCIENCE THESIS

ŁUKASZ PINDEL

METHODS FOR FINGERPRINT-BASED IDENTIFICATION OF PEOPLE

SUPERVISOR:
Paweł Rotter Ph.D

Krakow 2011

Serdecznie dziękuję promotorowi za poświęcony czas i wskazówki pomocne przy realizacji pracy.

Spis treści

1. Wstęp

1.1. Cel pracy

Celem pracy było stworzenie stanowiska laboratoryjnego do analizy i porównania linii papilarnych. Program miał umożliwiać analizę i dobór parametrów dla poszczególnych etapów identyfikacji: detekcji i lokalizowania cech, tworzenia modelu i jego porównywania z bazą wzorców. Stanowisko składa się z czytnika biometrycznego oraz oprogramowania opisanego w dalszych rozdziałach.

Pierwszy rozdział zawiera krótką charakterystykę linii papilarnych oraz opis zastosowanego czytnika. W kolejnych rozdziałach zostały opisane algorytmy i metody wykorzystywane do przygotowania i porównania linii papilarnych. W rozdziale 5 została opisana implementacja i szczegóły działania programów, które są przedmiotem niniejszej pracy. Rozdział 6 zawiera sprawozdanie z przeprowadzonych testów oprogramowania. Dodatki zawierają opis plików z kodem programów oraz sposób ich instalacji i przygotowanie do pierwszego uruchomienia.

1.2. Linie papilarne

Liniami papilarnymi nazywamy bruzdy znajdujące się na skórze ssaków naczelnych, w szczególności na wewnętrznej powierzchni dłoni, palcach stóp i wargach. Wysokość grzbietów na opuszkach palców wynosi od 0,1 do 0,4 mm, szerokość zwykle wynosi 0,2 - 0,7 mm. Badaniem linii papilarnych palców dłoni w celach kryminalistycznych zajmuje się daktyloskopia. U człowieka linie papilarne tworzą się jeszcze w łonie matki, gdy płód ma ok. 100-120 dni.

W 1892 roku Francis Galton wydał książkę „Fingerprints”[1], w której sformułował zasadę 3N na którą składają się 3 cechy linii papilarnych: są niepowtarzalne(nawet u bliźniąt jednojajowych), niezmienne i nieusuwalne([1, 2]). Te trzy cechy pozwalają wykorzystywać odciski palców do identyfikacji ludzi w kryminalistyce. Pierwszym krajem, który już w 1911 roku użył tej metody do potwierdzenia przestępstwa była Japonia. W Europie jednostki policyjne i milicyjne zaczęły się interesować zastosowaniem daktyloskopii w latach trzydziestych XX wieku, jednak w większości wdrożono ją dopiero po II Wojnie Światowej.

Linie papilarne możemy opisywać kilkoma metodami. Pierwsza metoda opiera się na wzorach jakie tworzą grzbiety i bruzdy na odcisku linii papilarnych. Podstawowe wzory: łukowy, pętlicowy oraz wirowy przedstawiono na rysunku 1.1.

Rysunek 1.1: Podstawowe wzory: łukowy, wirowy oraz pętlicowy (źródło: [3])

Te podstawowe typy można podzielić jeszcze na podgrupy, np. łukowy: prosty i namiotowy; wirowy: lewoskrętny, prawoskrętny, wielorodny; pętlicowy: lewy i prawy oraz wzory nietypowe niepodlegające klasycznej identyfikacji. W polskim systemie klasyfikacji daktyloskopijnej wzór namiotowy jest uznany jako czwarty typ podstawowy. Baza danych znajduje się wraz z kartami daktyloskopijnymi w Centralnej Registraturze daktyloskopijnej. Powyżej opisana klasyfikacja śladów odcisków linii papilarnych jest stosowana w systemach identyfikacji po to, by zawęzić ilość potrzebnych porównań do jednej podgrupy, zgodnej z podgrupą odcisku porównywanego.

Druga metoda oparta jest o tzw. minucje, czyli charakterystyczne punkty przebiegu grzbietów linii papilarnych. Najczęściej występującymi minucjami są rozgałęzienia i zakończenia. Do rzadszych minucji zalicza się np. haczyki, oczka, mostki, odcinki, skrzyżowania itd. Niektóre z nich zostały przedstawione na rysunku 1.2. Przy pomocy minucji, a dokładnie ich wzajemnego położenia i orientacji na obrazie odcisku palca można jednoznacznie zidentyfikować osobę, do której należy odcisk. Zakłada się, że 12 minucji jest progiem minimalnym do prawidłowego rozstrzygnięcia czy 2 ślady są tożsame, jednak prawo i procedury kryminalistyczne w poszczególnych krajach regulują ilość wymaganych minucji indywidualnie. Polskie prawo wymaga minimum 15 zgodnych minucji typu rozgałęzienie lub zakończenie, lub mniejszej ilości, jeśli będą wśród nich zgodne minucje rzadziej występujących typów.

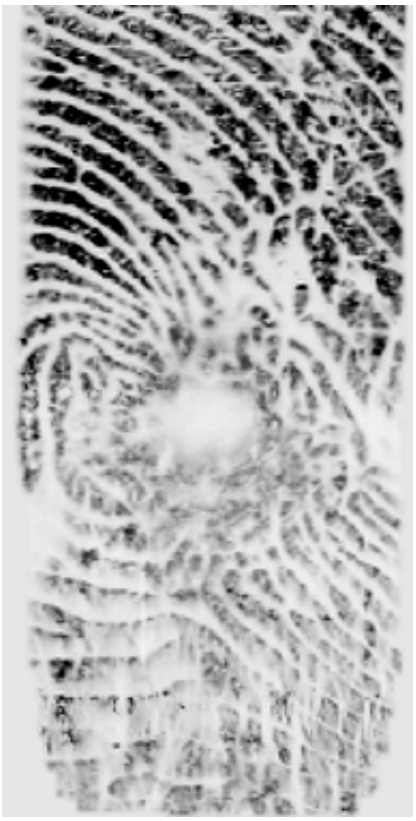
Zdarzają się też ludzie bez linii papilarnych, bądź przypadki gdy linie papilarne są z różnych powodów nieczytelne (przykłady takich nieczytelnych odcisków pokazują rysunki 1.3 oraz 1.4). Pewne rzadkie schorzenia genetyczne skóry mogą prowadzić do uszkodzeń białka kreatyny-14, czego efektem jest brak linii papilarnych. Linii papilarnych może nas też pozbawić głęboka dermabrazja (zabieg dermatologiczny) lub zażywanie pewnych leków przeciwnowotworowych (np. kapecitabiny). W obu przypadkach skóra zostaje złuszczona powodując utratę linii papilarnych.

1.3. Zastosowanie linii papilarnych w biometrii

Jak już wspomniano początki prac nad algorytmami związanymi z przetwarzaniem linii papilarnych sięgają końca XIX wieku. Wówczas odciski palców były wykorzystywane wyłącznie przez organy ścigania w celu identyfikacji przestępców na podstawie pozostawionych na miejscu zbrodni śladów. Minęło

Rysunek 1.2: Przykłady najczęściej występujących minucji

Rysunek 1.3: Struktura linii papilarnych nieczytelna wskutek poparzenia



Rysunek 1.4: Struktura linii papilarnych nieczytelna z powodu blizny

100 lat zanim linie papilarne znalazły kolejne zastosowanie jakim jest kontrola dostępu.

Jeszcze niedawno do ochrony obiektów przed niepowołanymi osobami stosowano zamki z mechanicznymi kluczami. Jednak klucze takie można zgubić. Zaczęto więc szukać nowych pomysłów. Idąc z postępem zaczęto wprowadzać systemy elektroniczne, które były łatwiejsze w użyciu i bezpieczniejsze. Systemy oparte o karty magnetyczne potwierdzane kodem PIN lub hasłem pozwoliły spać bezpiecznie użytkownikom, którzy utracili kartę. Jednak wpisywanie hasła zajmowało wiele czasu, i groziło możliwością zapomnienia lub podejrzenia hasła prze osoby niepowołane. Jeżeli nie można stosować tego co można zapomnieć (hasło, PIN), ani tego co można zgubić (klucz, karta), to należy zastosować to czym się jest. To podstawowa zasada wskazująca na zastosowanie biometrii. Biometria przez użycie czytników współpracujących z komputerami pozwala również chronić dane na nich zgromadzone. Coraz większa ilość oprogramowania umożliwia zastąpienie haseł zastosowaniem skanerów linii papilarnych.

Należy wyjaśnić czym różni się zastosowanie odcisków w daktyloskopii od zastosowania w kontroli dostępu. W procesie identyfikacji jeden ślad odcisku palca jest porównywany z wieloma zgromadzonymi wcześniej w bazie wzorców w celu identyfikacji osoby (tzw. porównanie 1:N). Długość wyszukiwania jest zależna od ilości bazowych odcisków, kartoteki policyjne mogą ich zawierać nawet miliony. W przypadku weryfikacji jeden ślad porównywany jest z jednym, (bądź maksymalnie kilkoma) zapamiętanymi wzorcami w celu potwierdzenia tożsamości (tzw. porównanie 1:1). Porównanie dwóch odcisków jest szybkie, i nawet w przypadku zastosowania wolnego komputera, czas obliczeń jest mniejszy niż czas, który użytkownik poświęciłby na wpisanie silnego hasła.

Sposoby implementacji systemów weryfikacyjnych są różne, w zależności od zadeklarowanej ilości

użytkowników systemu. W przypadku małej liczby korzystających osób można pozwolić systemowi na porównanie wczytywanego odcisku z np. pięcioma czy ośmioma odciskami zapamiętanymi w bazie. W przypadku większej ilości użytkowników wykorzystuje się numery ID. Użytkownik wpisuje swój numer ID przy pomocy klawiatury, po czym skanuje opuszek palca. System porównuje wczytaną próbkę z tylko jedną - wskazaną przez użytkownika próbką z bazy. Wykorzystywany numer ID może być znany osobom trzecim, gdyż do weryfikacji potrzebny jest i tak odcisk palca. Systemy stosujące ten pomysł coraz śmielej pojawiają się na rynku.

W przedstawionym rozwiązaniu pojawił się jednak kolejny problem: powrót do użycia klawiatury. Zauważono jednak, że szybsze od wpisywania kodu z klawiatury byłoby użycie kart magnetycznych. I również takie systemy są wprowadzane na rynek. Powstają również systemy, w których użytkownik sam decyduje czy użyć karty czy klawiatury do wprowadzenia numeru ID (przykład takiego urządzenia przedstawia rysunek 1.5). Bez względu na wybór użytkownik i tak jest bezpieczny, gdyż weryfikacja opiera się o cechy biometryczne. Producenci poszli nawet krok dalej: zaproponowano, by bazowy odcisk przechowywać nie w centralnej bazie danych, a na karcie chipowej z mikroprocesorem [4]. Zabezpiecza to użytkownika przed wyciekiem danych z systemu kontroli dostępu.

Z uwagi na to, że ułożenie palca na czytniku podczas pobierania poszczególnych próbek jest zróżnicowane, dopasowanie 100% zgodności nie jest możliwe. Dlatego algorytmy porównujące powinny wyznaczyć stopień zgodności modeli reprezentujących obie próbki i porównać z zadeklarowaną akceptowalną niedokładnością. Jeżeli zgodność porównywanych próbek jest większa niż próg zgodności (czyli maksymalnej akceptowanej niedokładności), system akceptuje wczytaną próbkę orzekając potwierdzenie tożsamości. Jednak automatycznym systemom biometrycznym zdarzają się błędy. Aby określić poziom tych błędów wprowadzono dwa określenia:

FAR (False Acceptance Rate - Poziom błędnych akceptacji) - parametr określa ilość porównań, które powinny zostać odrzucone, a przez pomyłkę zostały zaakceptowane,

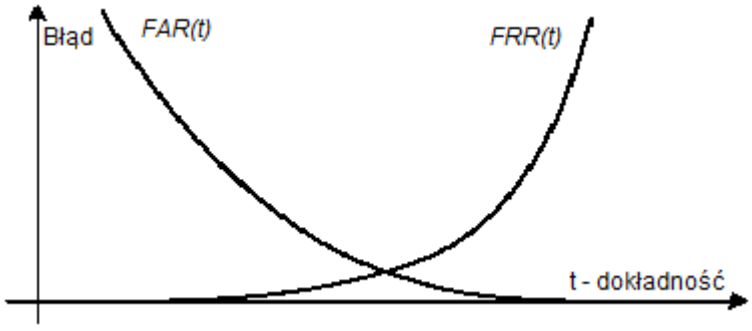
FRR (False Rejection Rate - Poziom błędnych odrzuceń) - parametr określa ilość porównań, które powinny zostać pozytywnie rozpatrzone, podczas gdy zostały uznane za nieprawidłowe.[5]

Współczynniki FAR i FRR silnie zależą od ustawionego progu zgodności. Większość produkowanych obecnie systemów posiada FAR < 0,001 i FRR < 0,01[6]. Dla podanych wartości oznacza to, że system odrzuci około 10 na 1000 prawidłowych porównań, oraz zaakceptuje około 1 na 1000 prób podszycia się pod inną osobę. Wartości współczynników są powiązane i obrazują wpływ zmian progu zgodności na działanie systemu (przedstawia to rysunek 1.6).

1.4. Pobranie odcisków palców

Pobieranie odcisków palców można dokonać dwiema metodami: pośrednią oraz bezpośrednią. Pośrednia metoda polega na pobraniu odcisku z użyciem pośredniego medium, najczęściej atramentu. W niektórych przypadkach jest to jedyna metoda akwizycji możliwa do zastosowania. Tak jest głównie w kryminalistyce gdy zachodzi potrzeba pobrania śladów pozostawionych na różnych powierzchniach

Rysunek 1.5: Centralka ze skanerem linii papilarnych i czytnikiem kart magnetycznych (źródło: [7])



Rysunek 1.6: Wpływ współczynników FAR i FFR na dokładność systemu biometrycznego (źródło: [5])

i przedmiotach przez nieznane osoby. Do pobrania takich odcisków używa się odpowiednich środków chemicznych.

Metoda bezpośrednia polega na odczytywaniu struktury linii papilarnych bezpośrednio z opuszka palca przez przeznaczony do tego skaner. Skanowanie odbywa się na jeden z dwóch sposobów:

– przez przyłożenie palca do powierzchni sensora,

– przez przesunięcie palca po jego powierzchni.

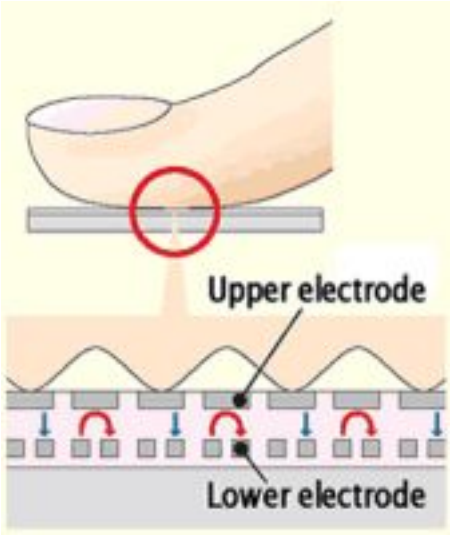
W pierwszym przypadku czytnik posiada obszar, do którego należy przyłożyć palec, w drugim przypadku obszar skanowania jest bardzo wąski, i w czasie pozesuwania palca czytnik składa odcisk z zeskanowanych pasków. Najczęściej stosowane typy skanerów zostaną po krótce opisane w dalszej części pracy.

1.4.1. Skaner optyczny

Pierwsze urządzenia do pobierania odcisków palców były skanerami optycznymi. Ich działanie polega na fotografowaniu powierzchni palca. Nowsze modele używają do tego matryc CCD(ang. Charge Coupled Device) czyli matryc zbudowanych ze światłoczułych elementów. Matryca taka generuje obraz w postaci cyfrowej. Zaletą takiego czytnika jest brak bezpośredniego kontaktu palca z sensorem (sensor się nie brudzi, nie koroduje). Jednak czytniki te są niezbyt skuteczne, gdyż wytworzony przez nie obraz ma znikomy kontrast i łatwiej można na nim zobaczyć brud niż linie papilarne. Niemożliwe jest zeskanowanie palców z nawet lekko zniszczonym naskórkiem. Poza tym bardzo łatwo można taki czytnik oszukać przez przyłożenie do czytnika spreparowanego odlewu (np. z silikonu), a nawet skanując pozostawiony na powierzchni czytnika tłusty ślad po palcu(!)[5].

