Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych
Instytut Automatyki i Informatyki Stosowanej

Rok akademicki 2012/2013

Praca dyplomowa inżynierska

Cezary Guz

Odległość, podobieństwo i niepodobieństwo obrazów na przykładzie odcisków palców

Opiekun pracy:
prof. dr hab. Andrzej Pacut

Ocena .....

.....
Podpis Przewodniczącego
Komisji Egzaminu Dyplomowego

Specjalność: Informatyka –
Systemy informacyjno decyzyjne

Data urodzenia: 8 stycznia 1989 r.

Data rozpoczęcia studiów: 1 października 2008 r.

Życiorys

Nazywam się Guz Cezary urodziłem się 8 stycznia 1989r. w Lubartowie. W 2005 roku ukończyłem Publiczne gimnazjum Nr 4 im Komisji Edukacji Narodowej w Puławach. Swoją edukację następnie kontynuowałem w II liceum ogólnokształcącym im. Komisji Edukacji Narodowej w Puławach, w klasie o profilu matematyczno-fizyczno-informatycznym. W październiku 2008 roku rozpocząłem studia na wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej na kierunku Informatyka.

.....
podpis studenta

Egzamin dyplomowy

Złożył egzamin dyplomowy w dn. .....

Z wynikiem .....

Ogólny wynik studiów .....

Dodatkowe wnioski i uwagi Komisji .....

.....

Streszczenie

Praca ta opisuje implementację i wyniki różnych wariantów algorytmu kodującego. Przedstawia wady i zalety takiego podejścia. Praca nie ma na celu stworzenia kolejnego oprogramowania do porównywania odcisków, a jedynie zbadanie możliwości zastosowania innego podejścia. W pracy przedstawiane są rożne metody porównywania kodów. Analizowane są metody porównywania w skali podobieństwa, niepodobieństwa oraz obu tych wskaźników jednocześnie. Wyniki uzyskanego rozwiązania prezentowane są na tle komercyjnego rozwiązania.

Słowa kluczowe: Biometria, odcisk palca, kod odcisku

Distance, similarity and dissimilarity of images for fingerprints

Abstract

This thesis describes implementation and score of fingerprints compare. Described implementation uses code algorthm. Thesis shows advantages and disadvantages of presented methods. Crating a new software for fingerprints compare is not a target of this thesis. Thesis use several methods of code compare. Presented methods compare fingerprints using similarity, dissimilarity and both of this feature. Score of using algorith is compare with commercial solution.

Key words: Biometric, fingerprint, finger code

Spis treści

1. Wstęp

Rozdział ten poświęcony jest opisowi aktualnie używanych metod w biometrii służących do identyfikacji poprzez porównywanie odcisków palców, oraz sposobów pozyskiwania obrazów odcisków.

1.1. Sposoby pozyskiwania obrazów odcisków

Aby można było mówić o identyfikacji użytkownika poprzez odcisk palca obraz taki należy od niego pobrać. Istnieje kilka metod pobierania odcisków. Niektóre z nich są używane powszechnie, inne to albo specyficzne sposoby dla policji lub testowane nowinki techniczne. Ciężko jest je klasyfikować pod względem lepszy-gorszy, Każdy z przedstawianych sposobów jest przeznaczony do innych celów i powinien być stosowany w zależności od przeznaczenia pobranej próbki.

Sposoby pobierania odcisku można klasyfikować poprzez sposób przykładania palca do sensora. Mówiąc sensor, chodzi o dostarczone urządzenie zapisujące odcisk. Może to być zarówno czytnik optyczny jak i atrament z kartką papieru. Wyróżniono trzy metody:

- Rolowanie (ang. Rolled finger)
- Pobór statyczny (ang. Static sensing)
- Pobór poprzez poślizg (ang. Sweep (swipe) reading)

Rolowanie, sposób aktualnie używany głównie przez służby policyjne, tylko w połączeniu z kartką i atramentem, Jest to jeden z pierwszych sposobów pobierania odcisków.[1] Główną zaletą takiego sposobu jest pozyskanie całego odcisku od "brzegu do brzegu", ma to sprzyjać maksymalizacji prawidłowych rozpoznań podejrzanych. Policja, jak też inne służby mundurowenie nie polega jedynie na komercyjnych systemach biometrycznych. Zamiast tego mają sztab specjalistów rozpoznających odciski po najdrobniejszych szczegółach. Często muszą badać odciski utajone, czyli takie pozostawione niejawnie np. na szklance lub innym przedmiocie zbrodni. Dlatego ważne jest aby pozyskać jak największą część odcisku. Metoda ta nie ma najmniejszych szans komercyjnego zastosowania, gdyż niesie za sobą silne skojarzenia kryminalne i źle kojarzy się społeczeństwu.

Pobór statyczny, jedna z obecnie najpopularniejszych metod poboru odcisków. Użytkownik musi wyłącznie położyć palec w wyznaczonym miejscu i nie ruszać nim do momentu pobrania próbki. Rozwiązanie jest na tyle proste, że użytkownik nie

[1] Oczywiście istnieją starsze sposoby pozyskiwania odcisków, jak choćby stosowane w starożytności gliniane tabliczki z odciskiem dłoni pomagające identyfikować kupców

potrzebuje żadnego przeszkolenia aby samodzielnie go używać. Niestety rozwiązanie to ma kilka wad. Po pierwsze nie jest oczywiste z jaką siłą należy nacisnąć na sensor. Silniejszy nacisk spowoduje dokładniejsze pobranie próbki. Jeżeli pobranie odcisku trwa zbyt długo, użytkownik ma naturalną tendencję do niecierpliwienia się, oraz niepotrzebnego naciskania na sensor. Zbyt silny nacisk ,oże w przyszłości skutkować uszkodzeniem urządzenia. Niemożliwe jest również utrzymanie palca w całkowitym bezruchu, co w oczywisty sposób powoduje niekorzystne zniekształcenia obrazu. Dodatkowo sensor po niedługim czasie używania staje się brudny, a zabrudzenia są rejestrowane jako fragmenty odcisku. Najważniejszą wadą rozwiązania jest pozostawianie własnego odcisku na samym sensorze. W przypadku gdy użytkownik stosuje odcisk palca jako klucz do systemu zabezpieczeń, zostawia wszystkie dane niezbędne do włamania na czytniku.

Pobór poprzez poślizg jedna z nowszych metod. Używana głównie w zabezpieczaniu urządzeń takich jak laptop, tablet czy telefon. Pomimo iż urządzenia tego typu są pełne odcisków użytkownika sposób pobrania próbki nieco utrudnia powielenie odcisku i zaaplikowanie go na czytniku. Rozwiązanie to również nie jest pozbawione wad. Przede wszystkim nie jest to naturalny sposób pozostawiania odcisku, czyli osoby korzystające z takiego czytnika muszą najpierw nauczyć się go obsługiwać. Zaletą tego rozwiązania jest to iż użytkownik nie zostawia na samym czytniku swojego odcisku. Jeżeli taki sposób pobierania odcisku zamontowano by np w bankomatach oszuści nie byli by w stanie ukraść naszych odcisków. Nieco inaczej sprawa ma się z naszymi prywatnymi urządzeniami. Dodatkowo czytnik jest czysty, każde pobranie odcisku czyści sensor, w ten sposób eliminuje się zakłócenia obrazu.

Sposoby pobierania odcisku można również podzielić ze względu na zastosowaną technologię.

- Atrament i kartka
- Czytnik optyczny
- Czytnik optyczno elektryczny
- Czytnik pojemnościowy
- Czytnik naciskowy
- Czytnik ultrasonograficzny

Czytniki optyczne wykorzystują zasadę całkowitego wewnętrznego odbicia światła. Użytkownik przykłada palec do szklanej powierzchni, która podświetlana od spodu przez lampę. W klasycznym rozwiązaniu użytkownik przykłada palec do podświetlanej szybki. W miejscu, w którym grzbiety linii papilarnych dotykają czytnika światło jest pochłaniane, pozostały odbity obraz służy do rekonstrukcji odczytanego odcisku. Istnieją również wersje bezdotykowe, w których palec jest oświetlany od spodu, jak też wersje do pobierania odcisku przez poślizg.

Czytniki optyczno elektryczne wykorzystują do swojego działania polimery, które są w stanie emitować światło, gdy są pobudzane odpowiednim napięciem. Czytnik emituje światło w miejscu gdzie grzbiety linii papilarnych go dotykają. Ponieważ czytnik połączony jest z kamerą CMOS, w łatwy sposób można odtworzyć obraz odcisku.

Czytniki pojemnościowe i naciskowe działają na podobnej zasadzie i wykorzystują nierówności palca wynikające z istnienia grzbietów i dolin linii papilarnych. W pierwszym wariancie w miejscu gdzie palec nie dotyka do czytnika tworzy się "kondensator" o mierzalnej pojemności, w przypadku drugim badamy miejsca gdzie grzbiety naciskają na czytnik.

Czytniki ultrasonograficzne są nowinką techniczną w sposobie pobierania odcisków. Niestety urządzenia te są dość dużych rozmiarów i są drogie. Dodatkowo pobranie odcisku zajmuje więcej czasu niż w przedstawianych powyżej metodach. Na razie wydają się być nieodpowiednie dla komercyjnej produkcji.

1.2. Dotychczasowe metody identyfikacji

1.2.1. Metody minucyjne

Metody minucyjne są obecnie jednymi z najczęściej stosowanych metod porównywania odcisków palców. Wynika to z ich prostoty opisu podobieństwa, a także akceptacji tego sposobu przez prawo w większości krajów. Metoda porównuje zebrana próbkę biometryczną z zapisanym w bazie wzorcem. Można to nazwać relacją wzorzec-próbka, gdzie jeden lub więcej odcisków stanowi bazę wiedzy, próbka to odcisk palca który chcemy porównać z wzorcem. Metody minucyjne szukają jak największej liczby zgodnych minucji i na jej podstawie orzekają podobieństwo bądź niepodobieństwo, nie jest brana pod uwagę procentowa zgodność, a jedynie liczba minucji. Metody te nie biorą pod uwagę kształtu ani układu linii papilarnych. W powszechnie stosowanych systemach liczba niezgodnych minucji nie jest brana pod uwagę. Systemy tak naprawdę szukają jedynie podobieństw. Jest to jedna z pierwszych metod porównywania odcisków i jednocześnie najskuteczniejsza.

1.2.2. Metody nieminucyjne

Pomimo wysokiej skuteczności metod minucyjnych mają one kilka wad dla których warto jest szukać metod nieminucyjnych. Po pierwsze wiarygodne wyznaczenie minucji z obrazu o niskiej jakości jest bardzo trudne, a czasem wręcz niemożliwe. Ekstrakcja minucji jest kosztowna. Pomimo iż postęp obliczeniowy jest ogromny i być może doczekamy się czasów gdy ilość pamięci RAM nie będzie praktyczni ograniczona, to na razie czas obliczeń jest istotnym problemem. Metody nieminucyjne powinny być "lżejsze" obliczeniowo. Metody nieminucyjne wcale nie muszą być używane jako zastępnik metod minucyjnych, a jedynie jako metoda wspomagająca. Być może rozwiązanie takie przyczyni się do zwiększenia skuteczności porównywania odcisków.

Metody te opierają się na porównywaniu innych cech odcisków. Należą do nich:

- Rozmiar, kształt i ogólna sylwetka odcisku
- Liczba, typ i pozycja punktów osobliwych odcisku
- Relacje przestrzenne między liniami papilarnymi
- Rozkład, położenie i liczba porów

1.3. Cel Pracy

Celem pracy jest zbadanie możliwości odejścia od obiektowego modelu reprezentacji minucji w porównywaniu odcisków palców. Należy również zdefiniować podobieństwo i niepodobieństwo odcisków. Należy zbadać możliwość stworzenia kodu odcisku, zaprogramować narzędzia umożliwiające porównywanie odcisków poprzez jego kod. Porównywania powinny odbywać się pod względem podobieństwa, niepodobieństwa oraz obu tych cech jednocześnie. Jest to o tyle ważne iż obecnie stosowane metody pomijają aspekt niepodobieństwa obrazów. Należy przeprowadzić analizę otrzymanych wyników i porównać je na tle komercyjnego rozwiązania. Celem pracy nie jest stworzenie kolejnego oprogramowania do porównywania odcisków. Nie jest to zadanie łatwe, a tym bardziej nie jest przewidziane dla jednej osoby.

2. Kod odcisku

Rozdział ten poświęcony jest opisowi zastosowanego algorytmu, pozwalającego na uzyskanie kodu odcisku. Metody opisywane w poprzednim rozdziale skupiają się na porównywaniu obiektów(minucji) uwzględniając ich położenie, kąt oraz typ. Zaawansowane algorytmy wykorzystują dodatkowo jakość odcisku (jest to wyliczana dla każdej minucji wielkość na podstawie jakości próbki biometrycznej). Kolejną cechą którą można brać pod uwagę jest wzajemna korelacja minucji. Dopasowane minucje można utożsamiać z wierzchołkami grafu, wtedy dopasowywane punkty w obu obrazach powinny być podobne.

Wadą takiego podejścia jest ich złożoność obliczeniowa dlatego zaproponowano podejście prostsze, opierające się na stworzeniu kodu odcisku. Dodatkowo aktualne rozwiązania stwierdzają zgodność w porównywaniu odcisków biorąc pod uwagę ilość zgodnych minucji. Kod odcisku umożliwia porównywanie całości próbki biometrycznej nie tylko jej części podobnej.

2.1. Algorytm Kodujący

Tworzony algorytm od samego początku miał narzucone ograniczenia a mianowicie:

- unikanie podejścia obiektowego
- możliwie jak najprostszy
- łatwo konfigurowalny
- prosta funkcja porównująca kody(najlepiej XOR[1])
- powinien być skuteczny
- niska złożoność obliczeniowa

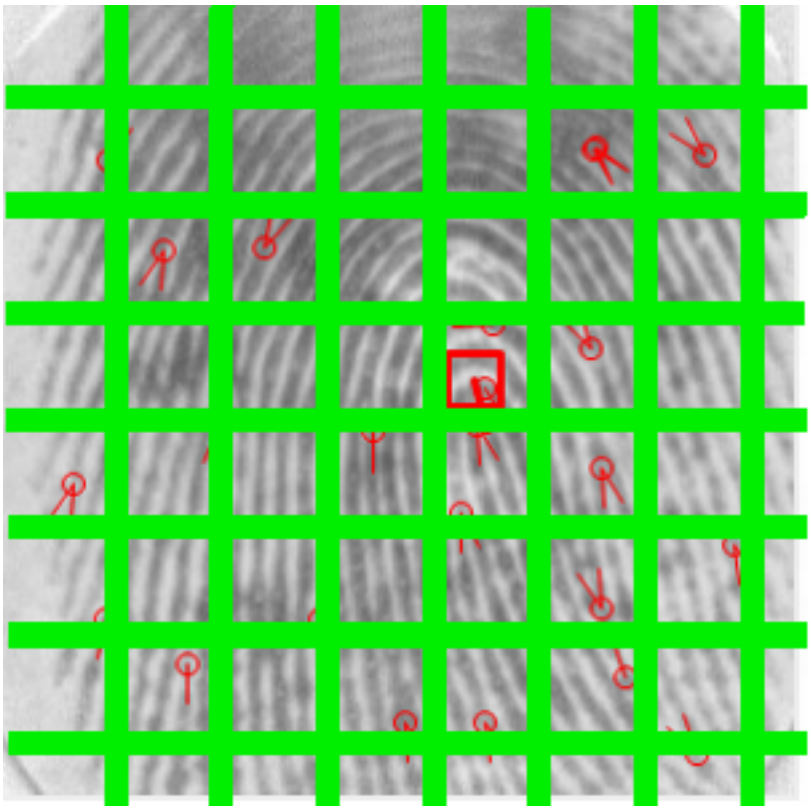
O ile spełnienie pierwszych trzech wymagań jest w miarę proste kolejne stwarzają problemy. Prosty algorytm kodujący niekoniecznie będzie miał prostą funkcją porównującą, dodatkowo nie zawsze musi być skuteczny. Zwiększanie stopnia skomplikowania algorytmu może skutecznie wpłynąć na jego skuteczność ale zwiększa złożoność obliczeniową. Dodatkowo w trakcie zwiększania trudności algorytmu kod magazynuję coraz więcej informacji o obrazie stając się niejako innym zapisem metod obiektowych. Nie trudno zauważyć iż spełnienie wszystkich wymagań staję się niemożliwe, a zaproponowane rozwiązanie jest jedynie najbardziej optymalnym wyjściem z sytuacji

2.1.1. Naiwny kod

Algorytm opiera się na bardzo prostym pojmowaniu kodu odcisku. Algorytm jako wejście otrzymuje listę minucji wraz z ich współrzędnymi XY obrazu oraz kąt

[1] ang. *Exclusive or* - Alternatywa wykluczająca zwracająca prawdę dla dwóch przeciwnych wartości

minucji. Algorytm zakłada, iż odciski pochodzące od tej samej osoby będą miały minucje poukładane w taki sam sposób. Dlatego koduje on sposób ułożenia minucji na odcisku. Zasada działania polega na podziale całego obszaru obrazu na mniejsze fragmenty. Ich wielkość jest tak dobrana aby znaczna część minucji znajdowała się samodzielnie w swoich fragmentach. W praktyce zdarza się, że dwie minucje leżą w tym samym fragmencie. Ponieważ kształt musi być taki sam dla wszystkich kodów przyjęto iż obszar obrazu podzielony będzie na jednakowe kwadraty tworząc siatkę na obrazie. Kwadrat, w którym znajduje się przynajmniej jedna minucja oznaczany jest jako "1"w kodzie pozostałe obszary to "0".



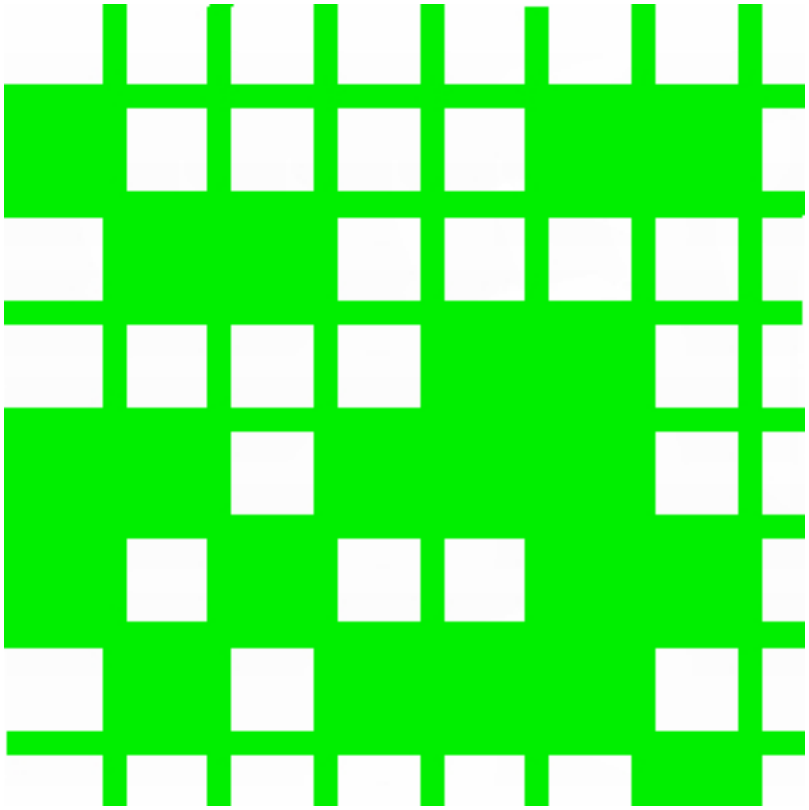
Rysunek 2.1. Przykładowa siatka kodowa na tle odcisku

Na ilustracji 2.1 można zauważyć fragmenty "kratki w której zmieściły się dwie minucję", po otrzymaniu kodu informacja ta zostanie stracona i silnik porównujący nie będzie wiedział czy w kratce była jedna czy więcej minucji. Rzeczywiste porównanie uwzględnia wartości kątów minucji leżących w tym samym kwadracie i dopiero na jej podstawie stwierdza zgodność.

Na rysunku 2.2 pokazano otrzymany kod, który dla odpowiedniej(stałej) kolejności odczytywania utworzy binarny kod zer i jedynek.

Ocena przydatności

1. Zalety rozwiązania
   - prostota algorytmu
   - kod odcisku zajmuje bardzo mało pamięci
   - XOR jako funkcja porównująca kody



Rysunek 2.2. Otrzymany kod

- bardzo niska złożoność obliczeniowa

2. Wady rozwiązania
   - utrata informacji o minucji w przypadku za dużych podprzestrzeni
   - wrażliwość na przekształcenia
   - niska, wręcz zerowa skuteczność porównywania odcisków
   - wysoka wrażliwość na odciski o bardzo dużej liczbie minucji

Reasumując rozwiązanie tego typu nie można w żadnym stopni nazywać biometrią, zastosowane zostało jedynie jako wprawka dla następnych wersji algorytmu kodującego. Ponieważ algorytm jest nieskuteczny należy zastanowić się nad przyczynami jego wad. Po pierwsze utratę informacji o minucjach zlokalizowanych w tej samej podprzestrzeni można zniwelować poprzez dodanie jeszcze jednego wymiaru do kodu, kąta pomiędzy styczną do grzbietu[2] a osia odciętych. Oczywiście może zdarzyć się, że minucje początkowo znajdujące się w tej samej podprzestrzeni nawet po dodaniu trzeciej współrzędnej dalej będą się w niej znajdować lecz prawdopodobieństwo to jest mniejsze. Świadczyć o tym może fakt iż nagromadzenia minucji często znajdują się w miejscach gdzie linie papilarne zakręcają, tworząc pętle i wiry. Kąty tych minucji powinny być zupełnie inne. Rozwiązanie to zapobiega również błędnym porównaniom odcisków o dużej liczbie minucji. Odciski takie często mają rozkład minucji powodujący żakodowanie"większości podprzestrzeni.

[2] przyjmuje się, że czarne części linii papilarnych nazywane są grzbietami białe dolinami

Sytuacja ta jest na tyle uciążliwa, że dla pewnych wielkości siatki dowolne dwa odciski były zaznaczane jako pochodzące od tej samej osoby. Dodanie kąta jako trzeciej wartości w kodzie nieco rozprasza zakodowanie. Kolejnym problemem jest wysoka wrażliwość na przekształcenia, rozwiązaniem było zastosowanie sztucznych przekształceń obrazu i kodowanie wszystkich uzyskanych obrazów. Tworzy się w ten sposób wiele kodów dla jednego odcisku. Wszystkie te rozwiązania mają na celu polepszenie skuteczności algorytmu.

2.1.2. Macierz kodów

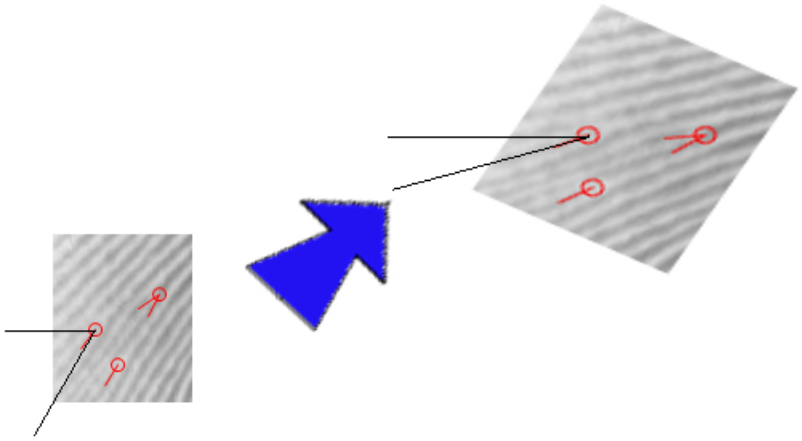
Opisywany poniżej algorytm jest ulepszoną wersja naiwnego kodowania, eliminującą podstawowe wady pierwszego rozwiązania. Algorytm polega na kodowaniu kolejnych transformacji obrazu odcisku. Transformacje polegają na wykonaniu translacji obrazu o sprecyzowane wektory, oraz obracanie obrazu dla każdej translacji. Nieruchoma pozostaje ramka w której znajduje się wzorcowy obraz. Minucje które po transformacji znajdują się poza ramką są pomijane. Symuluje to uzyskanie odcisku poprzez inne przyłożenie palca do czytnika. Dla każdej otrzymanej transformacji obrazu wejściowego tworzony jest naiwny kod. Przykładowa transformacja obrazu pokazana jest na rysunku 2.3 Wynikowym kodem odcisku jest sklejenie wszystkich otrzymanych kodów i liczba segmentów kodu czyli liczba translacji wykonanych na obrazie. W praktycznym zastosowaniu nie stosuje transformacji obrazu a jedynie listy minucji. W przypadku porównywania stosowana jest zasada wzorzec-próbka. Jeden z odcisków traktowany jest jako odcisk o większej jakości niż drugi, dla niego tworzony jest wyłącznie naiwny kod. Ma to odzwierciedlenie w pobieraniu próbek biometrycznych z reguły jest tak, że nad procesem pobierania próbki czuwa osoba odpowiedzialna za system, odcisk powinien być pobrany poprawnie tak aby był jak najlepszej jakości. Drugi odcisk pochodzi z systemu nad którym nikt nie czuwa, dla niego tworzona jest macierz kodów. Dokładnie jak w zastosowaniu codziennym użytkownik systemu proszony jest(najczęściej przez automat) o przyłożenie palca do skanera w celu pobrania próbki. Nie ma żadnej gwarancji iż poprawnie złoży próbkę. Założenie to odzwierciedla warunki rzeczywiste pozyskiwania odcisków. Następnym krokiem jest porównanie odcisku. Algorytm bierze kod wzorca i porównuje ze wszystkimi naiwnymi kodami macierzy kodów.

Porównując pojedyncze kody możliwe są trzy przypadki porównań

- **1 : 1** dopasowanie, elementy wzorca są zgodne z próbką
- **1 : 0** niedopasowanie wzorca, element wzorca nie istnieje w próbce
- **0 : 1** niedopasowanie próbki, element próbki nie istnieje we wzorcu

Wybierany jest najlepszy wynik porównania tzn. taki dla którego liczba porównań 1:1 jest największa. Jako wskaźnik stosowano również minimalizację liczby niedopasowań wzorca i próbki. Niestety dla takich wymagań algorytm wybierał taką transformację dopasowanego obrazu próbki dla której większość minucji znajdowała się poza ramką obrazu, czyli były pomijane. Ubocznym skutkiem poprawienia wskaźnika było zmniejszanie liczby porównywanych minucji. Zaburza to rzeczywiste porównywanie kodów.

Istotny wpływ na jakość porównania ma wielkość kodu czyli ilość i dokładność informacji w nim przechowywanych. Wielkość kodu musi być dobrana w taki sposób aby dopasowanie było w miarę poprawne i jednocześnie ilość obliczeń



Rysunek 2.3. W trakcie transformacji należy wyliczyć nowe wartości kątów

potrzebnych do porównania nie była zbyt duża. Na wielkość kodu składa się wielkość naiwnego kodu oraz ilość segmentów.

$$SC_{size} = [\frac{X_{cres}}{C_{sep}}] * [\frac{Y_{cres}}{C_{sep}}] * [\frac{A_{cres}}{A_{sep}}] \tag{2.1}$$

$$X_{step} = [X_{cres} * COV_{step}] \tag{2.2}$$

$$Y_{step} = [Y_{cres} * COV_{step}] \tag{2.3}$$

$$SEG_{size} = [\frac{X_{cres}}{X_{step}}] * [\frac{Y_{cres}}{Y_{step}}] * [2 * \frac{A_{res}}{A_{step}}] \tag{2.4}$$

- SC_{size} rozmiar naiwnego kodu
- SEG_{size} ilość segmentów kodowych
- MC_{size} rozmiar macierzy kodów
- X_{cres} szerokość (w pikselach) obrazu z odciskiem
- Y_{cres} wysokość (w pikselach) obrazu z odciskiem
- A_{cres} zakres kąta(podawany w stopniach) obrazu z odciskiem
- A_{res} wartość kąta(podawana w stopniach) o jaką maksymalnie należy obracać obraz
- C_{sep} rozmiar kratki kodu
- X_{step} odstęp w poziomie pomiędzy najbliższymi translacjami obrazu
- Y_{step} odstęp w pionie pomiędzy najbliższymi translacjami obrazu
- COV_{step} wskaźnik odstępu pomiędzy najbliższymi translacjami podawany jako wielokrotność kratki kodu
- A_{step} odstęp kątowy pomiędzy najbliższymi obrotami obrazu

$$MC_{size} = SEG_{size} * SC_{size} \tag{2.5}$$

Przykład 1

Dla obrazu o wymiarach [300 x 300 x 300], rozmiarze kratki kodu [25 x 25 x 30], wskaźniku odstępu 0.5 i odstępie kątowym 15° i maksymalnym kącie obroty 45° otrzymano kod o 2500 segmentów o wielkości jednego segmentu 1440 bitów(0,175kB) co daje razem 3600000 bitów(0,429Mb).

Ocena przydatności

1. Zalety rozwiązania
   - prostoty algorytm kodujący
   - eliminuje podstawowe wady naiwnego kodu
   - skuteczniejszy niż naiwny kod
   - łatwość konfiguracji
2. Wady rozwiązania
   - utrata informacji o minucji w przypadku za dużych podprzestrzeni
   - wrażliwość na przekształcenia nieliniowe
   - średnia skuteczność porównywania odcisków
   - duży rozmiar kodu (większy niż 2^{21} mniejszy niż 2^{22})
   - brak trywialnej funkcji porównującej kody

Mimo, iż kolejna wersja algorytmu jest lepsza od poprzedniej wciąż nie jest pozbawiona wad. Podobnie jak w poprzednim rozwiązaniu za duży rozmiar kratki kodowej może zniekształcać informacje i prowadzić do zbytniego uproszczenia kodu. Duża kratka może obejmować więcej niż jedną minucje. Z drugiej strony zbyt mały rozmiar kratka może prowadzić do zmniejszenia skuteczności algorytmu. Reprezentacja minucji jako otoczenia w którym się znajduje ma symulować zniekształcenia obrazu powstałe w procesie pobierania próbki biometrycznej. Należy uzyskać kompromis pomiędzy tolerancją na zniekształcenia, a skutecznością dopasowania. Prostą zależność pokazują schematyczne wykresy 2.4. Algorytm nie jest odporny na przekształcenia nieliniowe. W procesie pobierania próbki biometrycznej zniekształcenia takie jak spłaszczanie palca nie musi dotyczyć całej jego powierzchni, nie musi też być równomierne. Algorytm stara się kompensować zniekształcenia poprzez wskazywanie jedynie otoczenia w którym znajduje się minucja, a nie jej dokładnego położenia. Dlatego tak istotny jest właściwy dobór wielkości kratki kodu. Przekształcenia mające symulować pobranie próbki w inny sposób niż w procesie rejestracji są wykonywane na całym obrazie. Nie trudno się domyśleć, iż może istnieć takie zniekształcenie obrazu którego nie będzie dało się dopasować do wzorca poprzez obroty i przesunięcia całego obrazu. rozwiązaniem tego problemu byłoby wprowadzenie przekształceń na fragmentach obrazu, zamiast na całości. Wiąże się to ze sporym wzrostem rozmiaru kodu. Rozwiązanie to nie zostało zastosowane. Skuteczność porównywania znacząco wzrosła w porównaniu do poprzedniej metody, szczegółowe wyniki przedstawione są w rozdziale 4. Kolejną wadą jest znaczny wzrost rozmiarów kodu. Rozwiązaniem może być analiza wyglądu kodu. Kod odcisku można porównać wyglądem do macierzy rzadkich. Składa się głownie z "0", gdzieniegdzie występują "1". Dlatego można zastosować prostą kompresje zapisując jedynie położenie "1"domyślnie pozostałe miejsca w kodzie są "0". Jednak

w dobie szybko rozwijającej się techniki i ciągłego postępu pamięć dyskowa jest raczej tania i nie powinno się przejmować obszernością zaproponowanego rozwiązania. Zwłaszcza, że baza danych dla jednego wzorca przechowuje jedynie naiwny kod, a nie jego macierz kodów, macierze są tworzone jedynie dla próbek i są przechowywane przez pamięć operacyjną w trakcie porównywania.