1.4.2. Skaner pojemnościowy

Sensor czytnika pojemnościowego jest zbudowany z matrycy komórek. Każda komórka składa się z dwóch płytek przewodzących oddzielonych dielektrykiem. Powierzchnia palca tworzy trzecią warstwę przewodzącą odseparowaną kolejną warstwą dielektryka, a w przypadku doliny linii papilarnej również warstwą powietrza. W zależności od odległości pomiędzy trzema warstwami przewodzącymi zmienia się pojemność kondensatora w komórce. Pojemność ściśle zależy od odległości palca od sensora, zatem kondensator pod grzbietem linii papilarnej ma wysoką pojemność, a pod doliną linii papilarnej niską pojemność. Podając na każdy kondensator w matrycy jednakowy ładunek uzyskuje się napięcie wyjściowe wprost proporcjonalne do odległości powierzchni palca od sensora, które jest dalej przetwarzane na postać cyfrową i podawane na wyjście jako piksel obrazu. Ogólna zasada działania czytnika pojemnościowego została pokazana na rysunku 1.7 [8]. Ważną zaletą czytników pojemnościowych jest wymaganie prawdziwego i żywego palca, a nie tylko wypukłego jego obrazu. Są też stosunkowo odporne na niewielkie zabrudzenia. Jedyną wadą czytnika pojemnościowego jest zależność grubości grzbietów na obrazie od wilgotności naskórka i siły docisku palca do czytnika (mokry palec lub silny docisk dają szersze grzbiety niż normalnie)[6], wadę tą można jednak skorygować odpowiednim algorytmem filtracji.



Rysunek 1.7: Ogólna zasada działania czytnika pojemnościowego (źródło: [8])

1.4.3. Skaner termiczny

Skaner buduje informację o układzie linii papilarnych na podstawie różnic temperatury pomiędzy grzbietami linii papilarnych a powietrzem będącym w dolinach między tymi grzbietami. Skanery te są odporne na brud, mokre palce i zniszczony naskórek. Nie dają się oszukać sztucznym odlewom. Odmiana czytników, w której należy przesunąć palec po jego powierzchni daje obraz lepszej jakości niż odmiana, w której skanowanie odbywa się przez przyłożenie palca. Powodem jest lepsze doprowadzenie chłodnego powietrza do dolin linii papilarnych na czas skanowania. Sporą wadą czytników termicznych jest duże zapotrzebowanie na energię, gdyż sensor musi być w czasie pomiaru cieplejszy niż palec.[6]

1.4.4. Skaner ultradźwiękowy

Zasada działania bazuje na tym, że gdy do powierzchni ciała stałego, do której dociera dźwięk(czyli czytnika) przyłożymy inny obiekt (czyli palec), a kontakt między nimi nie jest idealny ze względu na różnego rodzaju krawędzie, bruzdy (w przypadku palca są to linie papilarne), to w miejscach kontaktu tych dwóch obiektów dochodzi do wielu zjawisk. Są to przede wszystkim dyfrakcja i odbicie dźwięku opisane klasycznymi wzorami[9], ale również dodatkowe rozproszenia wraz z przemianami na inne rodzaje fal. Nazwano je rozproszeniem kontaktowym, gdyż powstaje na skutek zmiany warunków propagacji fal przez zmienny kontakt miedzy dwoma ciałami stałymi. Ważne jest to, że każda powierzchnia oraz każdy materiał ma inny wpływ na rozproszenie kontaktowe. Pozwala to na skuteczną ocenę czy do czytnika przyłożono prawdziwy palec, a następnie umożliwia odwzorowanie jego powierzchni (czyli wzoru linii papilarnych). Co ważne, ultradźwięki przenikają przez brud i wilgoć, więc skaner jest odporny na zakłócenia przez nie generowane.[10]

Rysunek 1.8: Czytnik EIKON firmy UPEK (źródło: [11])

1.4.5. Skaner wybrany do celów aplikacji

Do pozyskiwania obrazów linii papilarnych użyto skanera EIKON firmy UPEK. Fotografia urządzenia przedstawiona jest na rysunku 1.8.

EIKON to skaner pojemnościowy, dzięki bibliotekom udostępnionym przez producenta jest obsługiwany przez aplikację, opisaną w sekcji 5.2. Przystępny opis biblioteki ([12]) ułatwia pracę programiście. Skanowanie linii papilarnych odbywa się przez przesunięcie palca po powierzchni sensora. Rozdzielczość otrzymanego obrazu wynosi 508 dpi, a jego rozmiar 192×512 pikseli. Komunikacja z komputerem odbywa się przez USB 2.0.

Na wybór skanera wpłynęły takie aspekty jak:

– dobra jakość obrazu otrzymanego ze skanera,

– dość duża odporność na niesprzyjające warunki pomiaru (np. wpływ zabrudzeń na jakość obrazu),

– dostępność urządzenia w sprzedaży,

– brak możliwości oszukania skanera (wymaga żywego palca - cecha skanera pojemnościowego).

2. Filtracja obrazu linii papilarnych

Filtracja obrazu linii papilarnych ma na celu poprawę jakości wczytanego obrazu zawierającego ślad linii papilarnych. Filtracja odbywa się w kilku etapach. Działania podejmowane na poszczególnych etapach filtracji zostaną opisane w tym rozdziale.

2.1. Tworzenie mapy orientacji

Na wstępie rozpocznie się tworzenie mapy orientacji grzbietów będących na obrazie (rysunek 2.1 b). Algorytm, który tego dokonuje jest przepisaną na język C++ wersją algorytmu autorstwa Raymonda Thai poprawionego przez Petera Kovesi [13], który w oryginale jest wykorzystany m.in. w pracy [14].

Generowanie mapy orientacji rozpoczyna się od utworzenia maski filtru Gaussa. Następnie liczony jest gradient tej maski w osiach X oraz Y. Używając powstałych masek, filtruje się obraz wejściowy otrzymując gradienty obrazu w kierunkach X i Y, odpowiednio G_x oraz G_y. Oszacowanie lokalnych orientacji grzbietów w każdym punkcie następuje poprzez znalezienie głównej osi zmienności gradientu obrazu. Zatem wyznacza się dane kowariancji gradientu obrazu poprzez wymnożenie macierzy gradientu czynnik po czynniku zgodnie z równaniami 2.1, 2.2, 2.3.

$$K_{xx} = G_x^2 \tag{2.1}$$

$$K_{xy} = G_x G_y \tag{2.2}$$

$$K_{yy} = G_y^2 \tag{2.3}$$

Wyznaczone w ten sposób macierze kowariancji wygładza się filtrem Gaussa, po czym wyznaczany jest główny kierunek zmian dla każdego punktu, wykorzystując równanie 2.4.

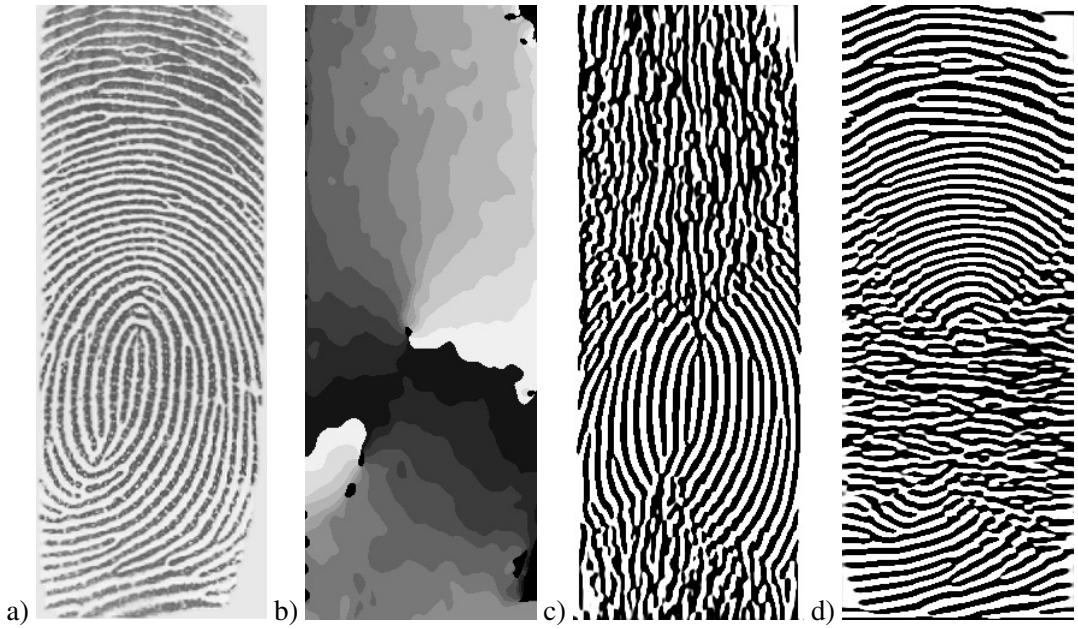
$$dzielnik = \sqrt{K_{xy}^2 + (K_{xx} - K_{yy})^2} \tag{2.4}$$

Na podstawie powstałej macierzy wyznacza się sinusoidę oraz cosinusoidę w dwóch osiach (w postaci macierzy) jak pokazują równania 2.5 i 2.6.

$$\sin(K) = \frac{K_{xy}}{dzielnik} \tag{2.5}$$

$$\cos(K) = \frac{(K_{xx} - K_{yy})}{dzielnik} \tag{2.6}$$

gdzie K to umowne oznaczenie kierunku na obrazie.



Rysunek 2.1: Filtracja: a) obraz z czytnika, b) mapa orientacji, c) obraz po filtracji pionowej, d) obraz po filtracji poziomej

Po wygładzeniu tych macierzy filtrem Gaussa wyznacza się ostatecznie orientację każdego punktu zgodne ze wzorem 2.7.

$$orientacja = \arctan(\frac{\sin(K)}{\cos(K)}) \tag{2.7}$$

Mapa orientacji ma służyć w dalszej części algorytmu do selekcji fragmentów przefiltrowanych obrazów do sklejania. Powoduje to, że orientacja nie może być podawana co do wartości, ale przedziałami. Dzieje się tak dlatego, że dla każdej wartości występującej na mapie orientacji trzeba utworzyć oddzielny filtr. Wybrano zatem 12 przedziałów, każdy obejmujący zakres orientacji o szerokości 15^o. Jako wynik, zwracany jest dla każdego piksela numer przedziału, w którym mieści się jego lokalna orientacja.

2.2. Filtracja Gabora

Kolejnym etapem jest przygotowanie zestawu filtrów. Na początku obraz poddawany jest normalizacji. Ma to na celu zmniejszenie kontrastu przed dalszą filtracją. Następnie tworzona jest grupa filtrów Gabora. Filtr Gabora jest liniowy, jego maskę można scharakteryzować jako dwuwymiarową funkcję Gaussa, która jest modulowana zespoloną sinusoidą. Wzór opisujący filtr Gabora przedstawia równanie 2.8[15].

$$g(x, y; \lambda, \theta, \psi, \sigma_x, \sigma_y) = \exp\left(-\frac{x'^2}{2\sigma_x} + \frac{y'^2}{2\sigma_y}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right) \tag{2.8}$$

gdzie:

$x' = x\cos\theta + y\sin\theta$

$y' = -x\sin\theta + y\cos\theta$

λ - długość fali sinusoidalnej

θ - orientacja filtru

ψ - przesunięcie fazy

σ_x, σ_y - odchylenie standardowe powierzchni Gaussa

Cechą charakterystyczną tego filtru jest selektywność częstotliwości oraz kierunku. Oznacza to, że pojedynczy filtr, wykryje tylko fakturę obrazu zgodną z częstotliwością i orientacją samego filtru (np. rysunki 2.1 c, d przedstawiają wynik filtracji odpowiednio pionowym i poziomym filtrem Gabora). Zatem aby przefiltrować obraz, na którym kierunki zmian faktury są różne w różnych częściach obrazu, należy użyć wielu filtrów o różnych współczynnikach θ. Gdyby zaszła potrzeba filtrowania wielu częstotliwości, należałoby stworzyć również filtry z różnymi współczynnikami λ, jednak do filtracji linii papilarnych nie jest to konieczne ze względu na jednakową strukturę całego śladu. Filtr wykorzystany w aplikacji składa się z 12 masek, każda z inną orientacją θ. Liczba masek nie przypadkowo równa się liczbie przedziałów na mapie orientacji. W efekcie każdy obszar wykryty na mapie orientacji ma swój dedykowany filtr. Następnie poszczególne przefiltrowane fragmenty obrazu są sklejane na podstawie wygenerowanej wcześniej mapy orientacji, tworząc gotowy binarny obraz po filtracji (rysunek 3.1 a). Program umożliwia użytkownikowi samodzielne dobranie współczynników λ oraz σ, gdzie $\sigma_x = \sigma$, oraz $\sigma_y = \sigma$. Przesunięcie fazy ma wartość zerową.

2.3. Finalizowanie filtracji

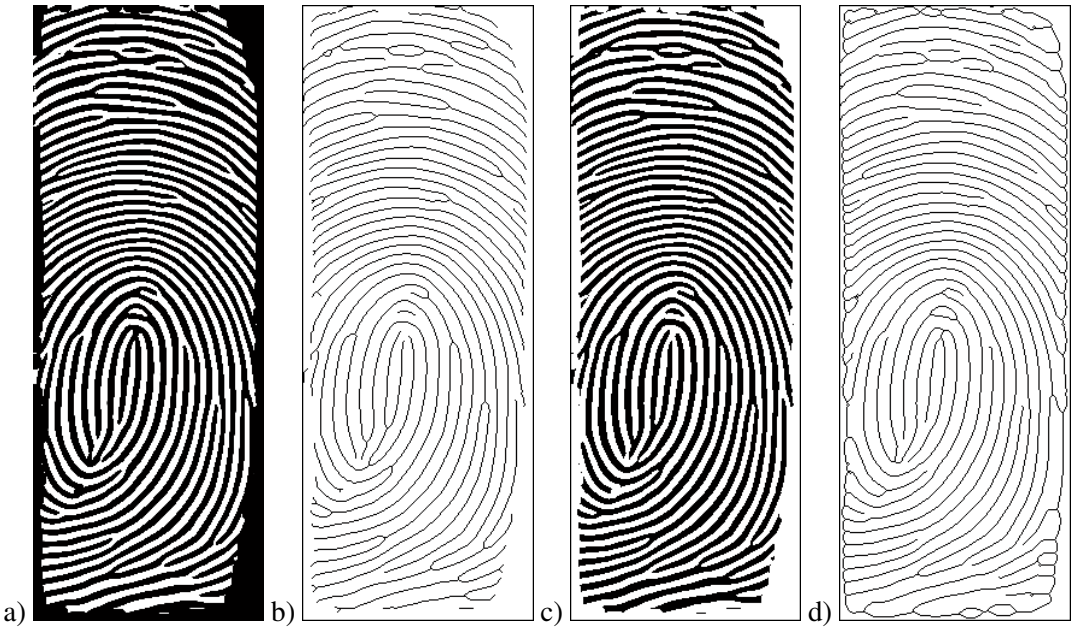
W następnej kolejności na podstawie pliku wejściowego tworzone są dwie dodatkowe maski filtracyjne (jedną z nich przedstawia rysunek 3.2 a). Niefiltrowany plik wejściowy jest binaryzowany, po czym dokonuje się morfologicznego zamknięcia obrazu. Próg binaryzacji, jak również rozmiar elementu strukturalnego do zamknięcia są parametrami, możliwymi do zmiany przez użytkownika. W efekcie zamknięcia otrzymywana jest maska, która przemnożona przez otrzymany wcześniej w wyniku filtracji obraz powoduje jego obcięcie zgodnie z konturem źródła. Obcięcie jest niezbędne, gdyż podczas tworzenia mapy orientacji w miejscach, gdzie nie było danych na obrazie (czyli w przypadku, gdy odcisk nie zajmuje całego obrazu) dane o orientacji takiego pustego obszaru są przypadkowe, powodując fałszywe powiększenie obszaru obrazu. Nałożenie odpowiedniej maski ograniczającej kontur obrazu wynikowego do konturu obrazu źródłowego pozwala się przed tym skutecznie zabezpieczyć. Aby otrzymać drugą maskę, poddaje się erozji maskę otrzymaną wcześniej. Maska ta ogranicza obszar bardziej niż poprzednia i zostanie wykorzystana dopiero w następnej sekcji.