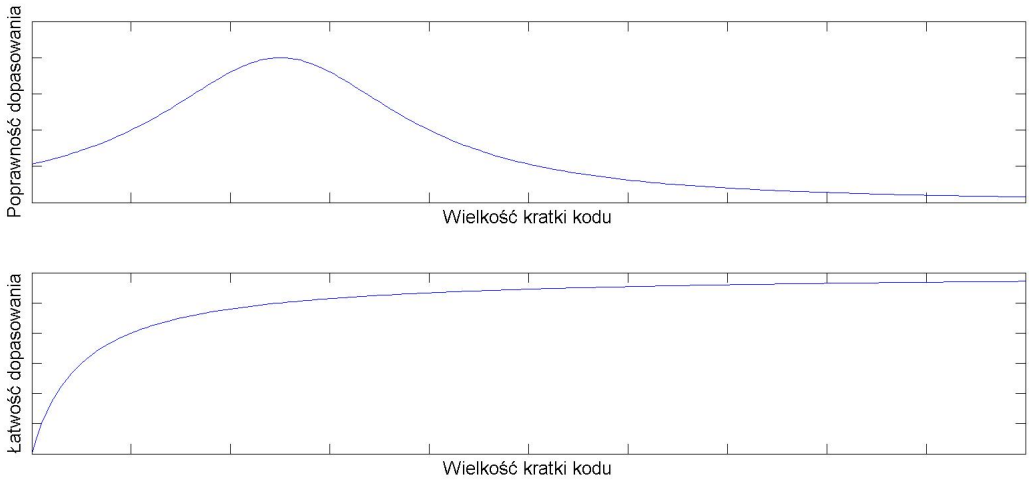
Przykład 2[3]

Baza danych 1 mln wzorców zajmuje 171,66 MB. Gdyby zaszła potrzeba przechowywania macierzy kodów zamiast naiwnego kodu baza zajmowałaby 419,09 GB.

Przykład 3[3]

Algorytm stwierdzając podobieństwo porównuje naiwny kod wzorca z macierzą kodów próbki, porównując każdy element wzorca z odpowiadającym elementem próbki w każdym jej segmencie. W trakcie całego procesu algorytm wykonuje 3600000 porównań.

Jak widać duży rozmiar kodu nie wpływa znacząco na zdolność magazynowania go w bazach danych, natomiast istotny jest w procesie porównywania. Wraz ze wzrostem skomplikowania funkcji kodującej wzrósł stopień skomplikowania funkcji porównującej kody. Utrudnienie polega na porównywaniu kodu wzorca z każdym segmentem kodu próbki. Porównanie to podobnie jak w początkowej wersji algorytmu może być wykonane funkcją XOR. Zmiana polega na tym, że algorytm spośród wszystkich porównań musi wybrać takie dla którego wystąpiło najwięcej dopasowań (1 : 1). Poniżej przedstawiono schematycznie wpływ wielkości kratki kodu na poprawność i łatwość dopasowania kodów.



Rysunek 2.4. Wpływ wielkości kratki kodu

[3] Wyniki dotyczą danych podanych w przykładzie 1

2.2. Przyczyny zastosowań kodu odcisku

W obecnie istniejących systemach biometrycznych wykorzystujących odcisk palca przeważają metody minucyjne. Jako sposób porównywania obrazów stosowane jest podejście obiektowe. Ideą tej pracy jest sprawdzenie możliwości zastosowanie prostego kodowania i odejścia od obiektowego sposobu porównywania odcisków. Praca stara się sprawdzić czy utrwalony model obiektowy o zmiennej liczbie obiektów można zastąpić kodem o stałej długości. Weryfikację odcisków można przeprowadzić za pomocą prostej funkcji porównującej. Gdyby rozwiązanie takie wprowadzono w życie, silnik liczący mógłby zostać przeniesiony na karty i urządzenia mobilne o ograniczonej mocy obliczeniowej. Weryfikacja mogła by przebiegać bezpośrednio na kartach przenoszonych przez użytkowników. Dodatkowym atutem metody jest pośrednie kodowanie informacji biometrycznej. Niepożądane zdobycie kodu odcisku powinno uniemożliwić odtworzenie obrazu lub listy minucji, bez posiadania wiedzy o algorytmie kodującym. Algorytm niekoniecznie musi zapisywać położenie minucji w tak prosty sposób jak w przedstawionym powyżej rozwiązaniu. Zawsze możliwe jest dołożenie kilku fałszywych minucji do kodu i zapamietaniu ich położenia. Kolejnym możliwym sposobem zabezpieczenia jest odrzucanie wyników zbyt zgodnych. Osoba posiadająca skradziony kod odcisku nie będzie miała możliwości weryfikacji używając skradzionego kodu wzorca. System z łatwością stwierdzi 100% zgodność i z będzie mógł odrzucić próbę fałszerstwa. Dowolne modyfikacje na kodzie w celu wprowadzenia zakłóceń do skradzionego kodu powinno uniemożliwić poprawną weryfikację. Niestety metoda nie jest pozbawiona wad. Szczegółowe wyniki działania przedstawiono w rozdziale 4.

3. Porównanie wyników algorytmu z komercyjnym rozwiązaniem

Systemy biometryczne powinny cechować się dużą skutecznością porównań odcisków, a czas potrzebny na porównanie odcisków powinien być jak najszybszy. Dlatego systemy należy oceniać biorąc pod uwagę właśnie te kryteria. Jakość obu cech jest bardzo istotna. Szybko działający system, dla którego EER[1] jest wysoki jest tak samo nieskuteczny jak bezbłędny system, który obliczenia prowadzi zbyt długo. Ten ostatni można wykorzystać jako wsparcie dla daktyloskopii[2]. Ocena rozwiązania podawana jest na tle komercyjnego programu Fingers Identification Technology[3]. Dla potrzeb tej pracy użyto 30 dniowej wersji demonstracyjnej. W zależności od metody porównującej zmieniany jest sposób dopasowywania minucji. Domyślnie SDK posiada swoje wskaźniki obliczające stopień dopasowania, aby porównywanie było miarodajne wykorzystuje ten program jedynie do ekstrakcji minucji i zwrócenia liczby dopasowanych i niedopasowanych minucji.

3.1. Analiza statystyczna kodu

Typowym sposobem oceny biometrycznych systemów używających kodowania jako sposobu przetwarzania informacji jest badanie właściwości statystycznych kodu. Dlatego też przeprowadzono porównania w dwóch grupach. Pierwsza z nich dotyczy porównania między odciskami pochodzącymi od tego samego palca. Druga grupa to porównania między różnymi odciskami. W przypadku analizy algorytmu wykonano 2800 porównań. Porównywania wykonywano na zbiorze 800 odcisków, pochodzących od 100 różnych palców, gdzie każdy palec posiada 8 pobranych od niego odcisków. Wybrano dużą liczbe odcisków aby możliwie najlepiej sprawdzic działanie algorytmu. Test wykonywany poprzez porównywanie każdy z każdym wewnątrz odcisków pochodzących od tego samego palca. Druga grupa dotyczy porównania między odciskami pochodzącymi od różnych palców. W przypadku analizy algorytmu wykonano 4950 porównań. Test wykonano na zbiorze 100 odcisków. W tym porównaniu zmniejszono liczbę odcisków, ponieważ nie porównywano odcisków pochodzacych od tego samego palca. Liczba 100 odcisków wystarczyła do wykonania zadowalającej liczby porównań. Jednocześnie mniejsza liczba odcisków jest niewiarygodna dla takiego testu. Porównania wykonano poprzez porównanie każdy - z każdym wewnątrz odcisków pochodzących od różnych palców. Analogiczne testy wykonano dla SDK Neurotechnology. Porównując te same odciski wykonano 300 porównań dla odcisków pochodzących od tego samego

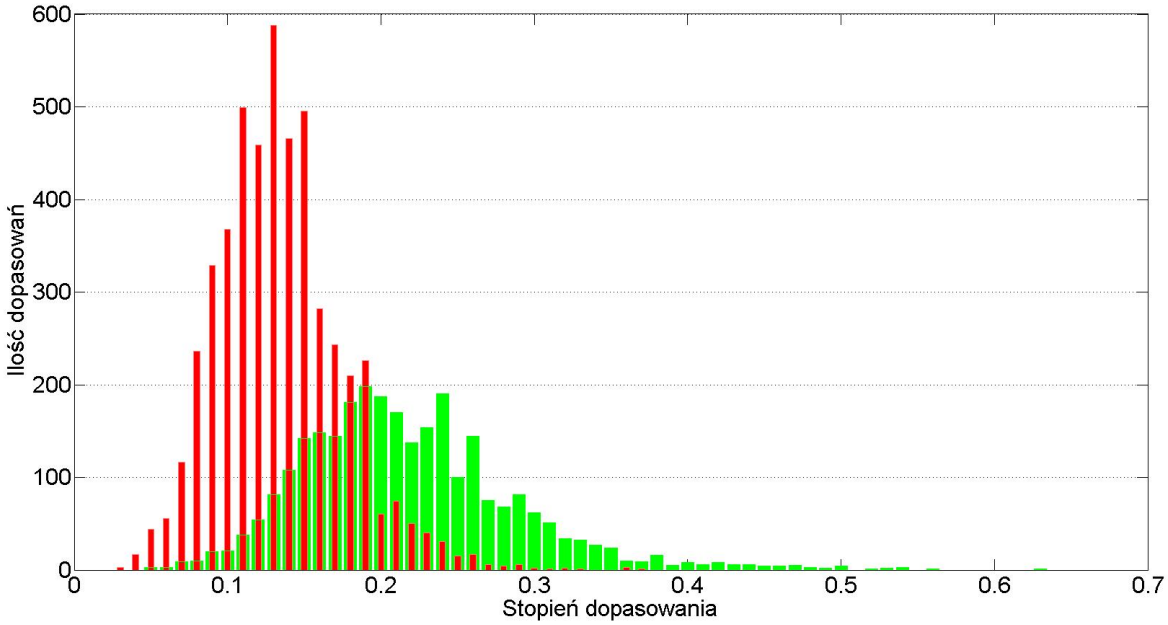
[1] *ang. Error equality rate, błąd zrównoważony - błąd, dla progu, dla którego błąd fałszywych odrzuceń i błąd fałszywych dopasowań jest taki sam*

[2] z gr. daktylos – palec i skopeo – patrzę, oglądam) – technika śledcza zajmująca się badaniami porównawczymi linii papilarnych

[3] SDK do porównywania odcisków na licencji Neurotechnology

palca i 300 dla odcisków pochodzących od różnych. Mała liczba porównań świadczy o tym iż uznano komercyjny program za dość powtarzalny, a testy wykonano dla zgromadzenia wyników na wspólnej bazie odcisków.

3.1.1. Rezultaty testów



Rysunek 3.1. Ilość dopasowań w zależności od jakości dopasowania do odcisku wzorca (test Algorytmu kodowego)

Rysunek3.1 przedstawia porównania wewnątrz tych samych i różnych odcisków. Kolorem zielonym zaznaczono porównania w grupie tych samych odcisków, czerwonym wewnątrz różnych. Na osi odciętych zaznaczono stopień dopasowania. Jest to iloraz ilości porównań 1:1 do ilości minucji we wzorcu. Świadomie nie stosowano tu klasycznego podejścia odległości Hamminga[4], ponieważ utworzony kod odcisku ma głównie "0", "1ćzyli minucje występują w nim rzadko. Zastosowanie odległości Hamminga dawało by wysoką zgodność kodów dla porównywania w obu grupach i wynik byłby nieczytelny. Na osi rzędnych zaznaczono ilość takich dopasowań. Dla czytelności wykres 3.2 przedstawia liniowy sposób reprezentacji poprzedniego wyniku. W dalszej części przedstawiono tylko rysunki liniowe. Wyraźnie widać iż rejony kreślone przez funkcje porównań tych samych i różnych odcisków pokrywają się(miejscie przecięcia zostało zaznaczone niebieską linią). Punk ten wykorzystywany jest do obliczania wskaźników FAR i FRR. Oznacza to, że metoda nie jest pozbawiona błędu. Istnieją odciski które zostaną błędnie zaklasyfikowane jako poprawne oraz te które zostaną błędnie odrzucone. EER dla takiej metody wynosi około 17%. Oznacza to że przy wyborze progu zaznaczonego niebieską linią

[4] ang. Hamming distance – w teorii informacji jest to wprowadzona przez Richarda Hamminga miara odmienności dwóch ciągów o takiej samej długości

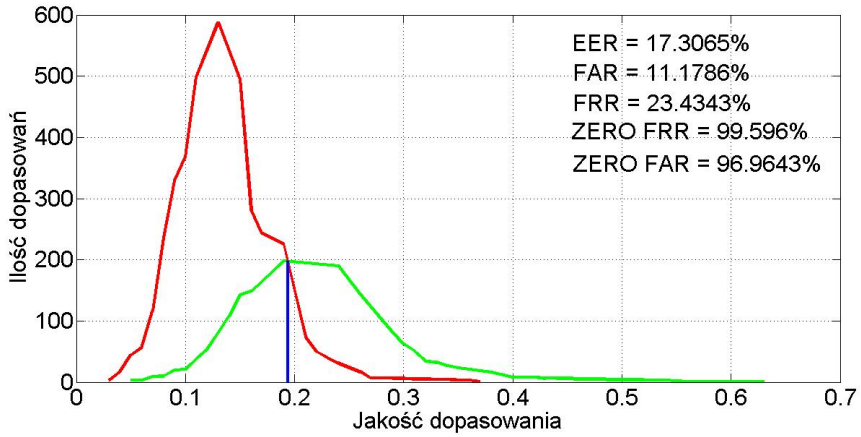
co piąte porównanie zostanie błędnie sklasyfikowane. Innowacyjność tej metody porównywania odcisków wskazywałaby na to, że metoda nie koniecznie powinna działać rewelacyjnie. Wynik w granicach 17% jest jednak wynikiem niedopuszczalnym i przekreśla jakąkolwiek użyteczność tej metody. Dla przedstawionego progu FAR[5] wynosi nieco ponad 11%, wiec ok. co dziesiąty odcisk oszusta zostanie zaakceptowany jako prawdziwy. W ustalonym progu FRR[6] wynosi ponad 23%, oznacza to, że prawie co czwarty prawidłowy odcisk będzie odrzucony. Błąd ten może nie prowadzi do akceptacji próby włamania ale może być uciążliwy dla właściwych użytkowników. Wyobraźmy sobie sytuacje, że ten system zainstalowano w jednym z bankomatów. Więc średnio co 4 wypłata pieniędzy będzie niemożliwa bo system odrzuci nasze logowanie i cały proces należy powtórzyć. Sytuacja wygląda znacznie gorzej gdy system ma zapewniać zerowy błąd akceptacji, wtedy błąd fałszywego odrzucenia wynosi ponad 96%. Rysunek 3.3 pokazuje te same wyniki dla porównywania na tle próbki. Podobnie jak poprzednio wynik do ilość porównań 1:1 do ilości minucji w próbce. Współczynnik EER wzrósł a współczynniki ZERO FRR i ZERO FARi różnią nieznacznie. Większe różnice występują na poziomie FAR i FRR. Wynika to ze sposobu porównywania odcisków przez algorytm. W skutek dopasowywania kodów algorytm dokręca obrazy odcisków tak aby liczba porównań 0:1 I 1:0 była najmniejsza. Skutkiem ubocznym jest zmniejszanie ilości rozpatrywanych minucji po stronie próbki, dzięki temu liczba rozpatrywanych minucji w próbce jest generalnie mniejsza od liczby minucji we wzorcu. Zmniejszenie ilości minucji po stronie próbki zmniejsza FRR i zwiększa FAR. Dzieje się tak, ponieważ ta sama liczba dopasowanych minucji dzielona przez mniejsza liczbę rozpatrywanych minucji w próbce powoduje przesuwanie wykresu ku większym wartościom. Efekt ten dotyczy zarówno porównań wśród tych samych jak i różnych odcisków, a ponieważ porównania dla odcisków różnych mają mniejszą liczbę trafień niż dla tych samych wykres wykres ten szybciej przesunął się w prawą stronę, a wraz z nim próg EER. Zjawisko to nie występuje dla testów aplikacji $Neurotechnology$ wynika to z innego algorytmu dopasowującego minucje.

Wyniki przedstawione na tle odcisków próbki są nieznacznie gorsze od wyników przedstawianych na tle wzorca. Wynika to z tego, że algorytm szukając najlepszego dopasowania stosuje transformacje na obrazie próbki, w zależności od dokonanego przekształcenia liczba minucji biorących udział w porównaniu się zmienia mniejsza liczba niedopasowanych minucji może być spowodowana zmniejszeniem się liczby rozpatrywanych minucji próbki. Jest to negatywny skutek uboczny minimalizacji porównań 1:0 i 0:1.

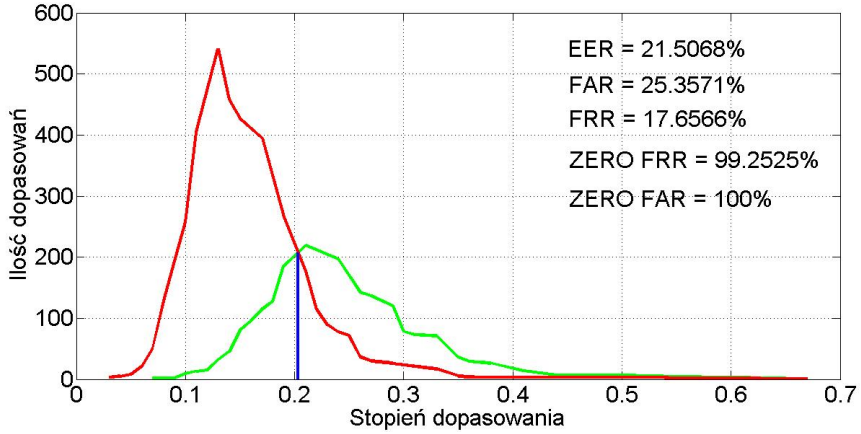
Komercyjne rozwiązanie wykorzystano w taki sposób aby algorytm porównujący miał zbliżone działanie do przedstawianego kodowania. Wykorzystano jedynie liczbę minucji dopasowanych i nie dopasowanych, generując takie same wykresy. Kolory na wykresach są takie same jak dla testów kodu. Zastosowane metoda nie sugeruje się wskaźnikiem dopasowania zwracanym przez program $Neurotechnology$. Algorytm komercyjny nie korzysta z kodowania odcisku. Wszelkie dopasowania odbywają się przez porównywanie obiektów, a nie kodów. Algorytm komercyjny

[5] ang. False Acceptance Rate -błąd fałszywej akceptacji, ilość odcisków błędnie sklasyfikowanych jako prawdziwe do całości porównywanych odcisków przez system

[6] ang. False Rejection Rate -błąd fałszywego odrzucenia, ilość odcisków błędnie sklasyfikowanych jako nieprawdziwe do całości porównywanych odcisków przez system



Rysunek 3.2. Ilość dopasowań w zależności od jakości dopasowania do odcisku wzorca (test Algorytmu kodowego)

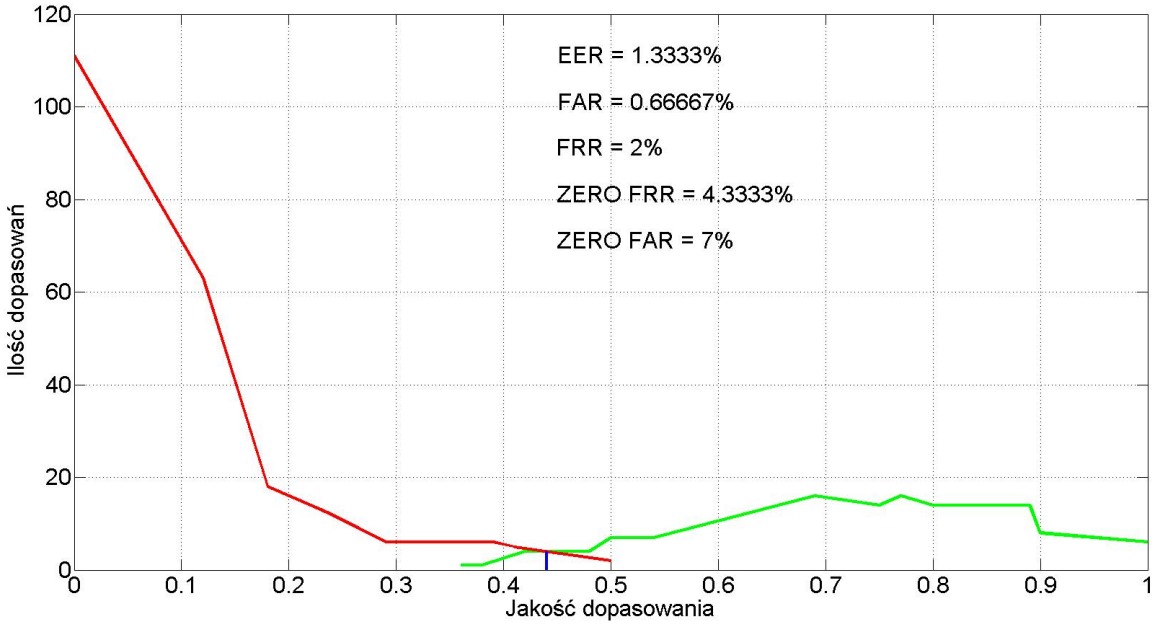


Rysunek 3.3. Ilość dopasowań w zależności od jakości dopasowania do odcisku próbki (test Algorytmu kodowego)

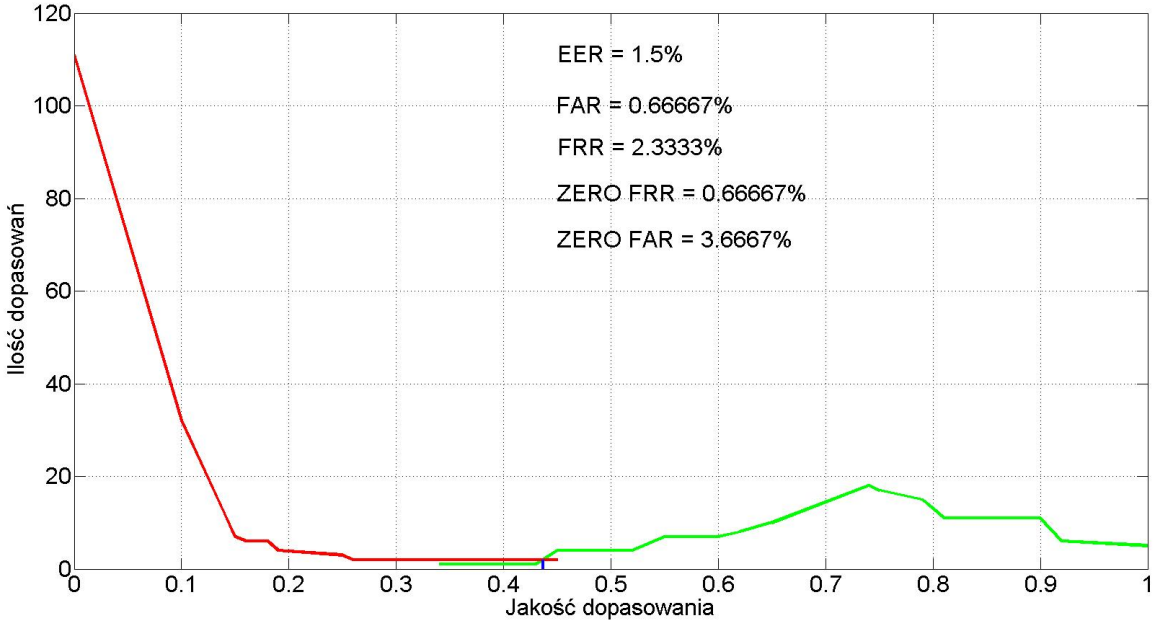
ma przewagę nad kodowym rozwiązaniem, przechowuje informacje o typie minucji oraz buduje siatkę łącząc dopasowywane minucje. Jest to kolejna informacja, dodatkowo stwierdzająca zgodność dopasowania. Na rysunkach **??** i 3.4 zaprezentowano analogiczne do poprzedniego testu wykresy jakości dopasowania względem odcisku wzorca. Rysunki **??** oraz 3.5 prezentują te same wyniki dla próbki.

Dla *Neurotechnology* wynik również nie jest pozbawiony błędu, choć jest o wiele lepszy niż w przypadku zastosowanego algorytmu kodującego. EER dla tej metody wynosi ok 1.5%, mimo iż jest 10 krotnie mniejszy niż dla kodu odcisku, nie zapewnia należytej ochrony wymaganej od systemów biometrycznych. Warto zauważyć, iż podawany wynik nie dotyczy skuteczności oprogramowania *Neurotechnology*, a jedynie jego udziału w proponowanym sposobie porównywania odcisków. Zdecydowanie lepiej wypadają też współczynniki FAR i FRR, które wynoszą odpowiednio ok 1% i 2%. Natomiast znacznie lepiej prezentuje się w przypadku błędów zerowych.

Wystarczy zaakceptować błąd w granicach 7% i system stanie się bezbłędny. Oczywiście wyniki dotyczą tylko bazy danych na której był przeprowadzany test. Przeprowadzany test dowodzi, iż kodowa metoda porównywania odcisków dopasowuje odciski znacznie gorzej niż metoda obiektowa. Dodatkowo porównywanie jedynie liczby minucji pasujących nie wystarcza do jednoznacznego stwierdzenia podobieństwa bądź niepodobieństwa obrazów. Kolejną wadą przedstawianego rozwiązania jest brak porównań dla których jest 100% zgodność lub analogiczna niezgodność. Wynika to z doboru parametrów, a zwłaszcza wielkości kratki kodu. Zmniejszenie kratki powodowało pojawianie się porównań o 100% niezgodności natomiast zwiększanie sprzyja łatwiejszemu dopasowywaniu kodów. Niestety ustawienia te źle wpływały na ilość poprawnych porównań. Ustawienie optymalnej wielkości kratki powoduje zmniejszenie liczby całkowitych zgodności i niezgodności.



Rysunek 3.4. Ilość dopasowań w zależności od jakości dopasowania do odcisku wzorca (test *Neurotechnology*)



Rysunek 3.5. Ilość dopasowań w zależności od jakości dopasowania do odcisku próbki (test *Neurotechnology*)

3.2. Rozkład punktów porównania

Przeprowadzone testy dowodzą jednoznacznie, że sposób porównywania tylko liczby zgodnych minucji nie wystarcza do stwierdzenia poprawnego podobieństwa lub nie podobieństwa obrazów. Choć metoda ta jest jedną z głównych metod stwierdzających prawidłowe dopasowanie odcisków to dla wykorzystań opisywanego algorytmu jest całkowicie zawodna. Przyczyn może być wiele

- Błąd algorytmu
- Błąd implementacji
- Niewystarczająca ilość kodowanej informacji
- Zbytnie uproszczenie kodowanej informacji
- Brak filtrowania minucji, chodzi tu o odrzucanie minucji o niewystarczającej jakości
- Błąd sposobu porównania kodów

Przyczyny związane z błędami w kodzie programu odrzucam, gdyż testy aplikacji dowodzą prawidłową realizację wymyślonego algorytmu. Błąd algorytmu stawiałby poważną barierę w rozwoju tego typu podejścia dlatego dla celów badawczych również jest tu pomijany. Zwiększenie ilości kodowanej informacji mogłoby przekształcić algorytm kodowania w inny zapis algorytmu obiektowego. Naturalne wydaje się iż w algorytmach tworzących kod jakiejś próbki biometrycznej wartości cech biometrycznych są uśredniane i wynik ten nie powinien przeszkadzać w porównywaniu obrazów. Dowodem tego może być algorytm Dougman-a do kodowania tęczówki oka. Obraz tęczówki jest świadomie uśredniany, a mimo wszystko nie przeszkadza to w dalszej identyfikacji. trafnym pomysłem jest odrzucanie minucji o złej jakości. W procesie przetwarzania obrazu znalezione współrzędne niekoniecznie muszą wyznaczać minucje odcisku. Nikt nie gwarantuje wysokiej jakości obrazu z czytnika. Dlatego znalezione punkty mogą być tylko zakłóceniem obrazu a nie rzeczywistą minucją. Mimo tego rozwiązanie to nie zostało zastosowane. Powodem jest rożna interpretacja minucji przez różne SDK'i do ekstrakcji minucji z obrazu. Metoda musi być niezależna od sposobu zdobywania minucji dlatego nie powinno polegać się na wskazaniach SDK'a. Zawsze można zastosować inne oprogramowanie. Metoda zmiany SDK nie jest metodą naprawy błędu omawianego rozwiązania. Jedynym zatem rozważanym sposobem poprawy jest zmiana sposobu porównywania już zakodowanych odcisków. Pierwotnie wzorem istniejących metod największy nacisk położony jest na liczbę dopasowanych minucji. Dlatego proponowanym wyjściem jest zastosowanie różnicy symetrycznej czyli liczby niedopasowanych minucji po stronie wzorca i liczby niedopasowanych minucji po stronie próbki. Są to dane obecnie nie wykorzystywane w procesie porównywania. Rozwiązanie to zostało szerzej omówione w rozdziale 5. Innym sposobem jest wykorzystanie wszystkich 3 rodzajów porównań i w zależności od ich liczby decydować czy odciski należą do tej samej osoby czy do różnych. Możliwe jest to poprzez zastosowanie wag dla każdego z tych porównań przyjmując przykładowo 1 punkt za porównanie 1:1 i -1 punkt za porównania 0:1 i 1:0. Rozwiązanie to jest jednak wrażliwe na sumę minucji w próbce i we wzorcu otrzymane ilości punktów będą więc różnić się w zależności od odcisków jakie będą porównywane. Zastosowanie takiej metody wymagało by stworzenie tablicy progów zamiast jednego globalnego progu separującego odciski.