3. Wykrywanie minucji

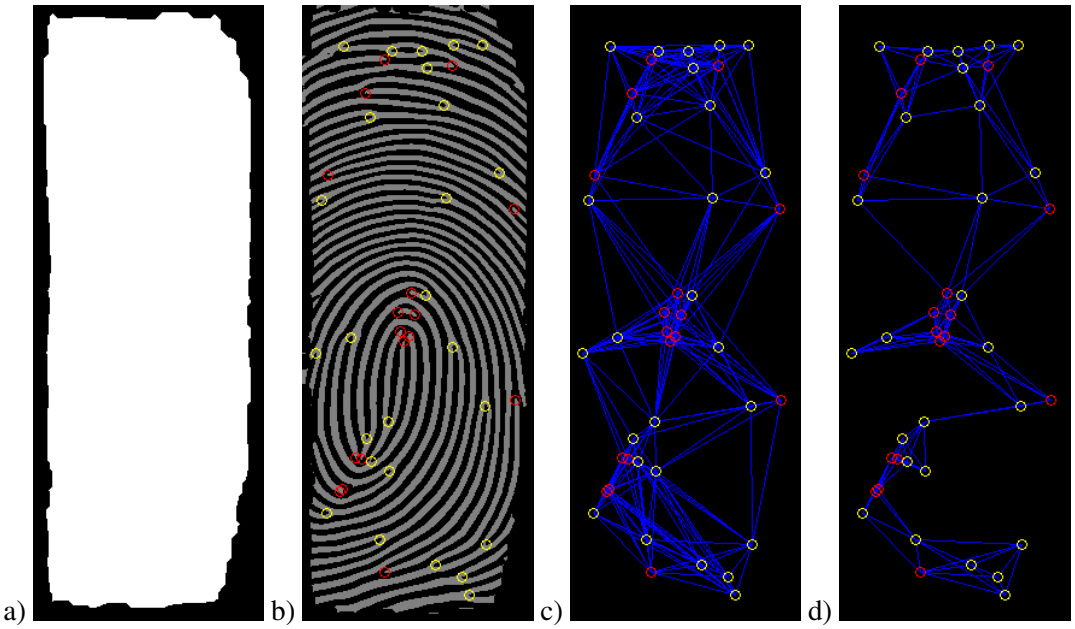
W tej części na przygotowanym wcześniej obrazie wyszukiwane są minucje. Program skupia się na wyszukiwaniu tylko dwóch, ale najczęściej występujących typów minucji: zakończeń i rozgałęzień. Grzbiety linii papilarnych na grafikach są oznaczone na biało (tak domyślnie jest reprezentowana w MATLABIE binarna *jedynka*), natomiast doliny na czarno (binarne *zero*). Łatwo w takim obrazie odnaleźć zakończenia, natomiast precyzyjne znalezienie rozgałęzień jest trudniejsze. Konieczne są wtedy skomplikowane algorytmy (np. opisane w [16]). Z uwagi na specyfikę tych typów minucji, oraz fakt iż obraz wejściowy jest binarny, łatwo można przekształcić algorytm, i wyszukiwać oba typy minucji wykorzystując do tego jeden algorytm. Dlatego tworzy się kopię wejściowego obrazu, oraz odwraca się w niej kolory (negatyw), co przedstawia rysunek 3.1 c. Powoduje to, że na drugim obrazie to doliny są na biało (*jedynki*) , a grzbiety na czarno (*zera*). Korzyść z tej operacji polega na tym, że analizując drugi obraz tym samym algorytmem, uzyskujemy drugi typ minucji (zakończenie doliny odpowiada rozgałęzieniu grzbietu).

Aby precyzyjnie wyznaczyć lokalizacje minucji, obrazy (źródłowy i odwrócony - oba będą analizowane jednocześnie) poddawane są szkieletyzacji. Operacja ta polega na zmniejszeniu grubości wszystkich łuków (grzbietów na pierwszym obrazie oraz dolin na drugim obrazie) do grubości jednego piksela (rysunki 3.1 b, d przedstawiają szkielety, widać brak konsekwencji oprogramowania MATLABA, gdyż teraz *jedynki binarne* reprezentowane są na obrazie kolorem czarnym). Otrzymane w ten sposób obrazy są przeszukiwane piksel po pikselu, w celu znalezienia zakończeń szkieletu. Dla każdego analizowanego piksela badane jest otoczenie, czyli 8 pikseli z nim sąsiadujących. Jeżeli piksel badany jest czarny, a wokół niego czarny jest dokładnie jeden sąsiad, to oznacza że analizowany piksel jest miejscem występowania minucji. Jak już wspomniano na początku pracy, analizowane obrazy są jedynie fragmentami śladów linii papilarnych. Często krawędź obrazu odcina część odcisku znajdującą się poza czytnikiem. Daje to wiele dodatkowych zakończeń szkieletu, które w rzeczywistości nie są minucjami. Jednak łatwo jest zlokalizować tego typu fałszywe minucje na krawędziach, gdyż znajdują się one w pobliżu brzegu śladu linii papilarnej wczytanego na oryginalnym obrazie. Wystarczy więc pominąć te minucje, które znajdują się w obszarach brzegowych, i do tego służy przygotowana wcześniej maska nr 2 (rysunek 3.2 a). Po nałożeniu maski na obraz z minucjami, minucje znajdujące się na zewnątrz obszaru maski są odrzucane. Pozostałe punkty są wynikiem prawidłowego przetwarzania (przedstawia je rysunek 3.2 b). Prawidłowe minucje są zapamiętywane wraz z cechami, które je identyfikują. Wartościami zapisywanymi są:

– numer minucji (minucje są numerowane kolejnymi liczbami naturalnymi)



Rysunek 3.1: Szkieletyzacja: a) obraz po filtracji, b) szkieletyzacja obrazu, c) negatyw obrazu przefiltrowanego, d) szkieletyzacja negatywu



Rysunek 3.2: Wyszukiwanie minucji: a) maska obcinająca fałszywe minucje b) znalezione minucje: czerwone rozgałęzienia, żółte zakończenia, c) graf z limitem 10 krawędzi, d) graf z limitem 5 krawędzi

– współrzędna X na obrazie

– współrzędna Y na obrazie

– typ minucji (cyfrą 1 oznaczono zakończenia, 2 oznaczono rozgałęzienia)

– orientacja grzbietu w miejscu wystąpienia minucji (odczytana z mapy orientacji)

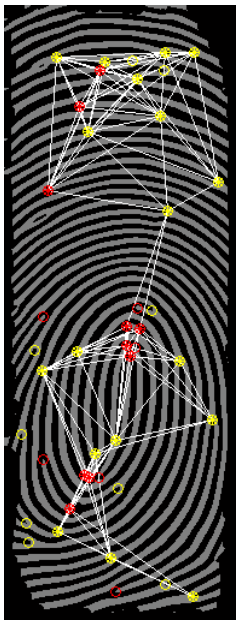
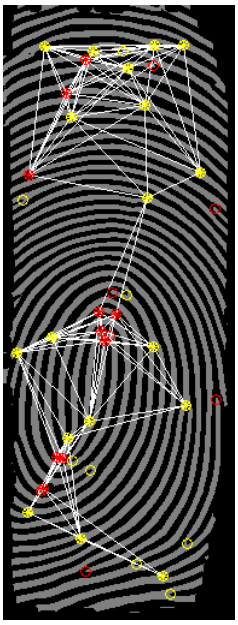
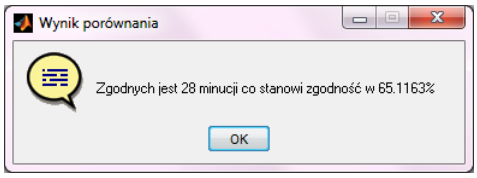
Po sporządzeniu listy minucji, tworzony jest graf opisujący je. Wierzchołkami grafu są minucje. Wierzchołkom przypisuje się wszystkie zapamiętane cechy odpowiadających im minucji. Wybrane wierzchołki łączy się krawędziami. Ilość krawędzi zależna jest od parametru. W przypadku grafu dla odcisku bazowego, łączone są wszystkie dostępne wierzchołki (taki graf nazywa się grafem pełnym). W przypadku grafu dla odcisku porównywanego, ilość krawędzi dla każdego wierzchołka jest ograniczona do 5 lub 10 (rysunki 3.2 c, d). Wybór zależy od użytkownika. Domyślnie kontrolka jest zaznaczona (co oznacza 10 krawędzi dla każdego wierzchołka). Krawędzie wchodzące w skład grafu wybierane są w ten sposób, aby łączyć bieżący wierzchołek z najbliższymi wierzchołkami sąsiednimi. Gwarantuje to spójność grafu, co z kolei przekłada się na skuteczność porównań w dalszej części programu. Ograniczenie ilości krawędzi jest podyktowane ilością obliczeń wymaganych podczas porównywania grafów. Jeżeli analizowane są 2 grafy pełne, to ilość obliczeń jest kilkukrotnie większa niż w przypadku grafu z mniejszą ilością krawędzi, natomiast nie ma to wpływu na poprawę wyniku. Nie należy jednak ograniczać ilości krawędzi do zbyt małej ich liczby, gdyż grozi to naruszeniem spójności grafów i skomplikowanie danych do porównań. Parametr zmieniający ilość tworzonych krawędzi pozwala zaobserwować wpływ ilości krawędzi na spójność grafu. Ten wpływ jest różny w zależności od liczebności wierzchołków, i jest mniejszy dla grafów mniej liczebnych, oraz grafów, których wierzchołki są równomiernie rozproszone, a nie skupione w grupach.

4. Porównanie układów minucji

Porównanie układów minucji polega na porównaniu dwóch grafów będących modelami opisującymi dwa ślady linii papilarnych. Należy we wspomnianych grafach wyszukać takie podgrafy, które będą względem siebie izomorficzne [17]. Badania nad algorytmami poszukiwania izomorfizmu w podgrafach są prowadzone od dawna przez liczne ośrodki badawcze[18], jednak żaden z istniejących algorytmów nie spełnia wszystkich wymogów opisywanej aplikacji. Istnieje wprawdzie kilka algorytmów o zbliżonym działaniu (np. algorytm J. R. Ullmanna z 1976 roku [19]) jednak wszystkie te algorytmy zakładają całkowite zawieranie się jednego grafu w drugim. Z uwagi na specyfikę obrazów w niniejszym opracowaniu, niemożliwe jest zagwarantowanie zawierania się grafów w bieżącym zastosowaniu, nawet w przypadku pełnej zgodności odcisków. Dzieje się tak dlatego, iż każdy obraz prezentuje nieco inny obszar opuszka palca. Wynika z tego, że zwykle pokrywają się jedynie pewne fragmenty obrazu, a co za tym idzie tylko pewne fragmenty grafów.

W związku z powyższymi problemami został opracowany własny algorytm przybliżony. Polega on na wyszukaniu w obu grafach takich podgrafów, które są względem siebie izomorficzne. Idea funkcji porównującej opiera się na przeszukiwaniu wszerz dwóch grafów jednocześnie. Na początku z jednego grafu wybierany jest dowolny wierzchołek. Następnie przeszukiwane są wszystkie wierzchołki drugiego grafu, w celu sprawdzenia, czy którykolwiek z nich ma takie same cechy charakterystyczne (typ i orientacja minucji odpowiadającej danemu wierzchołkowi) jak wybrany wierzchołek z pierwszego grafu. Jeśli żaden z wierzchołków nie pasuje, to z pierwszego grafu wybierany jest kolejny wierzchołek i sprawdzanie odbywa się do skutku. W przypadku przeszukania całego grafu i braku dopasowania odpowiedniego wierzchołka program kończy działanie ogłaszając brak zgodności między grafami. Jeżeli jednak zostanie znaleziony wierzchołek (lub kilka wierzchołków) ze zgodnymi parametrami, to zostają one kolejno weryfikowane, i jeśli potwierdzi się zgodność któregoś z nich, to program przejdzie do kolejnego etapu by poszukać kolejnych zgodnych wierzchołków. Jeśli żaden z wierzchołków nie okaże się zgodny, a cała lista kandydatów zostanie sprawdzona to program powróci do samego początku, wybierając kolejny wierzchołek z pierwszego grafu.

Weryfikacja polega na przeszukaniu sąsiadów obu wierzchołków. Jeżeli oba wierzchołki posiadają co najmniej jednego sąsiada w tej samej odległości (z tolerancją określoną parametrami) i sąsiedzi ci mają takie same cechy charakterystyczne, to zgodność wierzchołków-kandydatów zostaje potwierdzona i kandydaci zostają pierwszymi zgodnymi wierzchołkami. Zgodni sąsiedzi zostają natomiast wpisani na listę wierzchołków czekających na sprawdzenie (analogicznie jak w algorytmie przeszukiwania wszerz). Mając pierwsze zaakceptowane wierzchołki, za kandydatów przyjmuje się pierwsze wierzchołki dodane

a)  b)  c) 

Rysunek 4.1: Linie papilarne z naniesionymi zgodnymi częściami grafów: a) odcisk bazowy, b) odcisk porównywany, c) komunikat z podsumowaniem

wcześniej do listy do sprawdzenia (po jednym z obu grafów). Podobnie jak wcześniej poszukiwani są sąsiedzi w podobnej odległości i o tych samych cechach, w przypadku powodzenia zostają dopisani na koniec listy do sprawdzenia, a sprawdzana para wierzchołków uznana za zgodną.

Gdy lista wierzchołków do sprawdzenia się wyczerpie, dokonuje się ostatniej kontroli: każdy wierzchołek powstałego podgrafu, który ma tylko jednego sąsiada jest mało wiarygodny, gdyż jedyną informacją o jego odległości od tego sąsiada będzie fakt iż znajduje się on na okręgu o promieniu równym długości krawędzi. Dlatego wierzchołki takie są usuwane, a wierzchołki posiadające co najmniej dwóch sąsiadów pozostają niezmienione, gdyż minimum dwóch sąsiadów jednoznacznie określa ich położenie.

Po tej weryfikacji sprawdzana jest liczebność zgodnego podgrafu. W tym momencie program zakończy działanie jeżeli spełniony jest jeden z dwóch warunków:

– liczba zgodnych wierzchołków jest większa niż wymagana przez użytkownika wartość minimalna (domyślnie 12)

– stosunek liczby zgodnych wierzchołków do wszystkich wierzchołków *nowego* grafu jest większy niż podany przez użytkownika próg (domyślnie 40%)

Jeżeli żaden z powyższych warunków nie jest spełniony, to zapamiętywana jest ilość znalezionych zgodnych wierzchołków (jako maksymalna znaleziona ilość wierzchołków dla danych grafów), i program wraca do początku szukając nowych, niesprawdzonych jeszcze kandydatów.

W kolejnych iteracjach, gdy uda się znaleźć zgodny podgraf, ponownie jest badana liczebność wierzchołków. Jeżeli zostaną sprawdzone wszystkie możliwe dopasowania, a mimo to liczba zgodnych wierzchołków jest mniejsza niż wymagany próg, to grafy zostają uznane za niezgodne. Jeżeli jednak zostało znalezione jakiekolwiek dopasowanie (nawet niespełniające powyższych progów) to program zwróci właśnie to maksymalne dopasowanie. Takie działanie jest podyktowane faktem, że aplikacja umożliwia

użytkownikowi dobór najważniejszych parametrów. Zatem przez zwracanie maksymalnego znalezionego dopasowania umożliwione jest badanie wpływu zmian parametrów na działanie algorytmu. Ich wpływ zostanie opisany w rozdziale 6 opisującym testy oprogramowania, a pokazanie użytkownikowi wyników może pomóc dobrać najlepsze współczynniki.