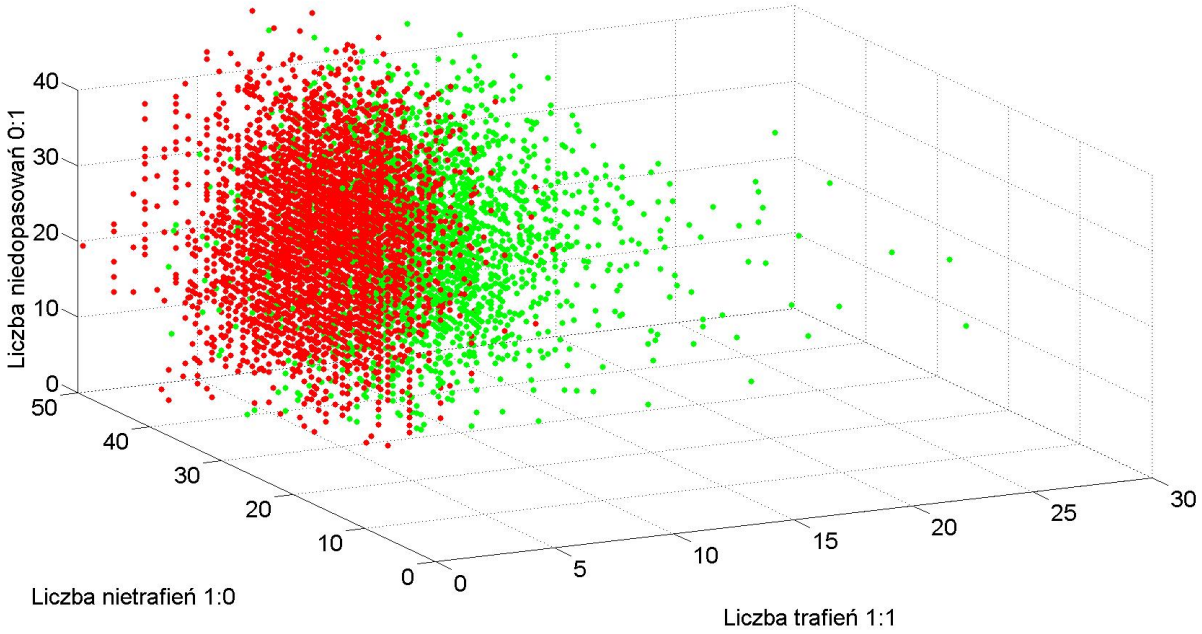
Przykład 1

Dane są cztery odciski po dwa od osoby A i B. Posiadają odpowiednio 30, 25 minucji dla osoby A oraz 12, 15 minucji osoby B. Porównując dwa odciski osoby A otrzymano 15 minucji zgodnych, porównując odciski osoby B otrzymano 8 odcisków zgodnych. Porównania między osobami A(30) i B(12) dają 6 odcisków zgodnych. Gdyby wyliczać współczynniki porównania w podany powyżej sposób dla porównania wewnątrz tych samych osób otrzymano wynik -10 i -3 punkty. Porównanie między osobami A i B daje wynik -30 pkt

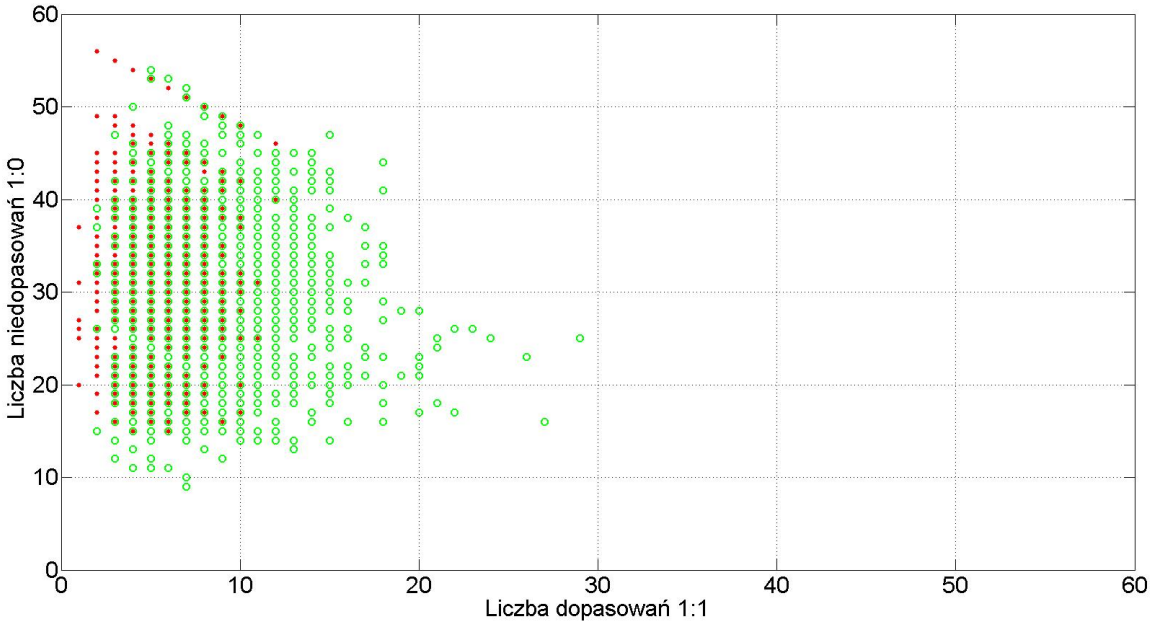
Przykład 2

Dane są dwa odciski po jednym od osoby A i B. Posiadają odpowiednio 30 i 40 minucji. Porównania między osobami A(30) i B(40) dają 15 odcisków zgodnych. Ten sam sposób wyliczeń współczynników dla porównania daje wynik -40 pkt.

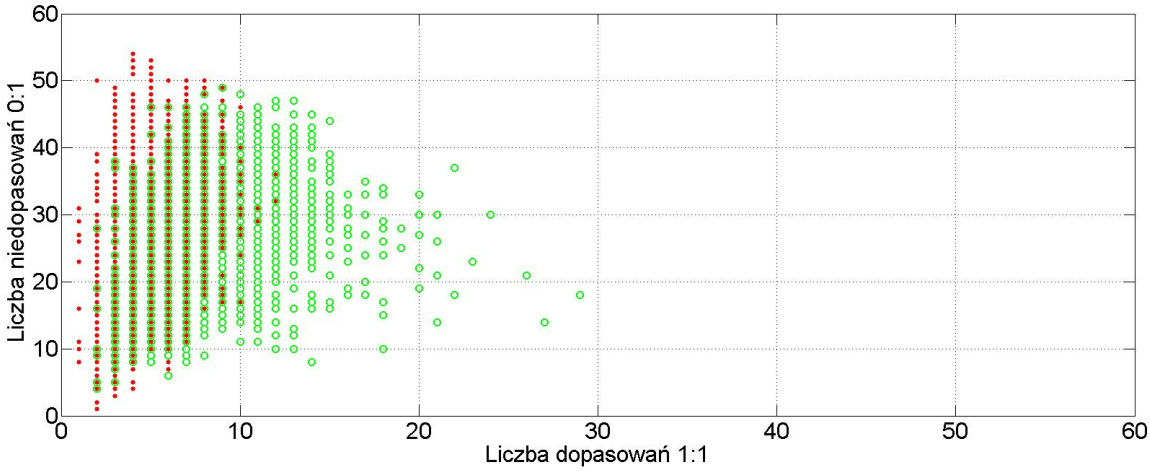
Przykład 1 prezentuje udaną próbę zastosowania zmienionego systemu porównań, jeżeli za próg separujący przyjmiemy liczbę z przedziału (-3; -30). Przykład 2 prezentuje porównanie które powinno być potraktowane jako niezgodne, natomiast dla progu z przykładu 1 zaklasyfikowane zostanie przez system jako zgodne. Te dwa proste przykłady prezentują ogromną wrażliwość przedstawionego rozwiązania. Można oczywiście zakładać iż przedstawione wyniki nie będą odzwierciedlać prawdziwej sytuacji. Jednak dla testowanej bazy danych liczba minucji dla jednego palca waha się od 14 do 51 minucji, więc gdyby dwa odciski o względnie dużej liczbie minucji i dość gęstym ustawieniu porównać testowanym algorytmem możliwe jest uzyskanie zgodności nawet większej od 10 minucji. Proste punktowanie nie uwzględnia jednak liczby minucji we wzorcu i w próbce. Dokonano doświadczenia i narysowano rozkład punktów porównań w przestrzeni przyjmując jako współrzędne XYZ punkty odpowiadające liczności porównań (1:1), (1:0) i (0:1) dla całej pary próbka wzorzec. Rozkład punktów przedstawia rysunek 3.6, natomiast rzuty na poszczególne dwuwymiarowe przestrzenie przedstawiają rysunki 3.7, 3.8, 3.9. Podobnie jak poprzednio kolorem zielonym zaznaczono punkty dla porównań między odciskami należącymi do tych samych palców, czerwonym dla różnych. Wykres potwierdza przypuszczenia o różnym rozłożeniu punktów w przestrzeni. Niestety żaden rzut na przestrzenie dwuwymiarowe nie zapewnia separacji tych punktów. Szersze zastosowanie tej metody porównania omówione zostało w rozdziale 6.



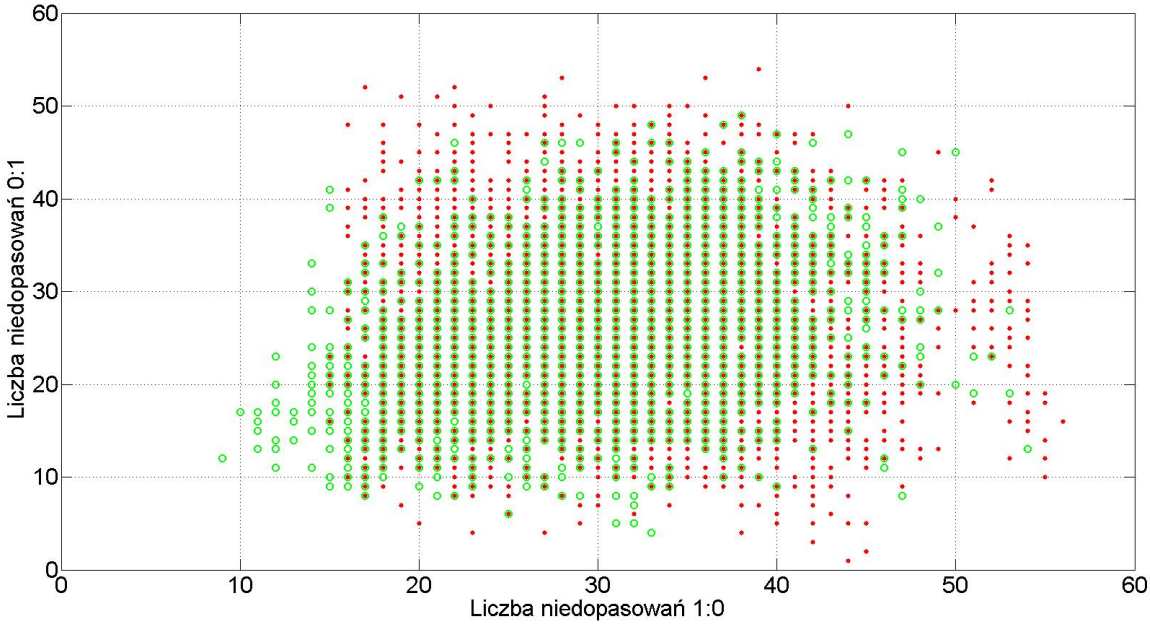
Rysunek 3.6. Rozkład punktów dopasowań dla porównań kodów odcisku



Rysunek 3.7. Rozkład punktów dopasowań dla porównań kodów odcisku na przestrzeni porównań (1:1) (1:0)



Rysunek 3.8. Rozkład punktów dopasowań dla porównań kodów odcisku na przestrzeni porównań (1:1) (0:1)



Rysunek 3.9. Rozkład punktów dopasowań dla porównań kodów odcisku na przestrzeni porównań (1:0) (0:1)

4. Różnica symetryczna jako sposób porównywania odcisków

Większość obecnie stosowanych metod porównuje odciski poprzez szukanie podobieństw między nimi. Metody nie zastanawiają się nad wpływem ilości niepodobnych minucji. Celem tej pracy jest stwierdzenie czy istnieje możliwość porównywania odcisków badając jedynie ich niepodobieństwo. Sposób ten stara się odpowiedzieć na pytanie co jest lepsze 10 zgodnych minucji, które stanowią jedynie 33% wszystkich minucji, czy wzorca, czy 4 zgodne minucje które stanowią 50% minucji wzorca. W obecnie stosowanych standardach liczba cech wspólnych wymaganych przez prawo lub procedury kryminalistyczne do ustalenia tożsamości wynosi[1]

- 8-12 Niemcy
- 17 Francja
- 15 Polska

A w przypadku minucji rzadziej występujących liczba ich może być nawet mniejsza. Natomiast nigdzie nie doszukano się maksymalnych liczb niezgodnych minucji. Jedyne dodatkowe ograniczenia wprowadzają same SDK'i. Przykładowo SDK *Neurotechnology* w standardowej konfiguracji przyjmuje minimalną liczbę minucji jako 10. Nie jest to jednak, żadna arbitralnie wyznaczana przez prawo wartość, a jej wartość można ustalać według własnych preferencji. Powstało założenie, że liczba minucji niezgodnych powinna być większa dla porównań między różnymi odciskami. Metoda podobnie jak sprawdzanie podobieństwa wrażliwa jest na liczbę minucji w porównywanych odciskach. Dlatego warto uzyskaną liczbę niezgodność podzielić przez ilość minucji w odciskach. Nie stosuje się oddzielnych statystyk dla niezgodności porównań 1:0 i 0:1. Statystyki te to nic innego jak porównywania przedstawione w rozdziale 4. Istnieje liniowa zależność między porównaniami co prezentuje wzór 4.3.

$$Q_s = \frac{C_{1-1}}{C_{pattern}} \tag{4.1}$$

$$Q_{ns} = \frac{C_{1-0}}{C_{pattern}} \tag{4.2}$$

$$Q_s = 1 - Q_{ns} \tag{4.3}$$

[1] dane zaczerpnięte ze strony http://pl.wikipedia.org/wiki/Linie_papilarne

- Q_s Jakość kodu(procent zgodności)
- Q_{ns} Jakość kodu(procent niezgodności)
- C_{1-1} Ilość porównań 1:1 dla
- C_{1-0} Ilość porównań 1:0
- $C_{pattern}$ Ilość minucji we wzorcu

- Q_{nss} Jakość kodu(procent niezgodności) dla różnicy symetrycznej
- C_{sample} Ilość rozpatrywanych minucji w próbce

4.1. Analiza statystyczna kodu

Zastosowane rozwiązanie wylicza jakość niedopasowania jako iloraz sumy ilości porównań 1:0 i 0:1 do sumy ilości minucji wzorca i rozpatrywanych minucji próbki. Rozwiązanie przedstawia wzór4.4

$$Q_nss = 1 - \frac{C_{1-1}}{(C_{pattern} + C_{sample})} \tag{4.4}$$

W odróżnieniu od poprzedniego jest liniowo niezależny od jakości kodu przedstawianych w rozdziale 4. Gdyby tak nie było zabieg taki nie miałby większego sensu. Podobnie jak w poprzednim rozdziale świadomie zastąpiono odległości Hamminga porównując jedynie charakterystyczne porównania. Zbędne tło (porównania 0:0) jest pomijane. Wyniki z komercyjnego oprogramowania przedstawiane są jako porównanie. Podobnie jak wcześniej SDK $Neurotechnology$ użyto jedynie do zdobycia liczby minucji zgodnych i niezgodnych. Liczba porównań jest taka sama jak w poprzednim rozdziale i przeprowadzona została na tej samej próbie odcisków.

4.1.1. Rezultaty testów

Kolorem czerwonym zaznaczono porównania między odciskami pochodzącymi od różnych palców, zielonym pochodzących od tych samych. Przedstawione wyniki również nie są wystarczajaco dobre, natomiast są nieco lepsze niż w przypadku porównywania podobieństwa. EER zmniejszył się i wynosi ok 17%. Nie jest to wynik zadowalający ale warto zauważyć, że zmieniono jedynie sposób porównania a nie sam algorytm kodowania. Każda poprawa wyniku jest więc sukcesem. FAR dla ustalonego progu wynosi ok 13% i w porównaniu do poprzedniej metody uległ niewielkiemu pogorszeniu(2 punkty procentowe). FRR wynosi ok 18% i uległo sporej poprawie względem poprzedniej metody (5 punktów procentowych). Błędy ZERO FAR i ZERO FRR dalej są na poziomie niedopuszczalnym i wynoszą ponad 95%. Kolejnym ciekawym zjawiskiem jest zwężenie przedziałów w jakich znajdują się wykresy. Niestety zmiana ta dotyczy zarówno porównań różnych jak i tych samych odcisków. Przedstawiona symulacja dowodzi iż sposoby porównywania niezgodności lepiej sprawdza się dla porównywania odcisków pochodzących od jednego palca. Natomiast sposób porównywania zgodności lepiej sprawdza się dla porównywania odcisków pochodzących od różnych palców. Wynik jest zaskakujący gdyż działa inaczej niż wskazywałoby na to logiczne rozumowanie.

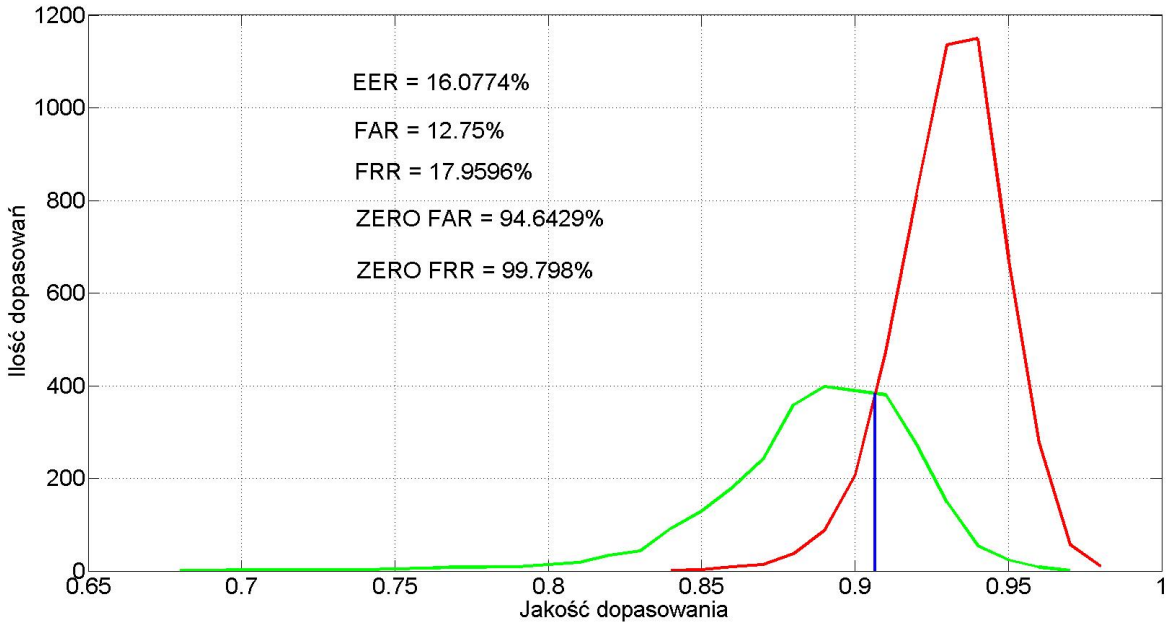
Teza 1

Stwierdzanie podobieństwa jest skuteczniejsze poprzez porównywanie niepodobieństw.

Teza 2

Stwierdzanie niepodobieństwa jest skuteczniejsze poprzez porównywanie podobieństw.

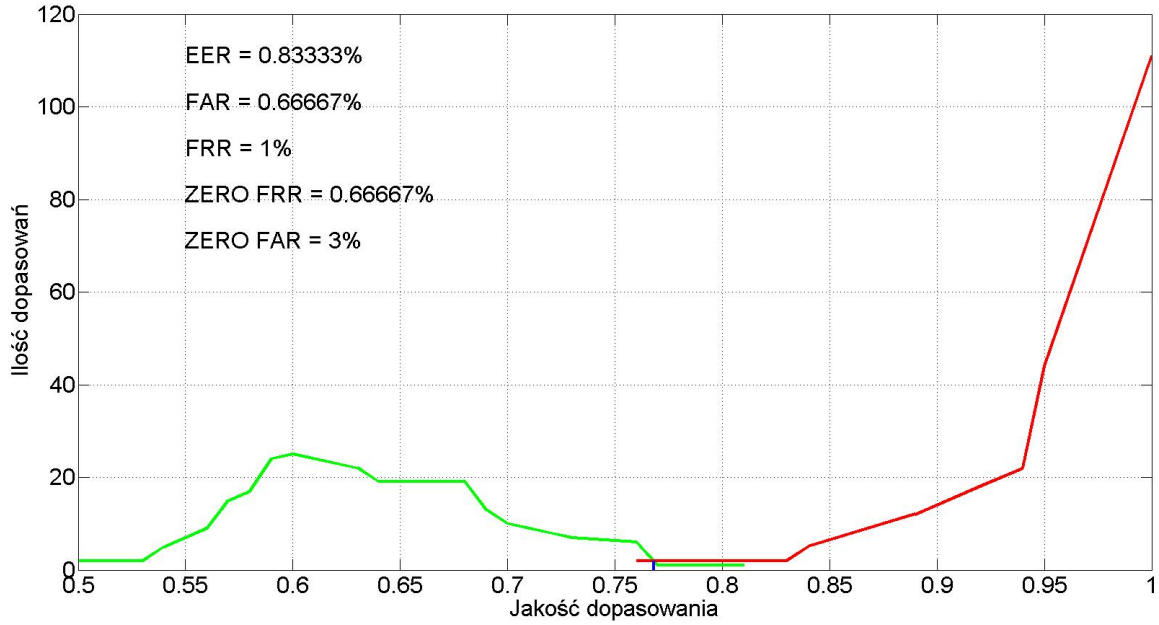
Plusem tej metody jest brak konieczności rozstrzygania wyniku dla porównań względem próbki i dla porównań względem wzorca. W poprzedniej metodzie właściwie otrzymywano dwa wskazania w zależności czy liczbę dopasowań podzielono przez liczbę minucji wzorca czy liczbę rozpatrywanych minucji próbki. W tej metodzie wyróżniono tylko jeden współczynnik. Możliwym wyjściem jest stosowanie hybrydy tych dwóch rozwiązań. Zastosować można to dla tych porównań które znajdują się blisko progu decyzyjnego. w takiej sytuacji należało by skorzystać z drugiego wskaźnika i w przypadku przeciwnego rezultatu, wybrać ten który znajduje się dalej od swego progu decyzyjnego. Być może taka hybryda łączyła by mocne strony obu sposobów porównań mogłoby to doprowadzić do kolejnego zmniejszenia błędów dopasowań odcisków.



Rysunek 4.1. Ilość niedopasowań w zależności od jakości niedopasowania do odcisków

Niestety test ten dowodzi iż niezależnie od metody porównywania dla zaproponowanego algorytmu jest nieskuteczne jeśli porównujemy tylko jeden wskaźnik takiego porównania. Niezależnie czy porównujemy podobieństwo czy niepodobieństwo błąd EER jest niedopuszczalny i nie czyni tworzonej metody systemem biometrycznym. Celowo nie sprawdzam hybrydowego rozwiązania gdyż nawet polepszenie wyniku o kilka punktów procentowych nie jest rozwiązaniem. Jedyną drogą rozwiązania problemu jest wykorzystanie wszystkich danych tworzonych przez porównywanie i badanie rozkładu punktów porównań w przestrzeni trójwymiarowej. Metoda oraz jej wyniki opisana jest w rozdziale 6.

Na wykresie 4.2 Przedstawiono tą samą metodę wykorzystując SDK *Neurotechnology* jedynie do zdobycia liczby minucji pasujących i niepasujących. W zastosowaniu algorytmu kodującego, jak również dla listy minucji z SDK'a zaobserwowano poprawę wskaźnika EER i FRR wskaźnik FAR został bez zmian. Ponieważ zmiana zachowań współczynników jest powtarzalna dla podejścia kodowego i obiektowego, dlatego tezy 1 i 2 można uznać za prawdziwe. Plusem tej obserwacji jest to iż postawione tezy nie zależą od stosowanych algorytmów. Podobnie jak w porównywaniu podobieństwa dane zebrane z komercyjnego SDK mają lepsze wyniki od podejścia kodowego. Dowodzi to iż kodowa wersja algorytmu w połączeniu z tymi metodami porównywania odcisków nie jest wystarczająco dobra do poprawnej weryfikacji.



Rysunek 4.2. Ilość niedopasowań w zależności od jakości niedopasowania do odcisków

5. Metoda 3D porównywania kodów

Metoda opiera się na porównywaniu poprzez wykorzystanie wszystkich danych o porównaniu kodów. Decyzje o weryfikacji odcisku podejmowana jest poprzez analizę podobieństwa i niepodobieństwa jednocześnie. Do dyspozycji są 3 porównania, ale ponieważ rozkład punktów porównania 1:0 i porównania 0:1 jest prawie taki sam podjęto decyzję o pominięciu tego porównania w procesie weryfikacji. Podobnie jak w poprzednich rozdziałach sposób działania będzie porównywany z wynikami otrzymanymi od SDK $Neurotechnology$. Ten sposób porównania ma służyć nie tylko polepszeniu ogólnego wyniku algorytmu ale też i sposobu porównywania bazującego tylko na liczbie minucji dopasowanych i niedopasowanych. Niemożliwe jest jednak przedstawianie błędów w takiej samej konwencji jak w poprzednich rozdziałach. Wynika to z tego iż próg decyzyjny nie dobierany jest w taki sposób aby równoważyć błąd FAR i FRR, a poprzez tworzenie funkcji separującej zbiory punktów od siebie. Jest to o tyle kłopotliwe iż nie mam możliwości manipulowania progiem decyzyjnym tak samo jak w poprzednich rozdziałach. Przedstawiane wyniki to wartości błędnych decyzji dopasowania i niedopasowania. Mimo iż rozpatrujemy rozkład punktów na płaszczyźnie metoda została nazwana porównaniem 3D, gdyż do weryfikacji używa trzech rodzajów porównań.

5.1. Opis metody

Testy metody wykonano na tej samej grupie porównań co testy opierające się na badaniu podobieństwa i nie podobieństwa. Zbiór ten podzielono na dwie części, część uczącą i część testującą. Część ucząca odpowiada za utworzenie funkcji separującej oba zbiory porównań, część testująca wykorzystana jest do testów poprawnej weryfikacji dla utworzonej funkcji separującej. Testy prowadzone są w trzech wariantach

- Rozkład porównań 1:1 do sumy porównań 1:0 i 0:1
- Rozkład porównań 1:1 do 1:0
- Rozkład porównań 1:1 do 0:1

Dla potrzeb tej pracy wybrano metodę Maszyny wektorów nośnych[1] oraz skorzystano z gotowych implementacji tej metody w środowisku MATLAB. W trakcie testów klasyfikator tworzy hiperpłaszczyzną separująca zbiory punktów, a następnie testuje wyniki dla zbioru testującego. Dopuszczalne są przypadki nie w pełni separowane. Oznacza to, że metoda ta może wnosić błąd już dla swojego zbioru uczącego. Kształt hiperpłaszczyzny zależy nie tylko od rozkładu punktów ale też od używanego jądra. Testy przeprowadzono dla czterech wariantów

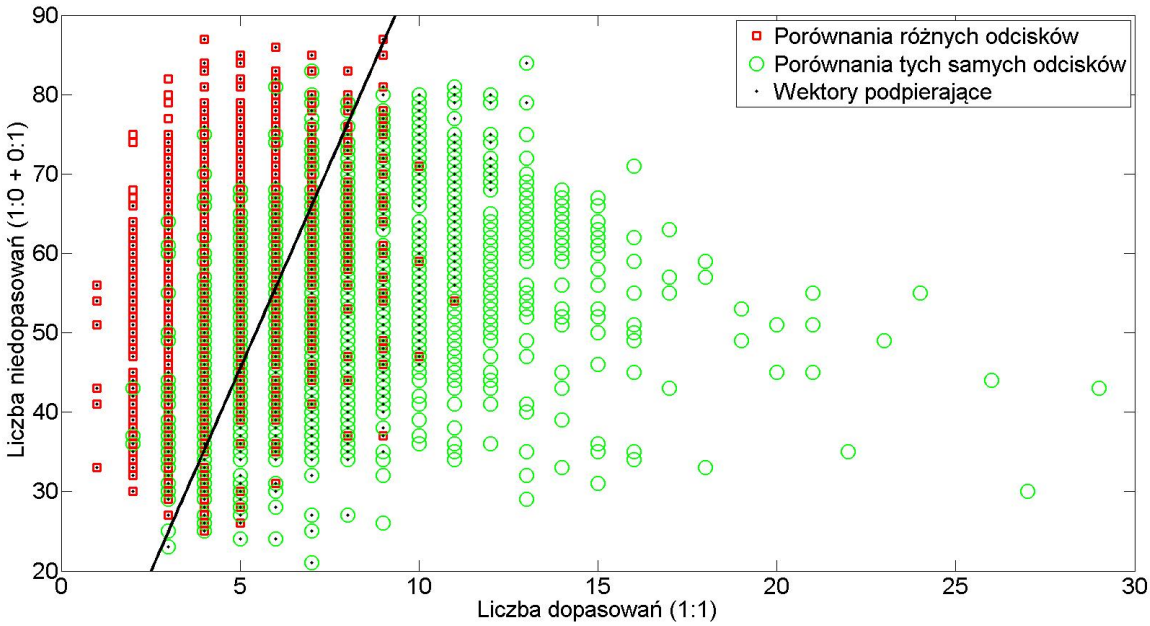
- Liniowe

[1] SVM z ang. support vector machine - klasyfikator, którego nauka ma na celu wyznaczenie hiperpłaszczyzny rozdzielającej z maksymalnym marginesem przykłady należące do dwóch klas.

- Kwadratowe
- Wielomianowe
- Normalne(Gaussowskie)

Zbiór uczący oraz zbiór porównań jest równo liczny choć liczba punktów uczących oraz liczba punktów na których przeprowadzane są testy może się różnić. Wynika to z tego iż rozkład liczby punktów porównań nie jest unikalny i powtarza się dla różnych porównań odcisków. Istnieją różne odciski o dokładnie o takiej samie liczbie minucji zgodnych i niezgodnych.

5.2. Wyniki



Rysunek 5.1. Separacja punktów dla jądra liniowego

Wykres 5.1 przedstawia liniową separacje punktów. Konsekwentnie zielonym kolorem zaznaczone są porównania między odciskami pochodzącymi od tego samego palca, czerwonym porównania między odciskami pochodzącymi od różnych palców. Doskonale widać iż nie możliwe jest liniowe separowanie punktów. Metoda już dla zbioru testowego wniesie pewien błąd. Nie jest to jednak ogromny problem. Możliwe jest bowiem stworzenie kilku równań prostych które w stu procentach rozdzielą oba zbiory. Niestety nie wpływa to na skuteczność metody, zbiór uczący powinien być w miarę uniwersalny, a funkcja powinna tylko uogólniać zachowanie punktów na płaszczyźnie.