Użytkownikowi zostaje przedstawiony obraz bazowy z naniesionymi minucjami oraz tą częścią grafu, która jest zgodna z obrazem porównywanym. Podobnie zostaje zaprezentowany obraz porównywany, z naniesionymi minucjami i zgodną częścią grafu (rysunki 4.1 a, b). Oprócz tego, użytkownik otrzymuje komunikat dotyczący ilości zgodnych wierzchołków, oraz procentowy udział zgodnych wierzchołków w całym grafie (rysunek 4.1 c).

5. Implementacja

5.1. Wstęp

Główne okno programu zostało ono przedstawione na rysunku 5.1. Okno podzielone jest na ponumerowane sekcje. Każda sekcja realizuje odrębny algorytm. Każda sekcja działa niezależnie od pozostałych, ale wymaga danych wyliczonych przez poprzednie sekcje. Program umożliwia samodzielne wybranie parametrów wedle uznania przed wykonaniem kodu. Algorytm zaczyna działanie w momencie wciśnięcia odpowiedniego przycisku w oknie głównym. Przyciski są ponumerowane, aby użytkownik mógł łatwiej nimi operować.

Po uruchomieniu programu wszystkie parametry mają wartości domyślne. Należy uruchamiać kolejno wszystkie sekcje (z domyślnymi lub zmienionymi parametrami). Podczas kolejnych eksperymentów w bieżącej sesji (tzn. przed zamknięciem aplikacji) w przypadku zmiany jakiegoś parametru nie trzeba powtarzać wszystkich obliczeń od początku, a jedynie od sekcji, w której nastąpiły zmiany do końca. Jeżeli użytkownik po obejrzeniu wyników zmieni parametry np. w przedostatniej sekcji to konieczne jest powtórzenie obliczeń tylko dwóch ostatnich sekcji.

W każdej sekcji po przetworzeniu danych zostanie wyświetlony od jednego do kilku obrazów prezentujących wyniki obliczeń (obrazy te są prezentowane na omawianych wcześniej rysunkach 2.1, 3.1, 3.2, 4.1).

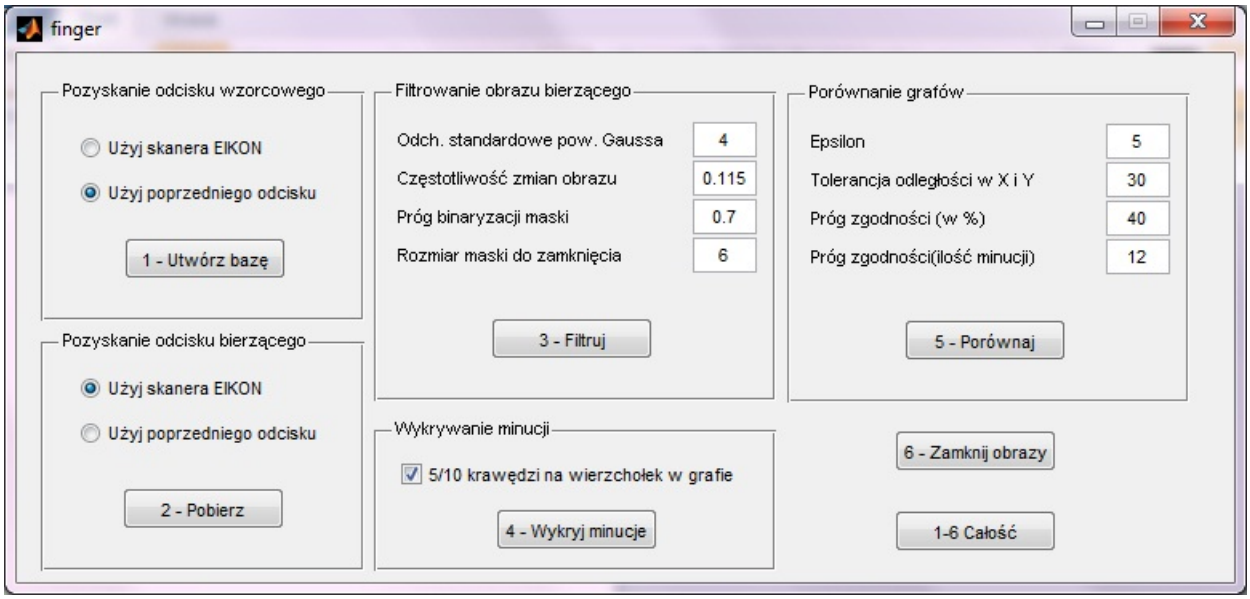
Oprogramowanie składa się z trzech modułów. Każdy moduł jest odrębną aplikacją napisaną w języku C++ lub w środowisku MATLAB. Aplikacje wywołują się w tle, zatem użytkownik nie odczuwa podziału programu na moduły. Szczegółowy spis plików źródłowych, wymaganych bibliotek dynamicznych i innych zostały opisane w dodatkach.

5.2. Aplikacja upek.exe

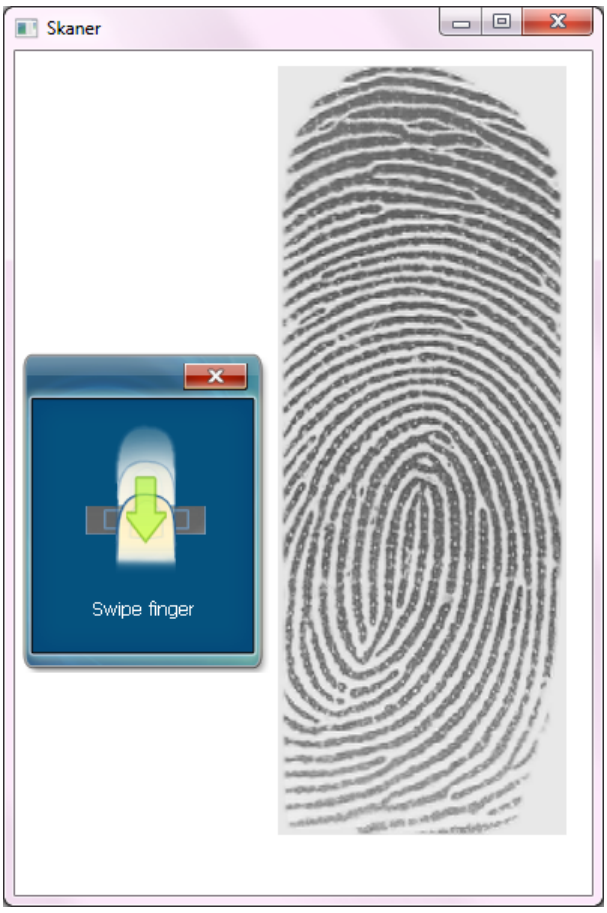
5.2.1. Opis

Aplikacja ma za zadanie komunikację z czytnikiem linii papilarnych UPEK EIKON, pozyskanie obrazu linii papilarnych, zapisanie powstałego obrazu na dysku, oraz wygenerowanie mapy orientacji grzbietów na obrazie linii papilarnych. Przed pierwszym uruchomieniem konieczna jest instalacja sterowników, która jest opisana w dodatku A.

Program został napisany w języku C++ z użyciem bibliotek *dll* udostępnionych przez producenta czytnika [11], a także pakietu bibliotek do obróbki obrazów OpenCV 2.2.0 [20].



Rysunek 5.1: Okno głównego programu z domyślnymi ustawieniami



Rysunek 5.2: Okno programu upek.exe, w czasie powtórnego pobierania śladu linii papilarnych.

Program można podzielić na dwa funkcjonalne bloki. Pierwszy z nich odpowiada za wczytanie linii papilarnych. Na początek sprawdzane jest czy do komputera podłączony jest czytnik linii papilarnych EIKON. W przypadku jego braku, użytkownik zostaje poproszony o jego podłączenie. Jeżeli czytnik jest prawidłowo podłączony (tzn. pogram nawiązał z nim komunikację), pojawia się nowe okno programu, które zostało przedstawione na rysunku 5.2. Następnie program wyświetla grafikę pokazującą kierunek przesuwania palca po czytniku wraz z komunikatem o konieczności dokonania zeskanowania palca. W razie problemów z odczytaniem danych z czytnika (np. przesunięcie palca zbyt szybko lub w złym kierunku), zostanie wyświetlony odpowiedni komunikat z grafiką informującą użytkownika, że nieprawidłowo korzysta z czytnika. Jeśli obraz zostanie pobrany prawidłowo, aplikacja wyświetla pobrany obraz oraz prośbę o zaakceptowanie przez użytkownika jakości pobranego obrazu. Można w tym momencie wybrać opcję TAK, a w efekcie zakończyć komunikację z czytnikiem linii papilarnych, lub wybrać opcję NIE i ponownie rozpocząć skanowanie opuszka palca.

Opisana część programu to sekwencja wywołań funkcji z bibliotek *dll* zakończona zapisaniem pobranego pliku na dysku. Funkcje wywoływane przez program odpowiadają za:

– nawiązanie i podtrzymanie komunikacji z czytnikiem,

– aktywację i dezaktywację sensora w czytniku,

– odbiór danych z czytnika,

– utworzenie z otrzymanych danych obrazu w pamięci programu,

– informowanie użytkownika o stanie i etapie skanowania przez wyświetlanie komunikatu popartego grafiką.

Jedynymi funkcjami w tej części programu, które nie są poparte bibliotekami dynamicznymi to zapis obrazu do pliku oraz wyświetlenie obrazu w oknie.

Drugi funkcjonalny blok to tworzenie mapy orientacji grzbietów będących na obrazie. Program Petera Kovesi[13] został przepisany do języka C++. Umożliwiła to biblioteka OpenCv ([21]), która zawiera funkcje odpowiedzialne za operacje na obrazach. Wszystkie obliczenia (tworzenie filtrów, gradientów, mnożenie czynników macierzy i inne) są wykonywane na obiektach *Mat* zaimplementowanych we wspomnianej bibliotece. Obiekt taki posiada przydzieloną pamięć na dane obrazu, oraz zawiera wiele pomocniczych pól określających parametry obrazu, np. rozmiar obrazu, ilość kanałów, wielkość palety kolorów, itp. Ważnym ułatwieniem podczas obliczeń jest obsługa przez klasę *Mat* danych w różnych formatach. Jak wiadomo wartości pikseli w obrazie mają wartości całkowite (najczęściej zakres 0-255), jednak podczas różnego rodzaju filtracji, w fazach pośrednich dane te mogą przyjmować wartości ułamkowe a nawet ujemne.

Po wykonaniu wszystkich obliczeń i wygenerowaniu przez program mapy orientacji, jest ona zapisywana na dysku, po czym program zamyka komunikację z czytnikiem linii papilarnych i kończy działanie.

5.2.2. Możliwości zmian

Istnieje możliwość adaptacji aplikacji do pracy jako samodzielny program służący do zapisu odcisków palców do pliku. Należałoby wówczas dodać możliwość samodzielnego wyboru przez użytkownika nazwy i lokalizacji do zapisu pliku z zeskanowanym obrazem. Powyższa funkcjonalność nie była wymagana w bieżącej konfiguracji oprogramowania, dlatego zaniechano jej implementacji.

5.3. Aplikacja porownanie.exe

5.3.1. Opis

Aplikacja jest napisana w języku C++. Program nie posiada trybu okienkowego, i jest wywoływany z linii poleceń. Jego zadaniem jest porównanie dwóch grafów pod kątem izomorfizmu podgrafów. Opiera się na klasie *graf* napisanej specjalnie na potrzeby tej aplikacji. Konstruktor klasy *graf* wczytuje z pliku dane na temat grafu, i tworzy obiekt zawierający kopie tych danych.

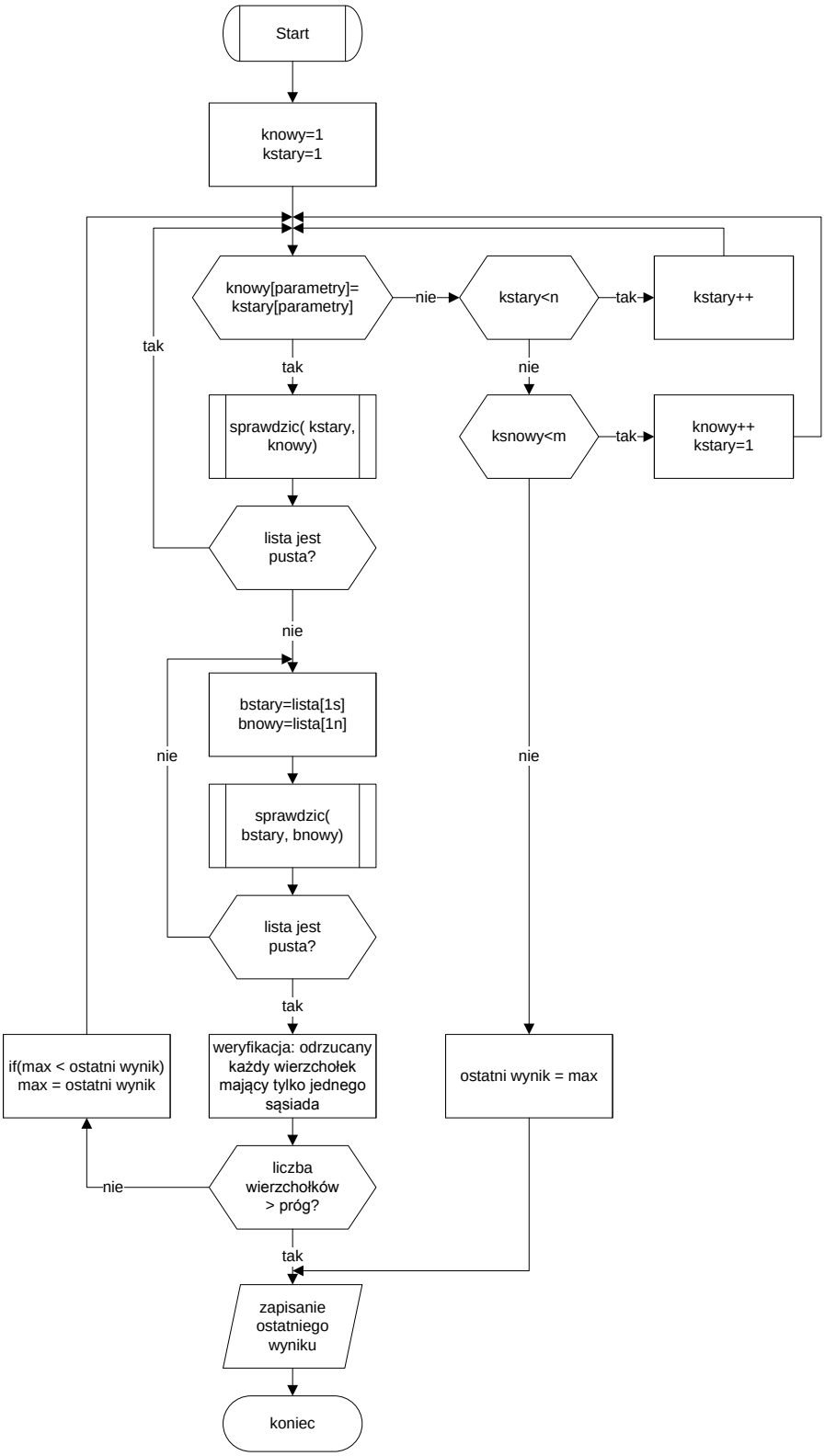
W głównej funkcji programu tworzone są dwa obiekty opisujące wskazane przez użytkownika grafy: graf o nazwie *nowy*, czyli graf opisujący świeżo wczytany z czytnika odcisk, oraz graf *bazowy* do którego będzie porównywany graf *nowy*. Wskazanie plików z opisem grafów odbywa się przez parametry w linii poleceń. Opcjonalnym parametrem przekazywanym również z linii poleceń jest plik ze współczynnikami używanymi przez funkcje porównujące. Jeśli ścieżka trzeciego pliku nie zostanie podana, to program użyje wartości domyślnych dla tych współczynników.