Tabele 5.1, 5.5, 5.6 przedstawiają wyniki dla algorytmu kodującego. Widać, że pierwotne przypuszczenia rokujące poprawę jakości dopasowania były błędne. wyniki porównań znacznie się pogorszyły. Może to wynikać z niemożności znalezienia

Wariant	1:1 do 1:0 + 0:1		
Zbiór testujący	ERRORS	FRR	FAR
Rodzaj jądra			
Liniowe	28.6722%	26.6393%	31.405%
Kwadratowe	28.6722%	25.6148%	32.7824%
Wielomianowe st. 3	28.9072%	29.3033%	28.3747%
Wielomianowe st. 7	31.2573%	29.3033%	33.8843%
Wielomianowe st. 12	29.3772%	32.7869%	24.7934%
Normalne(Gaussowskie)	28.6722%	24.7951%	33.8843%

Tablica 5.1. Wyniki testów algorytmu kodowego metoda 3D porównywania kodów

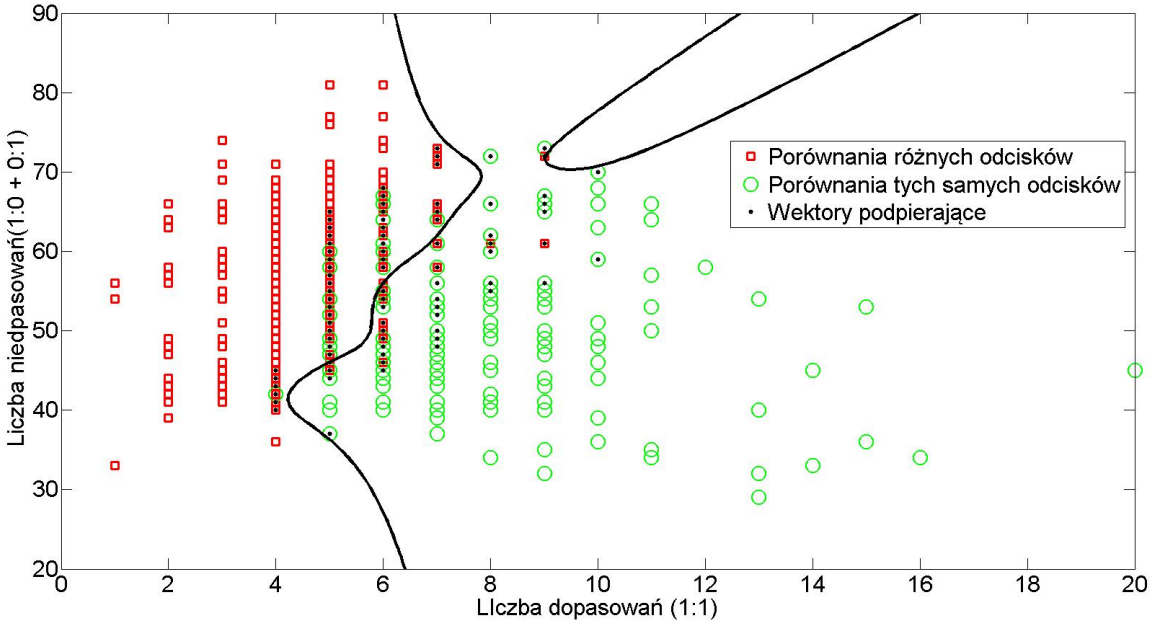
Wariant	1:1 do 1:0		
Zbiór testujący	ERRORS	FRR	FAR
Rodzaj jądra			
Liniowe	29.6703%	13.6232%	57.2139%
Kwadratowe	29.8535%	12.4638%	59.7015%
Wielomianowe st. 3	29.3040%	7.8261%	66.1692%
Wielomianowe st. 7	30.2198%	8.9855%	66.6667%
Wielomianowe st. 12	36.8132%	35.3623%	39.3035%
Normalne(Gaussowskie)	31.1355%	3.7681%	78.1095%

Tablica 5.2. Wyniki testów algorytmu kodowego metoda 3D porównywania kodów

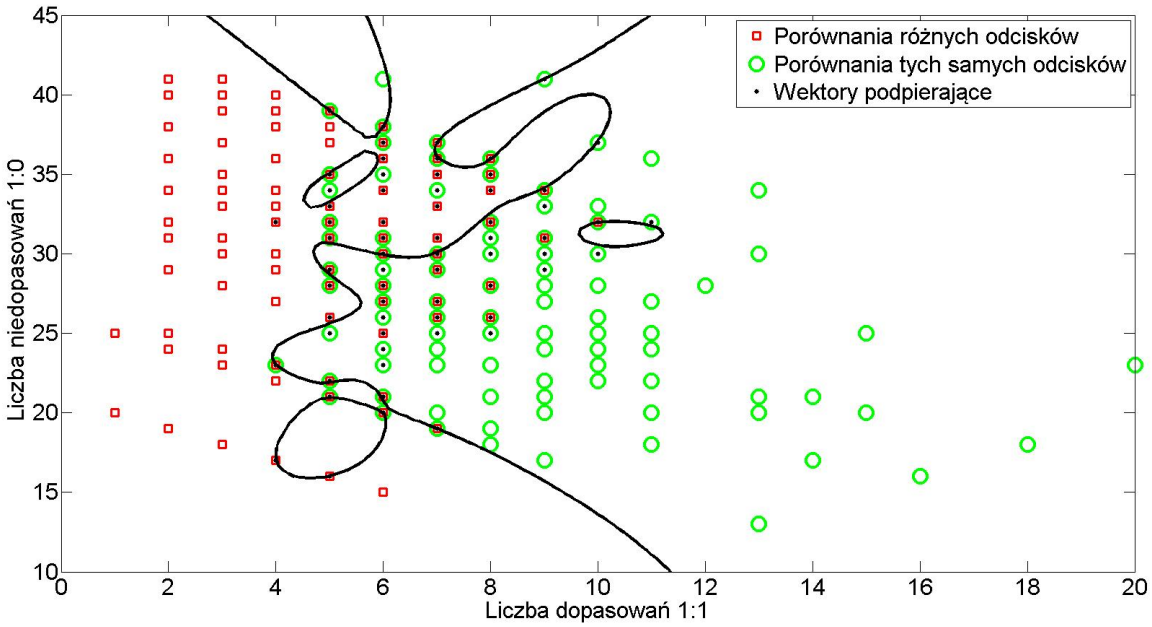
Wariant	1:1 do 0:1		
Zbiór testujący	ERRORS	FRR	FAR
Rodzaj jądra			
Liniowe	31.2024%	30.2778%	32.3232%
Kwadratowe	30.5936%	31.6667%	29.2929%
Wielomianowe st. 3	31.3546%	29.7222%	33.3333%
Wielomianowe st. 7	34.2466%	43.0556%	23.5690%
Wielomianowe st. 12	32.5723%	36.3889%	27.9461%
Normalne(Gaussowskie)	31.2024%	29.7222%	32.9966%

Tablica 5.3. Wyniki testów algorytmu kodowego metoda 3D porównywania kodów

krzywej która w pełni separuje grupy punktów. Dodatkowo żaden z wariantów porównywania nie wyróżnia się na tle innych. Ilość błędnych decyzji jest podobna dla wszystkich wariantów i waha się od 28% do 35%. Jedyną wyraźną poprawę widać dla wskaźnika FRR dla wariantu porównań 1:1 do 1:0, gdzie dla jądra gaussowskiego wskaźnik ten wynosi ok 3% , gdzie dla poprzednich metod wahał się od 17% do 23%. Niestety FAR dla tej metody wynosi ponad 78% co oznacza iż skuteczniejszy byłby tu klasyfikator losowy. Poprawę wskaźnika FRR można motywować tym iż klasyfikator SVM uczył się tylko na wynikach wzorca, gdyż oba porównania 1:1 i 1:0 dotyczą minucji wzorca. Niestety analogiczna sytuacja dla próbki gdzie klasyfikator uczył się na porównaniach 1:1 i 0:1 nie wykazała poprawy dla wskaźnika FAR. Gdyby tak się stało sensowne było by uczenie się na klasyfikatora SVM nie w przestrzenie dwu a trzy-wymiarowej. Dodatkowo większe skomplikowanie funkcji separującej nie przyczynia się do wzrostu tylko do spadku jakości porównania. Ponieważ nie do się idealnie rozsmarować punktów najlepiej spisują się metody o

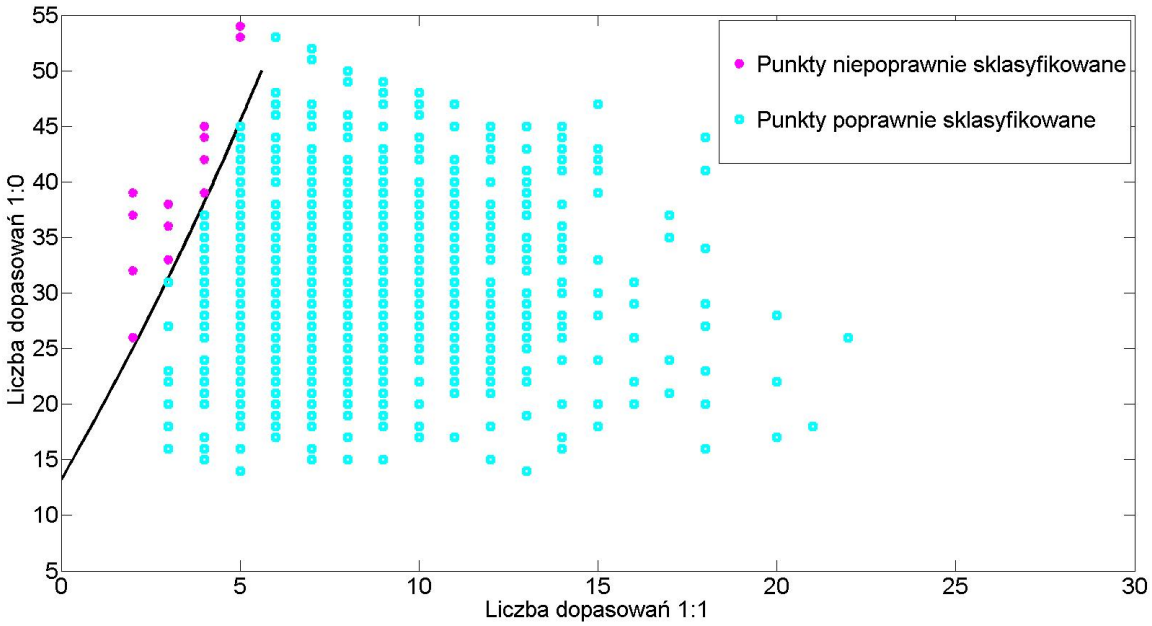


Rysunek 5.2. Separacja punktów dla jądra wielomianowego stopnia 7



Rysunek 5.3. Zły wpływ zbyt dużego stopnia jądra wielomianowego(12)

liniowym kształcie hiperprzestrzeni metody SVM. Zbyt zawiła funkcja prowadzi do bezsensownych rozgraniczeń takich ja pokazanych na rysunku 5.4.



Rysunek 5.4. Przykład klasyfikacji punktów algorytmu kodującego dla wariantu 1:1 do 1:0 jądro gaussowskie

	Wariant	1:1 do 1:0 + 0:1		
	Zbiór testujący	ERRORS	FRR	FAR
Rodzaj jądra				
Liniowe		0.4975%	0.0%	1.3889%
Kwadratowe		0.4975%	0.0%	1.3889%
Wielomianowe st. 3		0.4975%	0%	1.3889%
Wielomianowe st. 7		0.9950%	0.7752%	1.3889%
Wielomianowe st. 12		0.9950%	0.7752%	1.3889%
Normalne(Gaussowskie)		0.4975%	0%	1.3889%

Tablica 5.4. Wyniki testów algorytmu kodowego metoda 3D porównywania kodów

	Wariant	1:1 do 1:0		
	Zbiór testujący	ERRORS	FRR	FAR
Rodzaj jądra				
Liniowe		2.5641%	2.0%	5.8824%
Kwadratowe		1.7094%	1.0%	5.8824%
Wielomianowe st. 3		1.7094%	1.0%	5.8824%%
Wielomianowe st. 7		2.5641%	2.0%	5.8824%
Wielomianowe st. 12		6.8376%	7.0%	5.8824%
Normalne(Gaussowskie)		11.1111%	0%	76.4706%

Tablica 5.5. Wyniki testów algorytmu kodowego metoda 3D porównywania kodów

Dla SDK *Neurotechnology* wyniki generalnie uległy poprawie wahając się pierwotnie w granicach 0.8% - 1.5% dla porównywania oddzielnie podobieństwa i niepodobieństwa do najlepszych wyników ok 0.5% dla porównywania 3D. Dla SDK

Wariant	1:1 do 0:1		
Zbiór testujący	ERRORS	FRR	FAR
Rodzaj jądra			
Liniowe	1.0638%	0.9091%	1.2821%
Kwadratowe	0.5319%	0.0%	1.2821%
Wielomianowe st. 3	0.5319%	0.0%	1.2821%
Wielomianowe st. 7	1.0638%	0.9091%	1.2821%
Wielomianowe st. 12	1.5957%	1.8182%	1.2821%
Normalne(Gaussowskie)	0.5319%	0.0%	1.2821%

Tablica 5.6. Wyniki testów algorytmu kodowego metoda 3D porównywania kodów

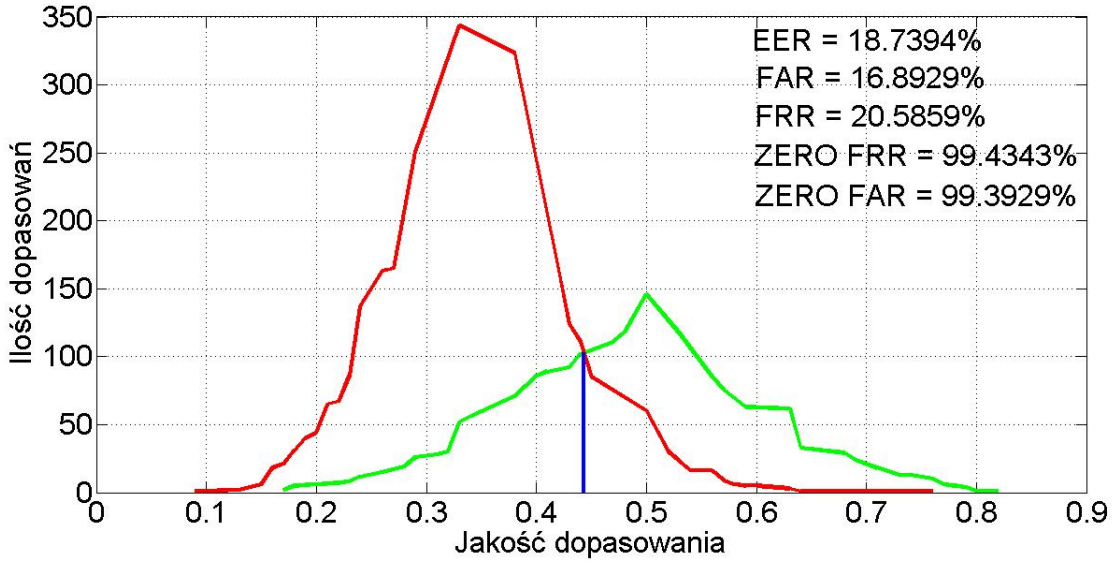
najkorzystniejszym wariantem okazał się wariant uwzględniający porównania 1:1 do sumy 1:0 i 0:1. Przyczyną poprawy jest możliwość znalezienia funkcji ńiemalżeśeparujących grupy punktów. Oznacza to, że sposób porównywania podobieństwa i niepodobieństwa jednocześnie podnosi skuteczność porównań. warunkiem koniecznym takiego zachowania jest możliwość separowania punktów porównań. W przeciwnym wypadku metoda prowadzi do pogarszania wyników.