Do opisu zależności między wierzchołkami została przeznaczona jedna macierz (nazwana *M*). Ma ona wymiary $n \times m$, gdzie *n* to liczba wierzchołków *nowego* grafu, a *m* to liczba wierzchołków grafu *bazowego*. Jedynka (oznaczona flagą *zgodne wierzchołki*) na przecięciu *i*-tego wiersza i *j*-tej kolumny oznacza, że *i*-ty wierzchołek *nowego* grafu odpowiada *j*-temu wierzchołkowi grafu *bazowego*. W każdym z wierszy i kolumn może znaleźć się tylko jedna flaga oznaczająca zgodność. Zatem z macierzy *M* można odczytać, które wierzchołki z obu grafów należą do podgrafów, które są izomorficzne względem siebie.

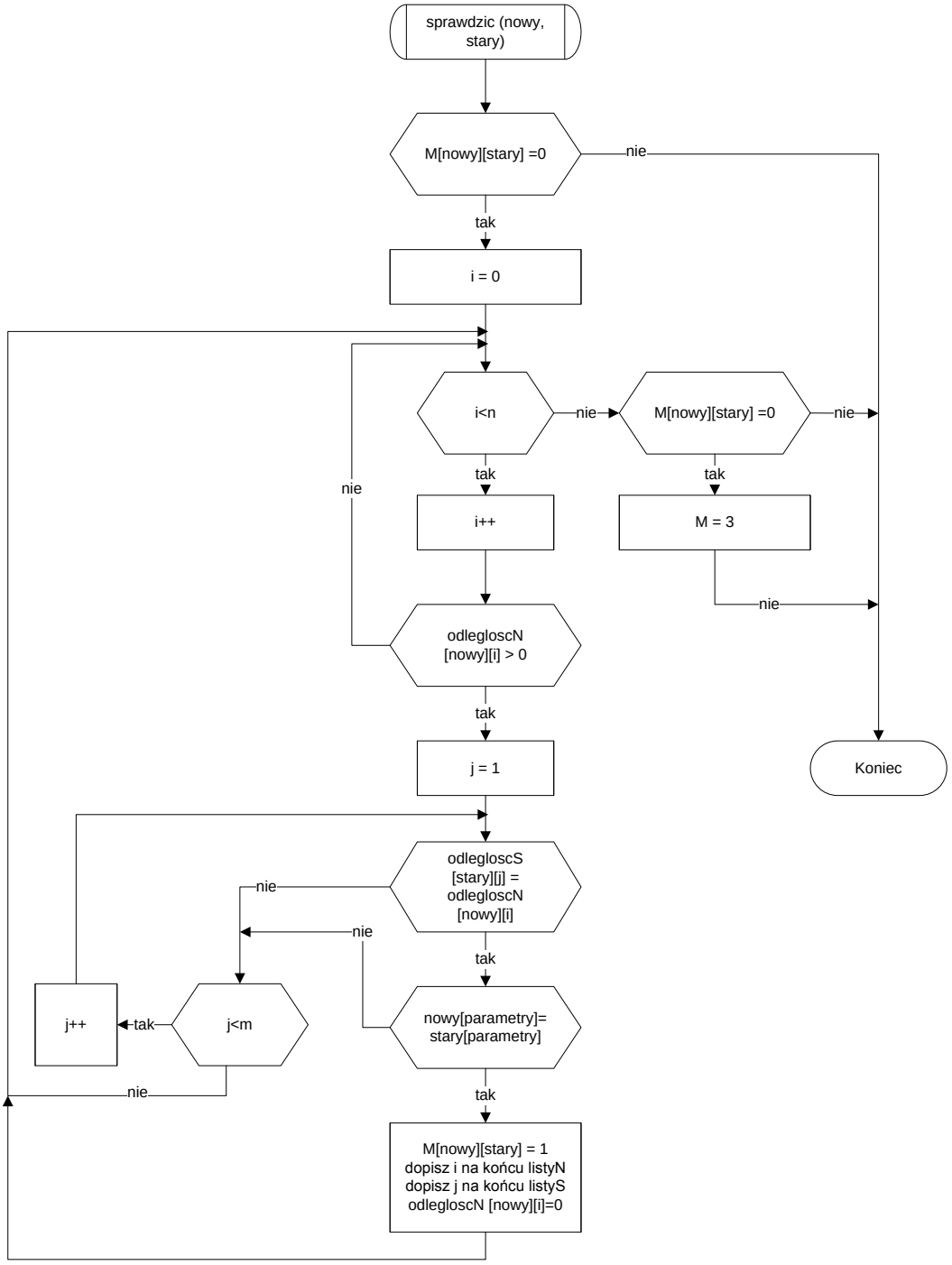
Do opisu krawędzi należących do tych podgrafów użyto również jednej macierzy (o nazwie *kr*). Macierz przyjmuje rozmiar macierzy odległości *nowego* grafu wczytywanej przez program na początku. Jedynka (oznaczona flagą *zgodna krawędź*) na przecięciu *i*-tego wiersza i *j*-tej kolumny oznacza, że wierzchołki *i*-ty oraz *j*-ty, oba należące do podgrafu *nowego* grafu są połączone krawędzią. Zgodnie z logiką długość krawędzi w podgrafie jest taka sama jak długość tej krawędzi w *nowym* grafie. Krawędzie należące do podgrafu grafu *starego* nie są zapamiętywane w oddzielnej macierzy. Ponieważ obydwa podgrafy o których mowa są dla siebie nawzajem izomorficznymi odpowiednikami, to krawędzie drugiego z podgrafów można łatwo uzyskać z macierzy *M* oraz *kr*.

Algorytm składa się z dwóch funkcji: *porownaj()* oraz *sprawdzic()*. Algorytmy obu funkcji są zamieszczone na rysunkach 5.3, 5.4. Na rysunku 5.3 przyjęto następujące nowe oznaczenia:

– *knowy*, *kstary* - numery wierzchołków z grafów odpowiednio *nowego* i *bazowego*, które na danym etapie przebiegu programu są kandydatami na odpowiadające sobie wierzchołki;



Rysunek 5.3: Diagram blokowy przedstawiający algorytm funkcji porownaj()



Rysunek 5.4: Diagram blokowy przedstawiający algorytm funkcji sprawdzic()

– *bnowy*, *bstary* - numery wierzchołków z grafów odpowiednio *nowego* i *bazowego*, które na danym etapie przebiegu programu są uznane za odpowiadające sobie wierzchołki i wyszukuje się wokół nich podobnych sąsiadów;

– *parametry* - zbiór wartości opisujących minucję odpowiadającą danemu wierzchołkowi (jest to tylko skrót w zapisie, gdyż macierze te są składnikami klasy *graf*);

– *lista* - lista wierzchołków, które są oznaczone jako zgodni sąsiedzi i należy sprawdzić ich przynależność do podgrafów w następnej kolejności;

– *ostatni wynik* - określenie zestawu danych, w skład których wchodzą macierze *M*, *kr*, liczba znalezionych minucji i procentowy współczynnik zgodności;

– *max* - zestaw danych o składzie takim samym jak ostatni wynik, ale zawierający dane o najlepszym znalezionym wyniku w danym etapie programu.

Na rysunku 5.4 przyjęto następujące nowe oznaczenia:

– *nowy*, *stary* - numery wierzchołków z grafów odpowiednio *nowego* i *bazowego*, które na danym etapie przebiegu programu są poddawane sprawdzeniu zgodności i poszukiwaniu podobnych sąsiadów;

– *odlegloscN*, *odlegloscS* - macierze odległości odpowiednio *nowego* i *bazowego* grafu wczytane przez program na początku (jest to tylko skrót w zapisie, gdyż macierze te są składnikami klasy *graf*).

Działanie funkcji *sprawdzic* polega na przeszukaniu sąsiadów obu zaproponowanych wierzchołków (kandydata z *nowego* grafu oraz kandydata z *bazowego* grafu). Jeżeli oba wierzchołki posiadają co najmniej jednego sąsiada w tej samej odległości (z dokładnością do *epsilon* pikseli, gdzie *epsilon* to parametr) i sąsiedzi ci mają takie same cechy charakterystyczne, to zgodność wierzchołków-kandydatów zostaje potwierdzona i kandydaci zostają zgodnymi wierzchołkami. Początkowe testy aplikacji pokazały, ze ilość dopasowanych wierzchołków z jednym sąsiadem (i zwykle były to błędne dopasowania) była większa niż ilość pozostałych wierzchołków. Dlatego podczas badania odległości sąsiada od bieżącego kandydata dodano kontrolę kierunku (przy pomocy parametru *tolerancja*), tzn czy znaleziony na obu grafach wspólny sąsiad jest na obu grafach w tym samym kierunku względem bieżącego punktu. Ponieważ grafy generowane są z punktów na obrazie, to można wykorzystać współrzędne tych punktów na obrazie. Jeśli różnica współrzędnych między wierzchołkiem i sąsiadem (w X lub w Y) jest zbyt duża to sąsiad nie jest akceptowany, nawet jeśli krawędź ma zgodną długość. Często zdarzało się bowiem, że wierzchołek kandydat miał na *nowym* grafie sąsiada powyżej, a na *bazowym* grafie poniżej (lub podobnie na jednym grafie na lewo, a na drugim na prawo), co powodowało błędne przyporządkowania. Jeżeli jednak przyporządkowanie jest zgodne, zostaje to odnotowane w macierzy *M*. Ważnym elementem jest przestrzeganie struktury macierzy *M*, więc zdarza się, że pozornie zgodnym sąsiadem jest wierzchołek wcześniej już uznany jako zgodny z innym wierzchołkiem, i wówczas takie porównanie nie zostaje zapisane na liście do sprawdzenia, a w macierzy *M* oraz macierzy opisującej krawędzie przyporządkowanie

zostaje uznane za nieprawidłowe. Każda krawędź, która została sprawdzona podczas poszukiwania sąsiadów zostaje odpowiednio oznaczona: albo jako niezgodna, albo jako zgodna (ten fakt pominięto na rysunku, gdyż sprawdzanie lub modyfikowanie macierzy następuje bardzo często podczas pracy programu, i utrudniłoby zrozumienie działania algorytmu).

Idea funkcji *porownaj* opiera się na przeszukiwaniu wszerz dwóch grafów jednocześnie. Na początku z *nowego* grafu wybierany jest dowolny wierzchołek. Następnie przeszukiwane są wszystkie wierzchołki *bazowego* grafu, w celu sprawdzenia, czy którykolwiek z nich ma takie same parametry (typ i orientacja minucji odpowiadającej danemu wierzchołkowi) jak wybrany wierzchołek z *nowego* grafu. Jeśli żaden z wierzchołków nie pasuje, to z *nowego* grafu wybierany jest kolejny wierzchołek i sprawdzanie odbywa się do skutku. W przypadku przeszukania całego grafu i braku dopasowania odpowiedniego wierzchołka program kończy działanie ogłaszając brak zgodności między grafami. Jeżeli jednak zostanie znaleziony wierzchołek (lub kilka wierzchołków) ze zgodnymi parametrami, to zostają one zapisane na pomocniczej liście potencjalnych kandydatów na wierzchołek powtarzający się na obu grafach (ta lista służy późniejszemu zapamiętaniu, które wierzchołki były już sprawdzane a które nie i jest ona pominięta na rysunku). Wierzchołki z tej tymczasowej listy są kolejno weryfikowane przez wywołanie funkcji *sprawdzic*, i jeśli żaden z nich nie okaże się zgodny, a cała lista kandydatów zostanie sprawdzona to program powróci do samego początku, wybierając kolejny wierzchołek z *nowego* grafu. Jeśli potwierdzi się zgodność któregoś z wierzchołków, to program przejdzie do kolejnego etapu by poszukać kolejnych zgodnych wierzchołków.

Lista kandydatów nie będzie teraz dalej sprawdzana, ale jest zachowana w pamięci dopóki nie zostanie zakończone porównanie z zaakceptowanymi wierzchołkami (w dalszej części zostanie wyjaśnione dlaczego tak się dzieje). Zgodni sąsiedzi zostają natomiast wpisani na nową listę: listę wierzchołków czekających na sprawdzenie (analogicznie jak w algorytmie przeszukiwania wszerz, który był inspiracją tego algorytmu). Należy nadmienić, że wierzchołki są dodawane na listę parami (po jednym z każdego grafu), więc są przechowywane na jednej liście, najpierw wierzchołek z *nowego* grafu, potem z *bazowego*. Mając pierwsze zaakceptowane wierzchołki, za kandydatów przyjmuje się pierwsze wierzchołki dodane wcześniej do listy do sprawdzenia (po jednym z obu grafów, zdejmowane w odpowiedniej kolejności). Podobnie jak wcześniej wykorzystywana jest funkcja *sprawdzic*, do poszukiwania sąsiadów w podobnej odległości i o tych samych cechach. W przypadku powodzenia zostają dopisani na koniec listy do sprawdzenia, a sprawdzana para wierzchołków uznana za zgodną.

Gdy lista wierzchołków do sprawdzenia się wyczerpie, macierz wierzchołków M poddana zostaje ostatniej kontroli: każdy wierzchołek powstałego podgrafu, który ma tylko jednego sąsiada jest mało wiarygodny, gdyż jedyną informacją o jego odległości od tego sąsiada będzie fakt iż znajduje się on na okręgu o promieniu równym długości krawędzi. Dlatego wierzchołki takie są usuwane, a wierzchołki posiadające co najmniej dwóch sąsiadów pozostają niezmienione, gdyż minimum dwóch sąsiadów jednoznacznie określa ich położenie.

Po tej weryfikacji sprawdzana jest liczebność zgodnego podgrafu. W tym momencie program zakończy działanie jeżeli spełniony jest jeden z dwóch warunków:

– liczba zgodnych wierzchołków jest większa niż wymagana przez użytkownika wartość minimalna (domyślnie 12)

– stosunek liczby zgodnych wierzchołków do wszystkich wierzchołków *nowego* grafu jest większy niż podany przez użytkownika próg (domyślnie 40%)

Jeżeli żaden z powyższych warunków nie jest spełniony, to zapamiętywana jest ilość znalezionych zgodnych wierzchołków (jako maksymalna znaleziona ilość wierzchołków dla danych grafów), i program wraca do poprzedniej pętli. W tym momencie staje się jasne, dlaczego lista kandydatów nie została wcześniej usunięta. Jeśli na tej liście znajdują się kandydaci, to program kontynuuje działanie od tego momentu (czyszcząc wszystkie później dokonane zmiany w macierzy M i macierzy kr), a jeżeli lista jest pusta, to program wraca do samego początku, poszukując kandydatów do kolejnego niesprawdzonego jeszcze w pierwszej iteracji wierzchołka z *nowego* grafu.

W kolejnych iteracjach, gdy uda się znaleźć zgodny podgraf ponownie jest badana liczebność wierzchołków. Jeżeli zostaną sprawdzone wszystkie możliwe dopasowania, a mimo to liczba zgodnych wierzchołków jest mniejsza niż wymagany próg, to grafy zostają uznane za niezgodne. Jeżeli jednak zostało znalezione jakieś dopasowanie (nawet niespełniające powyższych progów) to program zwróci właśnie to maksymalne dopasowanie.

Danymi wyjściowymi programu są zapisane do pliku: macierz krawędzi, macierz wierzchołków M, liczba zgodnych wierzchołków oraz stosunek zgodnych wierzchołków do wszystkich wierzchołków *nowego* grafu.

5.3.2. Możliwości zmian

Aplikacja przeszła bardzo wiele przemian zanim zaczęła działać w aktualnej postaci. Funkcją, o którą można by ją jeszcze rozbudować jest poszerzenie wyszukiwania o niespójne części grafu. Aplikacja przeszukuje graf począwszy od jednego wierzchołka, sprawdzając wszystkie sąsiednie, które zgadzają się z grafem *bazowym*. Może się zdarzyć, że pewne części grafu zostaną niesprawdzone, gdyż nie są spójne ze sprawdzoną częścią grafu. Dodanie algorytmu szukającego kilku podgrafów zgodnych zamiast poszukiwań jednego podgrafu zgodnego może poprawić współczynnik zgodności grafów. Należy jednak zaznaczyć, że program w aktualnej postaci umożliwia skuteczne potwierdzenie zgodności (bądź niezgodności) grafów jeżeli grafy wczytywane przez program są spójne.

5.4. Aplikacja finger.fig

5.4.1. Wstęp

Finger.fig jest najbardziej rozbudowanym modułem oprogramowania, całość napisana w środowisku MATLAB. Główną aplikacją zarządzającą całym projektem jest okno graficzne finger.fig utworzone w GUI(ang. Graphical User Interface) przy pomocy narzędzia GUIDE(ang. MATLAB GUI Development Environment). Okno jak wiadomo z rozdziału 5.1 jest podzielone jest na ponumerowane sekcje. Każda sekcja realizuje odrębny algorytm napisany w postaci funkcji programu MATLAB. Działanie tych al-

gorytmów zostało opisane w rozdziałach 2, 3 oraz 4. Ich implementacja zostanie opisana w kolejnych podrozdziałach. Dla porządku opisy są w takiej samej kolejności w jakiej wykonywany jest program.

5.4.2. Utworzenie modelu wzorcowego

Ta część programu odpowiedzialna jest za stworzenie grafu bazowego do przyszłych porównań. Część ta nie zawiera własnego algorytmu. Bazuje na funkcjach opisanych w dalszej części pracy, wykorzystując te same parametry, których użytkownik używa do eksperymentów na świeżym odcisku. Takie podejście gwarantuje, że oba ślady linii papilarnych (zarówno bazowy jak i drugi porównywany) będą filtrowane w taki sam sposób.

Jako bazowy można użyć nowy ślad linii papilarnych pobrany z czytnika EIKON, lub użyć poprzedniego odcisku bazowego (o ile taki jest dostępny). Wyboru dokonuje się zaznaczając kontrolkę radiobutton przy wybranej przez użytkownika opcji. Jeżeli dostępny jest poprzednio używany plik z odciskiem linii papilarnych, to domyślną opcją jest użycie właśnie poprzedniego odcisku (program zapisuje ostatni skan w postaci pliku graficznego w swoim katalogu, więc pamięta ostatni skan nawet po wyłączeniu programu i ponownym uruchomieniu, oczywiście o ile plik nie zostanie usunięty przez użytkownika poza programem). Jeżeli jednak plik nie jest obecny w odpowiednim katalogu to opcja ta jest nieaktywna, a domyślnie ustawiona jest opcja użycia czytnika EIKON.

Funkcja wywołuje więc z odpowiednio dobranymi parametrami (w nawiasach podano jakie parametry się różnią) algorytmy z rozdziałów 5.4.3 (domyślne wczytywanie obrazu z dysku o ile jest dostępny), 5.4.4 oraz 5.4.5 (parametr dotyczący liczebności grafu ustawiony na graf pełny). Poza tym program nie wyświetla okien z pośrednimi wynikami wczytania pliku i filtracji, a jedynie końcowy efekt po ekstrakcji minucji. Nazewnictwo wszystkich macierzy zawierających pośrednie wyniki dla śladu bazowego jest zachowane, z dodaniem przyrostka w postaci cyfry 1.

5.4.3. Pozyskanie odcisku bieżącego

Zadaniem tej części oprogramowania jest pobranie tzw. *surowego* śladu linii papilarnych (rysunek 2.1 a) z czytnika, lub z pliku jeżeli użytkownik chce odtworzyć ostatnie porównanie. Domyślną opcją jest pobranie śladu z czytnika. Wybór źródła obrazu można dokonać przy pomocy kontrolki radiobutton. Podobnie jak w poprzedniej sekcji, ostatni skan jest pamiętany jako plik graficzny na dysku komputera. Jeżeli plik z zapisem poprzedniego skanu nie istnieje, to opcja ta jest niedostępna i użytkownik jest zmuszony wykorzystać czytnik. Należy zaznaczyć, iż ostatnio używany plik bazowy nie jest tożsamy ostatnio używanemu plikowi porównywanemu. Zatem można w obu sekcjach zaznaczyć opcję wczytywania danych z plików, a wczytywane dane nie będą tym samym śladem.

Jeżeli użytkownik wybierze opcję wczytania pliku z czytnika, to program wywoła aplikację opisaną w podrozdziale 5.2, a następnie wczytuje przygotowane przez ww. aplikację pliki. Wybór ostatniego skanu spowoduje tylko wczytanie plików. Po ich wczytaniu, widok śladu linii papilarnej zostaje wyświetlony jako nowy rysunek.

5.4.4. Filtrowanie obrazu bieżącego

Na początku obraz poddawany jest normalizacji wykorzystywane są do tego dwie funkcje pomocnicze, które zmniejszają kontrast oraz zmieniają wartości na zmiennoprzecinkowe o zakresie $0 \div 1$. Tak przygotowane dane lepiej nadają się do filtracji. Następnie tworzona jest grupa masek filtru Gabora. Odbywa się to przez wielokrotne wywołanie funkcji zawierającej implementację wzoru na filtr Gabora[1]. Funkcja ta za każdym razem jest wywoływana z innym parametrem θ, który określa kierunek filtru. Wartości tego parametru oraz kolejność ich występowania muszą być dokładnie takie same, jak podczas generowania mapy orientacji, dlatego przypisane są na stałe, bez możliwości ingerencji w nie użytkownika. Filtracji dokonuje się wywołując kolejno funkcję *filter2*[2] jeden raz z każdą przygotowaną wcześniej maską. Otrzymuje się wówczas 12 przefiltrowanych obrazów. Złożenie tych obrazów polega na sprawdzeniu na mapie orientacji wartości każdego piksela. Następnie kopiuje się na obraz docelowy piksele z tego przefiltrowanego obrazu, który ma taki sam numer jak wartość sprawdzonego piksela z mapy orientacji, kończąc tym samym pierwszy etap filtracji.

Drugi etap polega na obcięciu pewnych obszarów obrazu. Obcięcie jest niezbędne, gdyż podczas tworzenia mapy orientacji w miejscach, gdzie nie było danych na obrazie (czyli w przypadku, gdy odcisk nie zajmuje całego obrazu) dane o orientacji takiego pustego obszaru są przypadkowe, powodując fałszywe powiększenie obszaru obrazu. Wymnożenie (iloczyn logiczny) każdego piksela obrazu przez odpowiadający mu piksel dobranej dla danego obrazu maski ograniczającej pozwala się przed tym skutecznie zabezpieczyć sprowadzając kontur obrazu wynikowego do konturu obrazu źródłowego .

Aby utworzyć taką maskę binaryzuje się niefiltrowany plik wejściowy, po czym dokonuje się morfologicznego zamknięcia obrazu (wykorzystując funkcję *bwmorph* z programu MATLAB z odpowiednimi parametrami). Próg binaryzacji, jak również rozmiar elementu strukturalnego do zamknięcia są parametrami, możliwymi do zmiany przez użytkownika. W efekcie zamknięcia otrzymywana jest binarna maska, posiadająca wartości 1 w obszarze który powinien pozostać w obrazie, oraz wartości 0 w obszarach wymagających odrzucenia z przefiltrowanego obrazu. Niejako przy okazji tworzy się drugą maskę, która ogranicza obszar bardziej niż poprzednia i zostanie wykorzystana dopiero przy wykrywaniu minucji. Powstaje ona przez wykonanie morfologicznej erozji na masce otrzymanej wcześniej.

5.4.5. Wykrywanie minucji

W celu wyszukiwania przy pomocy jednego algorytmu minucji dwóch typów: rozgałęzień i zakończeń, dokonuje się odwrócenia kolorów. Obraz po filtracji jest obrazem binarnym, więc odwrócenie kolorów polega na negacji elementów macierzy reprezentującej obraz. Od tej pory oba obrazy poddawane są takim samym operacjom.

Pierwszym etapem jest szkieletyzacja, która pozwala na zmniejszenie grubości każdego łuku do grubości jednego piksela. Wykonuje się ją używając ponownie funkcji *bwmorph*, tym razem z parametrem *skel*[3]. Gdy łuki, z których budowany jest szkielet są poszarpane (może się to zdarzyć przy sklejaniu

[1] dla przypomnienia wzór jest opisany równaniem 2.8

[2] jest to funkcja programu MATLAB realizująca konwolucję

[3] od angielskiego *skeleton* - szkielet

przefiltrowanych obrazków), może się zdarzyć, że szkielet będzie posiadał dodatkowe mini gałązki o długości od 1 do kilku pikseli. Aby nie zostały one wyszukane jako minucje, należy takie krótkie gałązki usunąć. Dokonuje tego pętla, która wyszukuje wszystkie zakończenia szkieletu, skracając je o 5 pikseli. Dzięki temu, gałązki krótsze niż 5 pikseli znikają. Jednak powoduje to skrócenie pozostałych gałęzi, a przez to przemieszczenie ich zakończeń względem ich początkowej lokalizacji. Dlatego dodano drugą pętlę, która odnajduje wszystkie zakończenia obciętego szkieletu które pozostały, porównuje je ze szkieletem oryginalnym(tym przed obcinaniem) i rekonstruuje zachowane gałęzie szkieletu przywracając im pierwotny kształt i długość.

Wyszukiwanie zakończeń gałęzi polega na przeszukiwaniu obrazu ze szkieletem piksel po pikselu. Dla każdego analizowanego piksela badane jest otoczenie, czyli 8 pikseli z nim sąsiadujących. Jeżeli piksel badany jest czarny, a wokół niego czarny jest dokładnie jeden sąsiad, to oznacza że analizowany piksel jest zakończeniem gałęzi szkieletu. Podczas rekonstrukcji, dodatkowo sprawdza się, czy piksel będący w tym samym miejscu na oryginalnym szkielecie miał większą ilość czarnych sąsiadów. Jeśli tak, to są to sąsiedzi usunięci i właśnie ich należy zrekonstruować. Dzięki analizie dwóch szkieletów jednocześnie krótkie gałęzie nie zostaną zrekonstruowane, gdyż na obciętym szkielecie ich nie ma.

Po poprawkach na szkielecie, wyszukuje się jeszcze raz jego zakończenia. Odpowiadają one położeniu minucji. Na jednym szkielecie znalezione punkty reprezentują minucje typu *rozgałęzienie*, a na drugim minucje typu *zakończenie*. Kolejnym krokiem jest pominięcie punktów, które powstają na krawędziach obrazu przez obcięcie części, która się nie zmieściła. W rozdziale 5.4.4 utworzona została maska filtracyjna, która pomoże pozbyć się tych punktów. Jej użycie jest takie samo jak przy obcinaniu obrazu: środek maski, czyli obszar w którym są prawidłowe minucje jest wypełniony jedynkami, obszary krawędzi są wypełnione zerami. Wymnożenie (iloczyn logiczny) piksel po pikselu obu obrazów, spowoduje zachowanie tylko punktów wewnątrz maski.

Punkty, które pozostały są wynikiem prawidłowego przetwarzania, należy je więc zapamiętać. Do opisu minucji przygotowano macierz, posiadającą ilość kolumn odpowiadającą ilości znalezionych minucji, oraz 5 wierszy. Każda kolumna tej macierzy opisuje jedną wykrytą na obrazie minucję. Wartości opisujące minucje zostały już wymienione w rozdziale 3. Dla przypomnienia są to:

– numer minucji (minucje są numerowane kolejnymi liczbami naturalnymi)

– współrzędna X na obrazie

– współrzędna Y na obrazie

– typ minucji (cyfrą 1 oznaczono zakończenia, 2 oznaczono rozgałęzienia)

– orientacja grzbietu w miejscu wystąpienia minucji (odczytana z mapy orientacji)

Następnym krokiem jest utworzenie grafu na podstawie znalezionych minucji. Graf reprezentowany jest przez macierz odległości. Jest to macierz kwadratowa o wymiarze równym ilości minucji. Na przecięciu i-tego wiersza oraz j-tej kolumny jest zero w przypadku, gdy nie ma krawędzi pomiędzy i-tym i j-tym wierzchołkiem, lub liczba większa od zera, jeżeli krawędź taka istnieje. Wartość ta określa długość krawędzi, czyli odległość między połączonymi wierzchołkami. Ponieważ graf jest nieskierowany,

taka sama wartość zostanie wpisana na przecięciu *j*-tego wiersza oraz *i*-tej kolumny. Odległości są obliczane na podstawie współrzędnych położenia tych minucji na obrazie wejściowym (wartości te były zapisywane w macierzy jako parametry opisujące minucje). Dla pełnego opisu grafu wymagane są więc zarówno macierz opisująca wierzchołki, jak również macierz odległości.

5.4.6. Porównanie grafów

Kod napisany w tej części programu, realizuje:

- zapis do pliku opisu grafu,
- wywołanie aplikacji *porownanie.exe* realizującej porównanie,
- analizę i wyświetlenie wygenerowanych przez nią wyników.

Plik z opisem zawiera same liczby, oddzielone tabulatorami i znakami nowej linii, ma stałą strukturę:

- jedna wartość, określająca ilość minucji, a zarazem rozmiar grafu,
- macierz o 5 wierszach i ilości kolumn wymienionej powyżej opisująca cechy minucji,
- macierz kwadratowa o wymiarze wymienionym na początku pliku, opisująca długości krawędzi.

Opis aplikacji *porownanie.exe* znajduje się w rozdziale 5.3. Program ten jest wywoływany przez MATLABA z parametrami wskazanymi przez użytkownika. Po jego wykonaniu następuje odczyt pliku wygenerowanego przez program. Plik podobnie jak powyższy zawiera same liczby, oddzielone tabulatorami i znakami nowej linii, i również ma swoją stałą strukturę:

- cztery wartości: dwie oznaczają liczby wierzchołków porównywanych grafów (*m* i *n*), kolejne dwie liczbę zgodnych minucji i procent zgodności,
- kwadratową macierz *kr* o rozmiarze *n*, opisującą istniejące krawędzie w podgrafie grafu porównywanego,
- macierz *M* o rozmiarze $n \times m$, opisującą przyporządkowanie wierzchołków jednego podgrafu do wierzchołków drugiego.

Po wczytaniu tych danych prezentowane są wyniki porównania. Najpierw wyświetlany jest obraz porównywanej linii papilarnej po filtracji. Na niego nałożone są w formie małych okręgów minucje wg macierzy opisującej minucje (przy użyciu funkcji *plot*, którą udostępnia MATLAB. Następnie przy użyciu tej samej funkcji zamalowywane są te kółeczka, które odpowiadają wierzchołkom wchodzącym w skład znalezionego zgodnego podgrafu. Wierzchołki te identyfikuje się przez analizę wiersz po wierszu macierzy *M*. Wiersz posiadający jedynkę oznacza należący do podgrafu wierzchołek. W następnej kolejności nanoszone są linie oznaczające krawędzie (wykorzystana funkcja *line*, którą zawiera MATLAB). Linie te generowane są na podstawie macierzy *kr* otrzymanej z pliku, oraz macierzy opisującej minucje (wykorzystuje ona współrzędne minucji, której numer odpowiada numerowi wiersza lub kolumny z macierzy *kr*).

W niemal identyczny sposób generowany jest drugi obraz dla odcisku bazowego. Jedyna różnica występuje przy selekcji wierzchołków do zamalowania, gdyż dla odcisku bazowego należy przeszukiwać macierz *M* kolumnami a nie wierszami jak robiono to wcześniej. Otrzymane w ten sposób dwa obrazy pozwalają użytkownikowi ocenić, czy program nie popełnił błędu (jest to możliwe przy nieodpowiednim doborze parametrów - zostanie to pokazane na przykładzie w rozdziale6). Na koniec generowany jest komunikat dotyczący ilości zgodnych wierzchołków, oraz procentowy udział zgodnych wierzchołków w całym grafie (przy użyciu funkcji *msgbox*).

5.4.7. Przyciski poza sekcjami

Program posiada dwa takie przyciski. Jeden z nich oznaczony numerem 6 - *Zamknij obrazy*, odpowiada tylko i wyłącznie za zamykanie wszystkich okien z wyświetlonymi obrazami prezentującymi wyniki pośrednie. Drugi z nich wywołuje wszystkie powyżej opisane algorytmy w odpowiedniej kolejności, bez konieczności przyciskania wszystkiego po kolei. Jedyną informacją jaką generuje wówczas program jest komunikat z informacjami, czy dwa odciski są zgodne czy nie (przy zadanych przez użytkownika parametrach) oraz opis procentowej zgodności znalezionego najlepszego dopasowania Przycisk ten powstał, aby program mógł funkcjonować w trybie automatycznej weryfikacji użytkownika, bez podglądu na bieżąco kolejnych etapów porównania. Oczywiście również w tym trybie brane są pod uwagę współczynniki widoczne w oknie głównym.

5.4.8. Możliwości zmian

Aplikacja jest przystosowana typowo do współpracy z konkretnym modelem czytnika, i w tej konfiguracji pracuje prawidłowo. Gdyby zaszła potrzeba obsługi innych czytników, bądź umożliwienia użytkownikowi samodzielnego doboru plików ze skanami linii papilarnych, łatwo można taką funkcjonalność dodać.

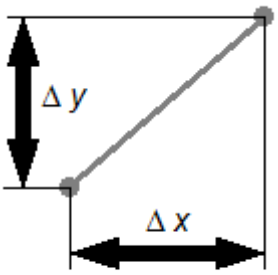
6. Testy

6.1. Zakres i sposób przeprowadzenia testów

W czasie prac nad aplikacjami zostało wykonanych wiele testów, które nie zostały udokumentowane. Na bazie tych doświadczeń, zostały wyciągnięte wstępne wnioski, pozwalające dobrać wartości domyślne wszystkich współczynników. W niniejszym rozdziale zostanie sprawdzony wpływ współczynników na wyniki porównań grafów. Testom zostanie poddany tylko algorytm porównania grafów, gdyż algorytm filtracji jest własną implementacją cudzego algorytmu ([14]), a algorytm ekstrakcji minucji nie wymaga doboru żadnych parametrów.

Najważniejszym parametrem jest *epsilon*. Jeżeli ten współczynnik ma wartość zerową, to mało prawdopodobne jest znalezienie dopasowania. Dzieje się tak dlatego, że zanim zostaną wykryte minucje, obraz jest filtrowany. Wynik filtracji zależy w dużym stopniu od jakości wczytywanego obrazu. Jak wspomniano w rozdziale 1.7 obraz otrzymywany z czytnika w dużym stopniu zależy od siły docisku palca do czytnika, a nawet wilgotności naskórka. Jest mało prawdopodobne, by dwukrotnie zeskanowany ślad linii papilarnych po filtracji i ekstrakcji minucji zwrócił takie same rezultaty co do jednego piksela. Jednak współczynnik ten nie powinien przyjmować zbyt dużej wartości, z uwagi na możliwe błędne przyporządkowanie. Domyślnie założono współczynnikowi nadać wartość 5 z możliwością jego zmniejszenia.

Drugim współczynnikiem mającym istotny wpływ na porównania jest tolerancja współrzędnych. Określa ona maksymalną dopuszczalną rozbieżność między konkretnymi współrzędnymi krawędzi na grafie bazowym i grafie porównywanym zgodnie z równaniami 6.1 oraz 6.2, w których indeks *B* oznacza



Rysunek 6.1: Sposób liczenia różnic współrzędnych X i Y dla każdej krawędzi

współrzędną grafu bazowego, natomiast *P* indeks grafu porównywanego. Tolerancja musi uwzględniać fakt, że wczytywane obrazy niekoniecznie są osiowe. Istnieje możliwość przesunięcia palca po czytniku po minimalnym skosie, co powoduje zmiany współczynników Δ nie wpływając na długość krawędzi.

$$\Delta_{x_B} - \Delta_{x_P} < tolerancja \tag{6.1}$$

$$\Delta_{y_B} - \Delta_{y_P} < tolerancja \tag{6.2}$$

Graniczna ilość minucji potrzebna do potwierdzenia zgodności domyślnie przyjmuje wartość 12, gdyż tyle zgodnych minucji gwarantuje potwierdzenie tożsamości. Jednak na czas testów wskaźnik został zwiększony do 30, aby wyszukiwać możliwie najlepszego rozwiązania. Próg procentowej zgodności został utworzony na wypadek małej liczby całkowitej minucji znalezionych na obrazie. Domyślna wartość została obliczona na podstawie testów przeprowadzanych w trakcie prac nad aplikacją. Wówczas została przeanalizowana średnia i minimalna ilość znalezionych na obrazie minucji. Wartość średnia oscylowała ok 45, minimalna ilość 31. Zatem za wartość domyślną przyjęto wyrażony w procentach stosunek granicznej ilości potwierdzonych minucji do minimalnej ilości znalezionych podczas wstępnych testów minucji, czyli jak opisuje równanie 6.3.

$$próg\,\% = \frac{próg\,minucji}{min.\,ilość\,minucji} = \frac{12}{31} \approx 40\% \tag{6.3}$$

Również ten współczynnik został na czas testów zwiększony do 60%.

6.2. Przykłady

Rysunek 4.1 przedstawia wynik porównania przykładowych grafów z domyślnymi współczynnikami. Otrzymano wówczas 28 zgodnych minucji dających zgodność 65%. Porównując te same grafy ze zmienionymi parametrami otrzymano wyniki przedstawione na rysunku 6.2. Wyniki w zestawieniu z wartością współczynników podane są w tabeli 6.1. Współczynnik *tolerancja* o wartości >30 nie dawał żadnych zmian rezultatu, i właśnie dla tej wartości otrzymano maksymalne dopasowanie.

epsilon	*tolerancja*	minucje	zgodność %
5	$\geq$30	28	65
1	30	27	63
1	20	17	40
5	20	22	51
5	10	11	25

Tablica 6.1: Wyniki porównania tych samych grafów dla różnych wartości parametrów

Rysunek 6.3 przedstawia kolejna parę grafów porównaną przy użyciu różnych współczynników. Wyniki w zestawieniu z wartością współczynników podane są w tabeli 6.2. Jak widać najlepszy wynik został zwrócony przy *tolerancji* zwiększonej do 40, jednak różnica nie jest wielka. Przy zmniejszeniu

tego współczynnika do 20 otrzymujemy bardzo słaby wynik (dlatego nie pokazano go na rysunku). Bardzo dobry wynik otrzymano po zmniejszeniu współczynnika *epsilon*. Jednak porównując rysunki można dostrzec, że niektóre krawędzie zgodne przy *epsilon* = 5 po zmianie parametru na 2 lub 1 zostają pominięte. Potwierdza to założenia o wpływie filtracji na dokładność lokalizacji minucji na dwóch obrazach.

epsilon	*tolerancja*	minucje	zgodność %
5	30	19	39
5	40	20	41
2	30	19	39
1	30	16	33
5	20	7	14

Tablica 6.2: Wyniki porównania tych samych grafów dla różnych wartości parametrów

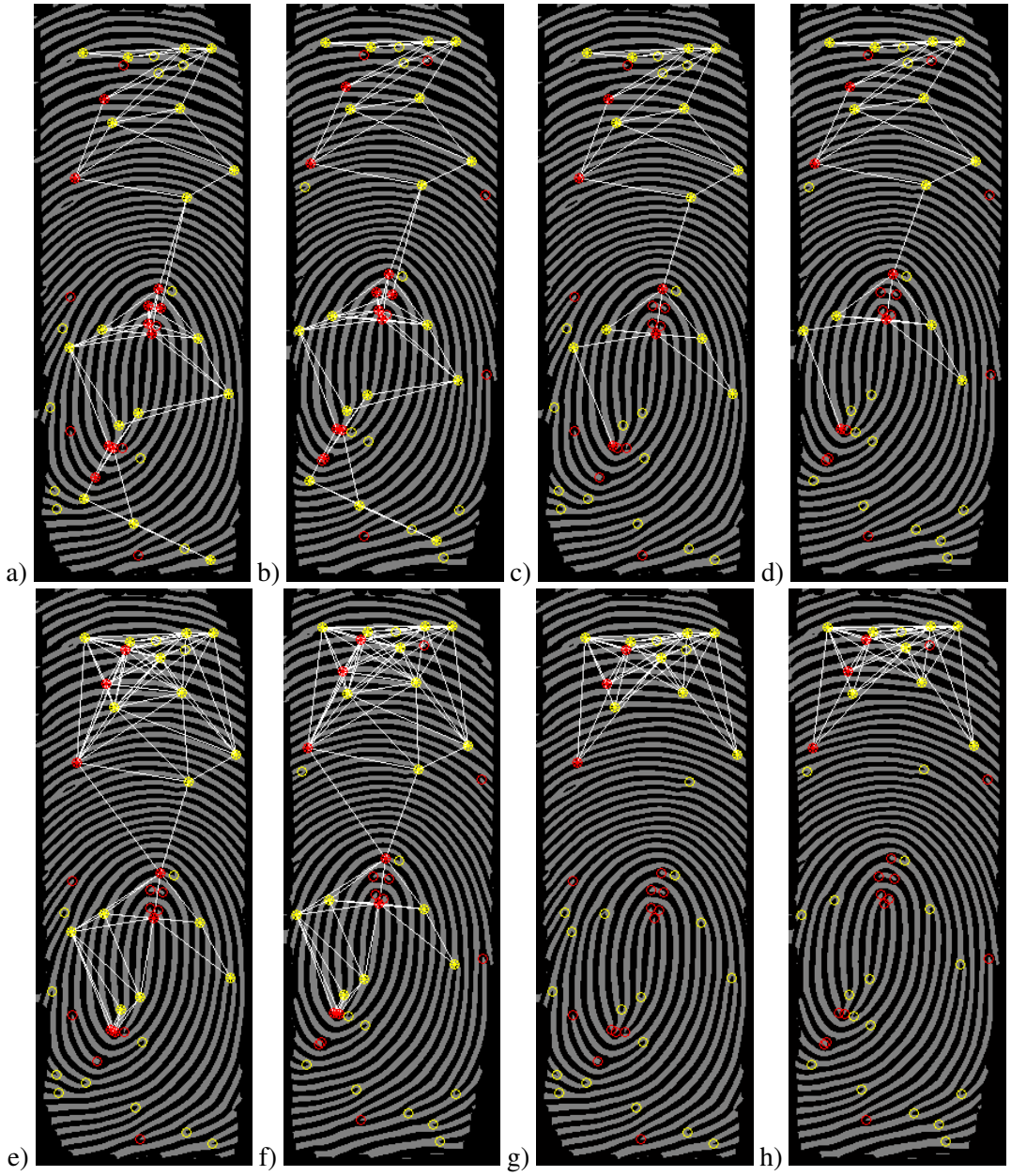
Kolejny przykład został przedstawiony na rysunku 6.4. Wyniki w zestawieniu z wartością współczynników podane są w tabeli 6.3. Wyniki tego eksperymentu są o tyle ciekawe, że najlepszy wynik otrzymano dla błędnego dopasowania (przypadek gdy *epsilon* = 5, *tolerancja* = 40, błędne dopasowanie zaznaczono kolorem). Płynie stąd wniosek, że ustawianie tak dużej tolerancji może wprawdzie wpłynąć na lepszy wskaźnik dopasowania, ale nie będzie to prawidłowy wynik ze względu na nieprawidłowo dopasowane minucje. Jeszcze ciekawiej wygląda porównanie, gdy zmniejszymy nieco *epsilon* a nie *tolerancję*. Okaże się wówczas, że otrzymamy najlepszy wynik, który nie zawiera przy tym błędnie dopasowanych minucji. Warto jednak zaznaczyć, że wartości domyślne dają wynik wystarczający do potwierdzenia zgodności obrazów i to ze sporym zapasem. Mało tego, zmniejszenie *tolerancji* do 20 dało wystarczającą zgodność. Identyczny wynik dało porównanie ze zmniejszonym *epsilon* i domyślną *tolerancją* (nie zamieszczono rysunku, gdyż był identyczny z poprzednimi).

epsilon	*tolerancja*	minucje	zgodność %
5	30	18	35
5	40	22 (fałszywe)	42
2	40	21	41
5	20	15	29
1	30	15	29

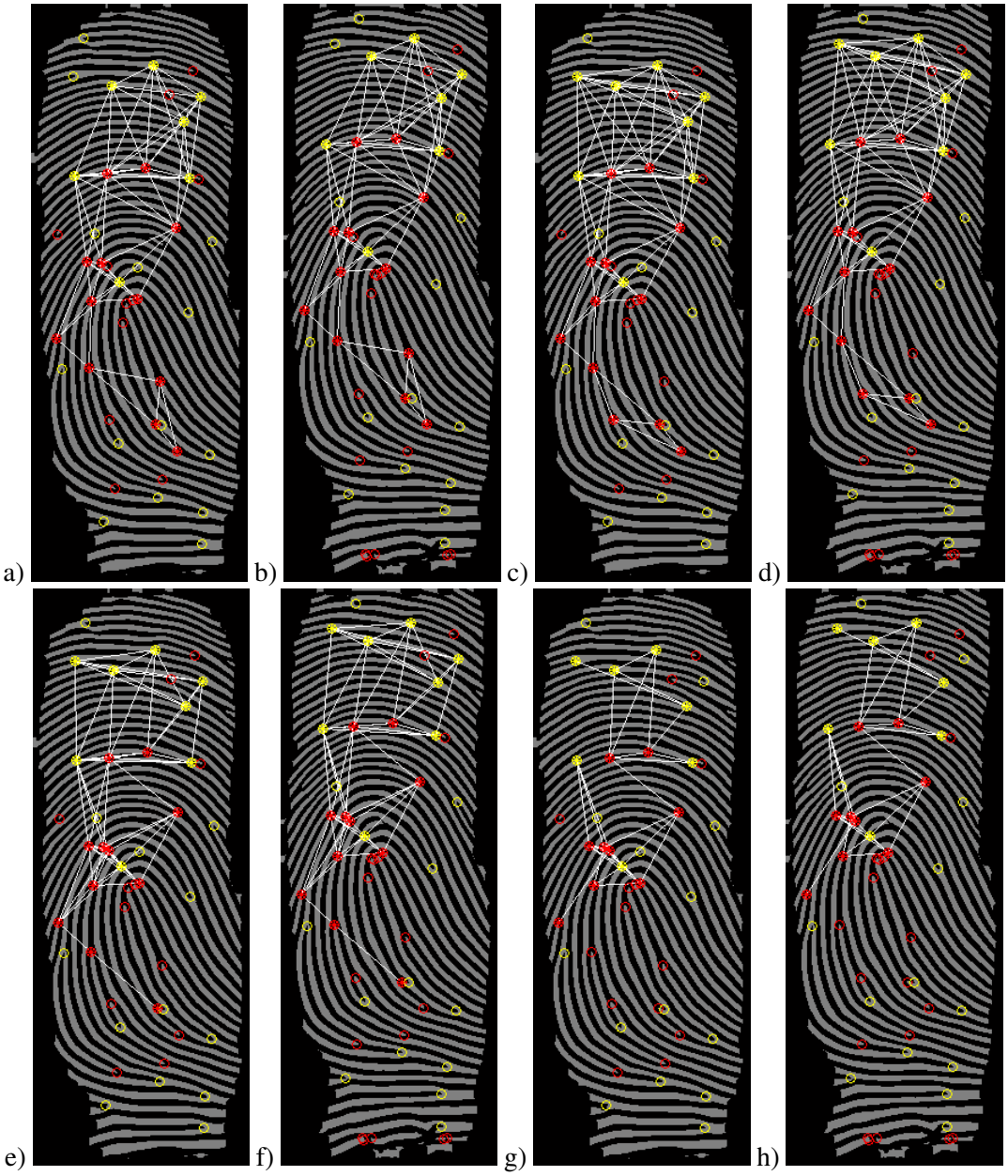
Tablica 6.3: Wyniki porównania tych samych grafów dla różnych wartości parametrów

6.3. Podsumowanie testów

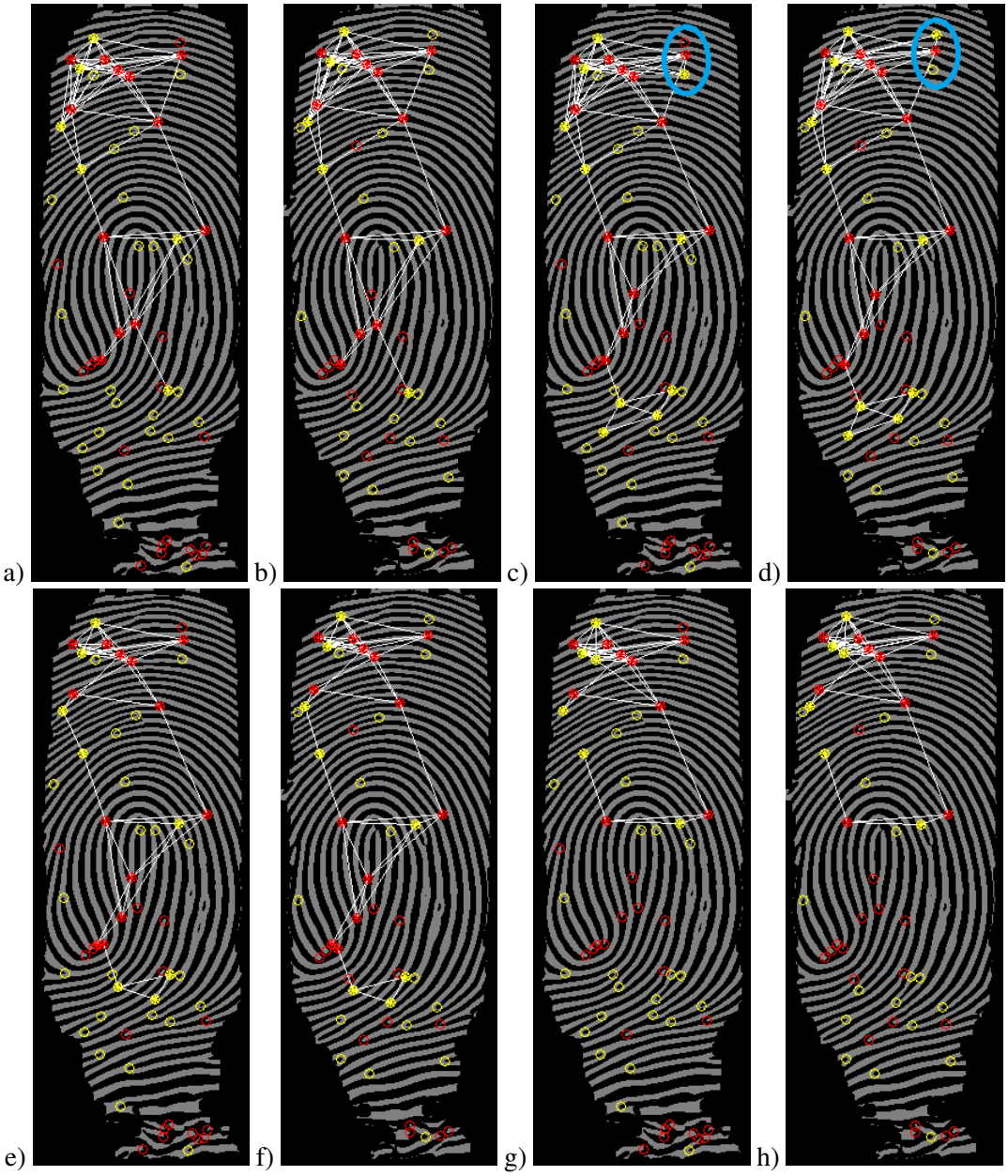
Powyższe przykłady pokazują, że każdy przypadek jest inny. Dlatego wszystkie testy zostały zestawione w tabeli 6.4. Podano w niej wyniki filtracji danych obrazów dla trzech zestawów parametrów, które zwracają najlepsze wyniki, a także parametry przy których dla danej pary obrazów otrzymano



Rysunek 6.2: Przykład 1. Wyniki eksperymentów dla różnych współczynników: a,b - *epsilon* = 1, *tolerancja* = 30; c, d - *epsilon* = 1, *tolerancja* = 20; e, f - *epsilon* = 5, *tolerancja* = 20; g, h - *epsilon* = 5, *tolerancja* = 10



Rysunek 6.3: Przykład 2. Wyniki eksperymentów dla różnych współczynników: a,b - *epsilon* = 5, *tolerancja* = 30; c, d - *epsilon* = 5, *tolerancja* = 40; e, f - *epsilon* = 2, *tolerancja* = 30; g, h - *epsilon* = 1, *tolerancja* = 30



Rysunek 6.4: Przykład 3. Wyniki eksperymentów dla różnych współczynników: a,b - *epsilon* = 5, *tolerancja* = 30; c, d - *epsilon* = 5, *tolerancja* = 40; e, f - *epsilon* = 2, *tolerancja* = 40; g, h - *epsilon* = 5, *tolerancja* = 20

Lp.	ilość zgodnych minucji (zgodność %)			parametry najlepszego wyniku
	dla:$\epsilon = 5$, *tol* = 30	dla:$\epsilon = 5$, *tol* = 40	dla:$\epsilon = 2$, *tol* = 40	*epsilon / tolerancja*
1	28 (65)	28 (65)	28 (65)	5 / 30 lub 2 / 40
2	19 (39)	20 (41)	20 (41)	od 2 do 5/ 40
3	18 (35)	*źle*	24 (44)	2 / 40
4	15 (27)	20 (35)	19 (34)	5 / 40
5	18 (19)	23 (25)	21 (23)	5 / 40
6	16 (30)	*źle*	15 (28)	5 / 30
7	*źle*	*źle*	24 (47)	2 / 40
8	*źle*	*źle*	26 (54)	2 / 40
9	19 (29)	*źle*	24 (36)	2 / 40
10	18 (30)	*źle*	14 (23)	5 / 30
11	*źle*	*źle*	13 (30)	2 / 40
12	11 (22)	17 (34)	12 (24)	5 / 40
13	11 (24)	*źle*	10 (22)	5 / 30
14	15 (38)	19 (48)	19 (48)	od 2 do 5 / 40
15	18 (40)	*źle*	21 (47)	2 / 40
16	12 (27)	13 (30)	10 (23)	5 / 40
17	12 (19)	15 (24)	11 (18)	5 / 40
18	18 (30)	*źle*	21 (35)	2 / 40
19	17 (36)	*źle*	17 (36)	5 / 30 lub 2 / 40
20	21 (53)	*źle*	20 (50)	5 / 30
21	19 (27)	*źle*	12 (17)	5 / 30
22	29 (40)	*źle*	29 (40)	5 / 30 lub 2 / 40
23	16 (22)	22 (30)	12 (16)	5 / 40
24	17 (27)	25 (40)	23 (37)	5 / 40
25	22 (31)	*źle*	21 (29)	5 / 30
26	15 (27)	*źle*	14 (26)	5 / 30
27	19 (30)	21 (34)	19 (30)	5 / 40
28	18 (35)	19 (37)	18 (35)	5 / 40
29	12 (29)	16 (39)	11 (27)	5 / 40
30	12 (30)	*źle*	6 (15)	5 / 30
31	*źle*	*źle*	27 (42)	2 / 40

Tablica 6.4: Zbiorcze zestawienie wyników testów

najlepszy wynik bez błędnych dopasowań. W przypadku złego dopasowania (którego przykład przedstawiono już na rysunku 6.4 c, d) w tabeli jest wpisywane słowo *źle* zamiast wyników, które i tak nie są wiarygodne.

6.4. Wnioski

Jak widać w powyższej tabeli dobranie jednych wartości parametrów jest bardzo kłopotliwe. Najwięcej prawidłowych maksymalnych dopasowań znaleziono dla parametrów *epsilon* = 5, *tolerancja* = 40. Jednocześnie dla tych samych wartości parametrów znaleziono najwięcej błędnych dopasowań. Kolejnym zestawem parametrów, dającym dużo maksymalnych dopasowań są: *epsilon* = 5, *tolerancja* = 30. Jednak ten zestaw również generuje nieliczne błędne dopasowania. Płynie z tego wniosek, że nie należy szukać za wszelką cenę maksymalnych dopasowań, a skupić się na dających wystarczająco dużą zgodność i będących zawsze prawidłowym dopasowaniem. Takie wyniki dają parametry *epsilon* = 2, *tolerancja* = 40.

7. Podsumowanie

Niniejsza praca prezentuje opis stanowiska laboratoryjnego do analizy i obserwacji linii papilarnych. Czytnik linii papilarnych umożliwia pobranie śladów linii papilarnych, natomiast dołączone oprogramowanie pozwala na filtrację, ekstrakcję cech(minucji), a także porównanie dwóch śladów pod względem zgodności. Wszystkie etapy pozwalają na dowolną konfiguracje parametrów, bądź użycie dla nich wartości domyślnych. Użytkownik może dokładnie analizować poszczególne etapy na wyświetlonych obrazach, bądź tylko otrzymać odpowiedź czy dwa pobrane ślady pochodzą z tego samego palca i w ilu procentach się pokrywają. Na koniec pracy pokazano jaka jest skuteczność działania oprogramowania w zależności od użytych parametrów.

Powstałe stanowisko laboratoryjne z powodzeniem może służyć do celów edukacyjnych.

Bibliografia

[1] Historia zastosowania linii papilarnych w kryminalistyce. http://www.kryminalistyka.fr.pl/crime_daktyloskopia.php.

[2] http://pl.wikipedia.org/wiki/Linie_papilarne.

[3] Źródło obrazka. http://www.paznokcie.wiedziec.pl/pielegnacja_dloni/odciski_palcow.html.

[4] id3 semiconductors. http://www.id3.eu/en/product_48/match-on-card.htm.

[5] Przedsiębiorstwo Innowacyjno-Wdrożeniowe FORTECH sp. z o.o. Opis systemów biometrycznych. http://www.fortech.krakow.pl/pl/sat_met_biomet.php.

[6] Olejowski W. Jackiewicz K. *Interfejs do akwizycji danych biometrycznych*. Politechnika Warszawska, 2005. http://home.elka.pw.edu.pl/~wolejows/fingerprint.

[7] Źródło obrazka. http://www.idstore.pl.

[8] Źródło obrazka. http://fingerchip.pagesperso-orange.fr/biometrics/types/fingerprint.htm.

[9] Resnick R. Halliday D. *Podstawy fizyki t.2*. Wydawnictwo PWM, Warszawa, 2008. ISBN 978-83-01-14107-3.

[10] PBP OPTEL Sp. z o.o. Odczytywanie struktury linii papilarnych za pomocą kamery ultradźwiękowej. http://www.optel.com.pl.

[11] *Pakiet SDK (ang. Software Development Kit) do obsługi czytnika EIKON*. http://www.upek.com/solutions/pc_and_networking/sdks/windows/.

[12] UPEK. *BSAPI Reference Manual*, 2007.

[13] Kovesi P. D. "MATLAB and Octave functions for computer vision and image processing". Centre for Exploration Targeting, School of Earth and Environment, The University of Western Australia. http://www.csse.uwa.edu.au/~pk/research/matlabfns/.

[14] Dziwiński T. *Identyfikacja osób na podstawie linii papilarnych*. Kraków, 2009. http://www.focus.agh.edu.pl/index.php?s=projects.

[15] Rotter P. *Transformacje kinematyczne i wizyjne w robotyce*. wykłady.

[16] Lamport L. *Automatic Fingerprint Recognition Systems*. Springer, New York, 2003. ISBN 0-387-95593-3.

[17] Sedgewick R. *Algorytmy w C++ : grafy*. Wydawnictwo RM, Warszawa, 2003. ISBN 83-7243-264-3.

[18] Drozdek A. *C++ - algorytmy i struktury danych*. Wydawnictwo Helion, Gliwice, 2004. ISBN 83-7361-385-4.

[19] Ullmann J. R. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(I):31–42, 1976. `http://www.engr.uconn.edu/~vkk06001/GraphIsomorphism/Papers/Ullman_Algorithm.pdf`.

[20] Strona zawierająca aktualną wersję biblioteki OpenCV. `http://sourceforge.net/projects/opencvlibrary`.

[21] Rusiecki A. Rafajłowicz E., Rafajłowicz W. *Algorytmy przetwarzania obrazów i wstęp do pracy z biblioteką OpenCV*. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2009. ISBN 978-83-7493-475-6.

Spis rysunków

A. Opis plików programu upek.exe

Program został napisany w języku C++ i skompilowany w środowisku projektowym Microsoft Visual C++ 2005. Poniżej znajduje się lista plików z programem:

main.cpp - główny plik, inicjujący komunikację z czytnikiem i wyświetlający okna

czytnik.cpp - zawiera funkcje komunikacji z czytnikiem i przygotowania danych do zapisu do pliku

filtr.cpp - zawiera funkcje odpowiedzialne za wyznaczenie mapy orientacji

Na płycie CD oprócz plików z programem zostały dołączone wymagane do uruchomienia programu biblioteki *dll* i inne dodatki:

bsapi.dll - biblioteka zwiera funkcje komunikacji z czytnikiem

bsgui.dll - zawiera implementację domyślnej funkcji wywołania zwrotnego (ang. callback) wywoływanej przez bsapi.dll

bsgui.zip - archiwum zawiera pliki graficzne wymagane przez bsgui.dll

msvcp100d.dll - implementacja CRT (ang. C Runtime Library), wymagana, gdy uruchamiamy program na komputerze bez zainstalowanego oprogramowania Microsoft Visual Studio

msvcr100d.dll - implementacja biblioteki Standardowej C++, wymagana, gdy uruchamiamy program na komputerze bez zainstalowanego oprogramowania Microsoft Visual Studio

opencv_core220d.dll - podstawowe struktury OpenCV

opencv_highgui220d.dll - obsługa wejścia-wyjścia oraz GUI

opencv_imgproc220d.dll - operacje na obrazach

Przed pierwszym uruchomieniem programu należy zainstalować sterowniki czytnika. Najświeższe sterowniki można pobrać ze strony internetowej producenta. Na płycie CD znajdują się sterowniki, które były dołączone do czytnika w momencie zakupu.

Najpierw należy podłączyć czytnik do wolnego portu USB. Jeżeli system nie wykryje samodzielnie nowego urządzenia, należy otworzyć *Menadżer urządzeń* w Panelu sterowania, i aktualizować listę urządzeń. Czytnik powinien być zidentyfikowany jako *Nieznane urządzenie*. Należy zatem zaznaczyć to urządzenie i z menu kontekstowego wybrać *Aktualizuj sterownik*. Następnie należy wybrać instalację użytkownika zamiast domyślnej i w kolejnym kroku wskazać lokalizację sterowników na dysku lub płycie CD. Po chwili czytnik jest gotowy do użycia.

B. Opis plików programu finger.fig

Program został napisany w środowisku MATLAB (wersja 2009b). Poniżej znajduje się lista plików z programem:

binaryzacja.m - funkcja pomocnicza dokonująca binaryzacji z zadanym progiem

finger.fig - zawiera opis wyglądu GUI

finger.m - zawiera implementację funkconalności GUI

filtr.m - zbiór funkcji dokonujących filtracji filtrem Gabora

maska.m - funkcja generująca maski do pomijania fragmentów obrazu i odrzucania fałszywych minucji

normalizacja.m - funkcja pomocnicza dokonująca normalizacji

normalizacja1.m - funkcja pomocnicza sprowadzająca wartości na obrazie do zakresu $0 \div 1$

wykrywaj.m - zbiór funkcji służacych do ekstrakcji minucji i generowania grafu

Do poprawnego działania programu wymagane jest umieszczenie w jednym katalogu z powyższymi skryptami zbudowanych aplikacji opisanych w dodatkach A oraz C.

C. Opis plików programu porownanie.exe

Program został napisany w języku C++ i skompilowany w środowisku projektowym Microsoft Visual C++ 2005. Poniżej znajduje się lista plików z programem:

main.cpp - program główny zawierający wszystkie funkcje

graf.cpp - plik zawierający implementację własnej klasy *graf*

Program nie wymaga żadnych dodatkowych bibliotek.

D. Zawartość płyty CD

Dołączona do pracy płyta CD zawiera:

- kod źródłowy programu upek.exe (projekt w środowisku Microsoft Visual C++ 2005)
- kod źródłowy programu porownanie.exe (projekt w środowisku Microsoft Visual C++ 2005)
- skompilowane wersje obu powyższych programów
- biblioteki *dll* niezbędne do uruchomienia aplikacji upek.exe
- kod źródłowy programu finger.fig wykonywany przez program Matlab
- sterowniki do czytnika EIKON firmy UPEK
- wersja elektroniczna niniejszej pracy (plik PDF)
- pliki użyte w testach