5.3. Wnioski

Metoda 3D porównywania kodów jest metodą której skuteczność jest ściśle uzależniona od rozkładu punktów na płaszczyźnie i możliwości ich oddzielenia od siebie. W przypadku punktów których nie można odseparować metoda jedynie pogorszy uzyskiwane wyniki w stosunku do porównań tylko jednej cechy. Nie wskazane jest używanie zbyt zawiłych funkcji separujących. Klasyfikator powinien jedynie określać przybliżony rozkład punktów funkcje złożone powodują uczenie się na podstawie pojedynczych punktów, które nie mają odzwierciedlenia do całości zbioru porównań odcisków. Metoda nie gwarantuje 100% skuteczności porównywania odcisków, ale istnieją takie konfiguracje metody gdzie pojedyncze współczynniki są pozbawione błędu(np FRR dla jadra liniowego wariantu porównań 1:1 do sumy 1:0 i 0:1 dla SDK *Neurotechnology*). Nie należy się jednak sugerować zbytnio otrzymanymi wynikami, twierdząc że powstała metoda bezbłędnie klasyfikująca odciski. Wszystko zależy od zbiory uczącego i zbioru testującego. Jednak opisana tu metoda dla SDK *Neurotechnology* jest najlepszą z przedstawianych metod.

6. Ulepszenia

W poprzednich rozdziałach przedstawiono wyniki algorytmu kodującego. Ponieważ nie są one zadowalające pomyślano nad sposobom poprawy algorytmu. Wadliwe działanie można usprawiedliwiać zbyt trudnym dopasowaniem kodów, ponieważ nie uwzględniano sąsiednich kratek kodowych. Podjęto więc decyzję o zmianie algorytmu w taki sposób aby porównywać nie tylko bezpośrednio odpowiadające kratki kodu, ale również ich sąsiadów. Tak więc dopasowanie jest nie tylko gdy zgodne są wszystkie współrzędne kodu x, y i kąt, ale również dopasowanie gdzie dowolna ze współrzędnych jest przesunięta o 1 kratkę kodową w górę lub w dól. Rozwiązanie sprzyja dopasowaniu kodów i powinno w jakiś sposób rekompensować nieliniowe przekształcenia opuszka palca.

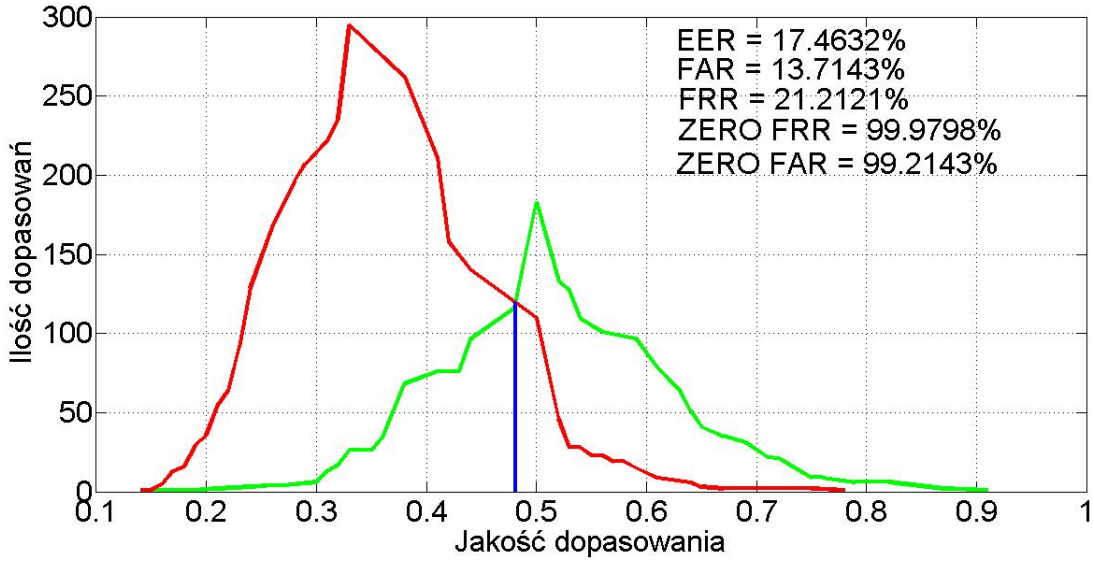


Rysunek 6.1. Ilość dopasowań w zależności od jakości dopasowania do odcisku wzorca (poprawiony algorytm)

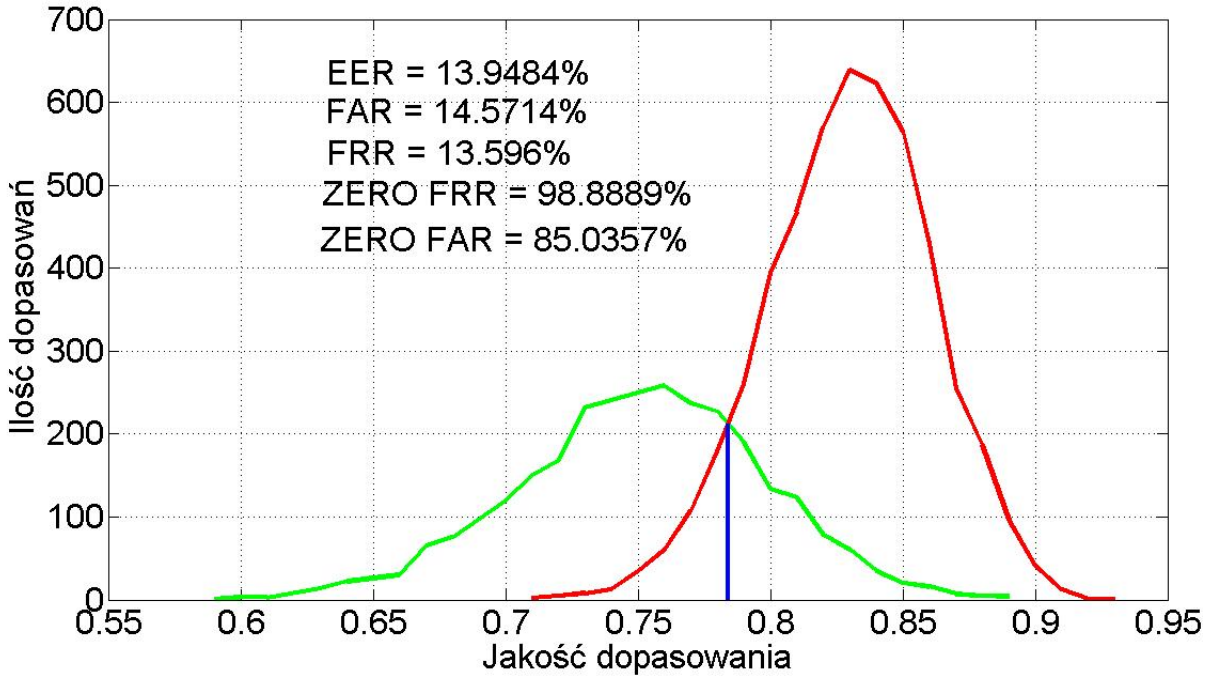
Wykresy 6.1 oraz 6.2 prezentują wyniki poprawionego algorytmu. Wskaźnik EER poprawił się tylko nieznacznie z 17%-21% na 18% - 19%. FAR zmienił się z 11% - 25% na 13% - 16%. FRR z 17% - 23% na 21% - 20%. Jednak ogólna ocena mówi o niewielkiej zmianie przedstawionej modyfikacji algorytmu.

Natomiast dal analizy różnicy symetrycznej wyniki nieco się poprawiły. EER spadł z 16% na 13%, FAR wzrósł z 12% na 14%, a FRR spadł z 17% na 13%. Wyniki przedstawione są na wykresie 6.3

Tabele 6.1, 6.2 i 6.3 przedstawiają wyniki metody 3D. Porównywania kodów również uzyskały podobne wyniki jak w niepoprawianej wersji algorytmu. Biorąc



Rysunek 6.2. Ilość dopasowań w zależności od jakości dopasowania do odcisku próbki (poprawiony algorytm)



Rysunek 6.3. Ilość niedopasowań w zależności od jakości niedopasowania do odcisków

pod uwagę najlepszy wynik metod poprawiła się o ok 4 punkty procentowe. Globalnie metoda 3D tak jak i poprzednio wypadła znacznie gorzej niż analizowanie

różnicy symetrycznej. Identycznie jak w poprzednich przypadkach przyczyną jest niemożliwość separowania zbiorów punktów porównań.

Wariant	1:1 do 1:0 + 0:1		
Zbiór testujący	ERRORS	FRR	FAR
Rodzaj jądra			
Liniowe	25.7359%	27.9635%	22.9755%
Kwadratowe	26.1564%	31.1550%	19.9623%
Wielomianowe st. 3	25.9882%	30.2432%	20.7156%
Wielomianowe st. 7	29.5206%	31.6109%	26.9303%
Wielomianowe st. 12	30.6981%	29.9392%	31.6384%
Normalne(Gaussowskie)	26.0723%	33.2827%	17.1375%

Tablica 6.1. Wyniki testów poprawionego algorytmu kodowego metoda 3D porównywania kodów

Wariant	1:1 do 1:0		
Zbiór testujący	ERRORS	FRR	FAR
Rodzaj jądra			
Liniowe	31.2500%	19.0283%	48.3051%
Kwadratowe	32.6651%	16.3968%	55.3672%
Wielomianowe st. 3	31.8396%	17.4089%	51.9774%
Wielomianowe st. 7	34.7877%	16.5992%	60.1695%
Wielomianowe st. 12	35.6132%	15.9919%	62.9944%
Normalne(Gaussowskie)	34.9057%	4.0486%	77.9661%

Tablica 6.2. Wyniki testów poprawionego algorytmu kodowego metoda 3D porównywania kodów

Wariant	1:1 do 0:1		
Zbiór testujący	ERRORS	FRR	FAR
Rodzaj jądra			
Liniowe	29.4652%	28.4872%	30.4979%
Kwadratowe	30.5752%	34.5776%	26.3485%
Wielomianowe st. 3	30.4743%	33.2024%	27.5934%
Wielomianowe st. 7	30.4743%	32.6130%	28.2158%
Wielomianowe st. 12	31.7861%	33.0059%	30.4979%
Normalne(Gaussowskie)	29.9697%	34.9705%	24.6888%

Tablica 6.3. Wyniki testów poprawionego algorytmu kodowego metoda 3D porównywania kodów

6.1. Wnioski

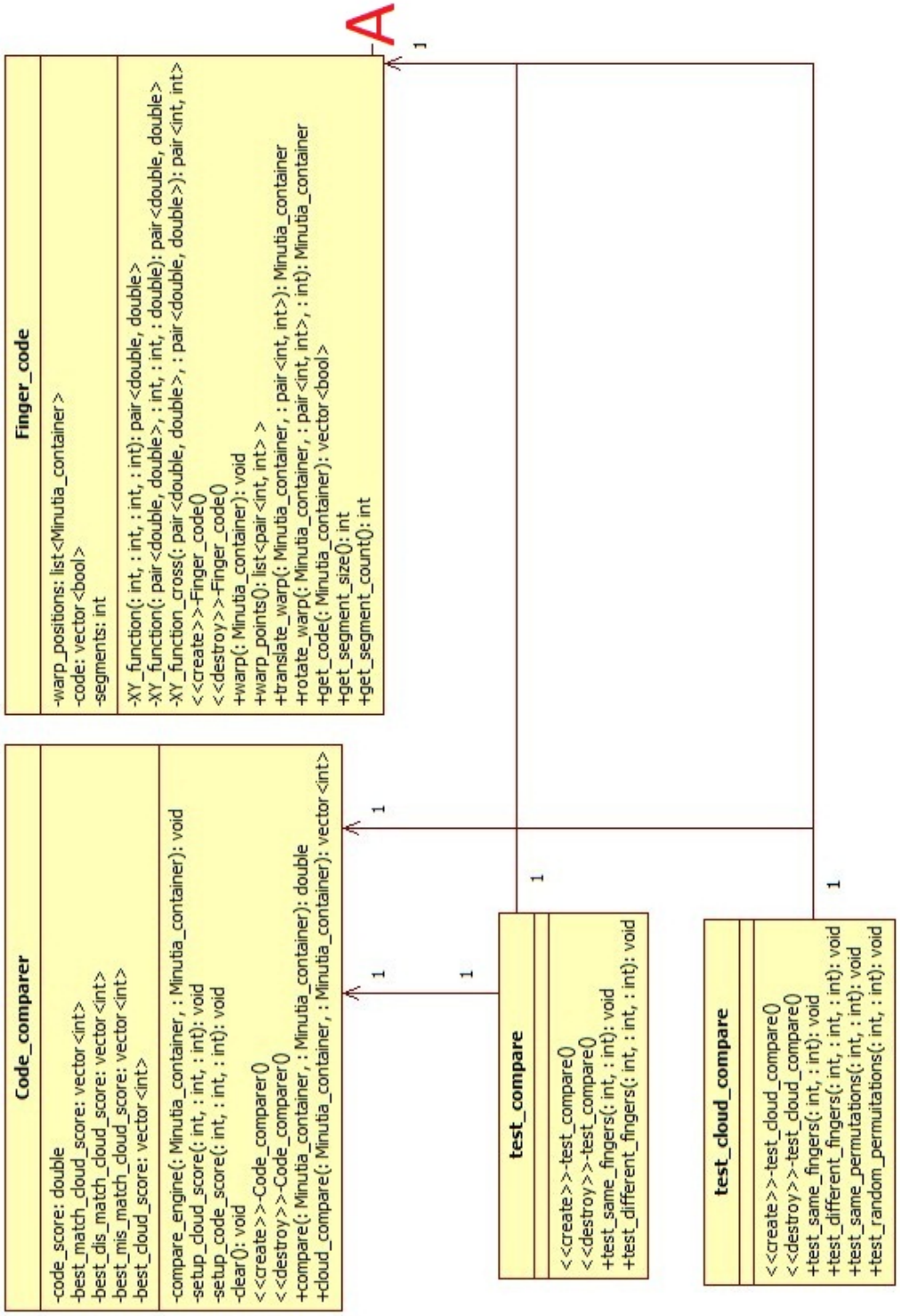
Oceniając zastosowane ulepszenie należy odnieść się do najlepszego uzyskanego wyniku, czyli analizy różnicy symetrycznej. Pomimo iż wskaźniki uległy poprawie nie należy uzyskanego wyniku uważać za satysfakcjonujący. 13% wynik nie jest akceptowalny dla systemów biometrycznych. Dodatkowo próby uzyskania błędów zerowych kończą się podobnie jak w niepoprawianej wersji algorytmu. Dla takich progów błędy FAR i FRR wynoszą ponad 80%. Świadczy to o wysokiej nieprzydatności zastosowanego rozwiązania. Pod sporym znakiem zapytania stoi również możliwość stosowania metod kodowania odcisku. Najważniejszym problemem jest niemożliwość spojrzenia na wszystkie minucje naraz. Oznacza to, że za każdym razem algorytm kodowania porównuje ze sobą dwa pojedyncze punkty i nie wie czy porównanie to ma jakikolwiek sens. Algorytmy obiektowe reprezentują minucje jako obiekty w programie, łatwe jest spojrzenie globalne na odcisk i wykluczanie dopasowań niemożliwych. Dla algorytmu kodującego porównanie kodu jest binarne i widzi on tylko pojedyncze elementy kodu.

7. Implementacja algorytmu

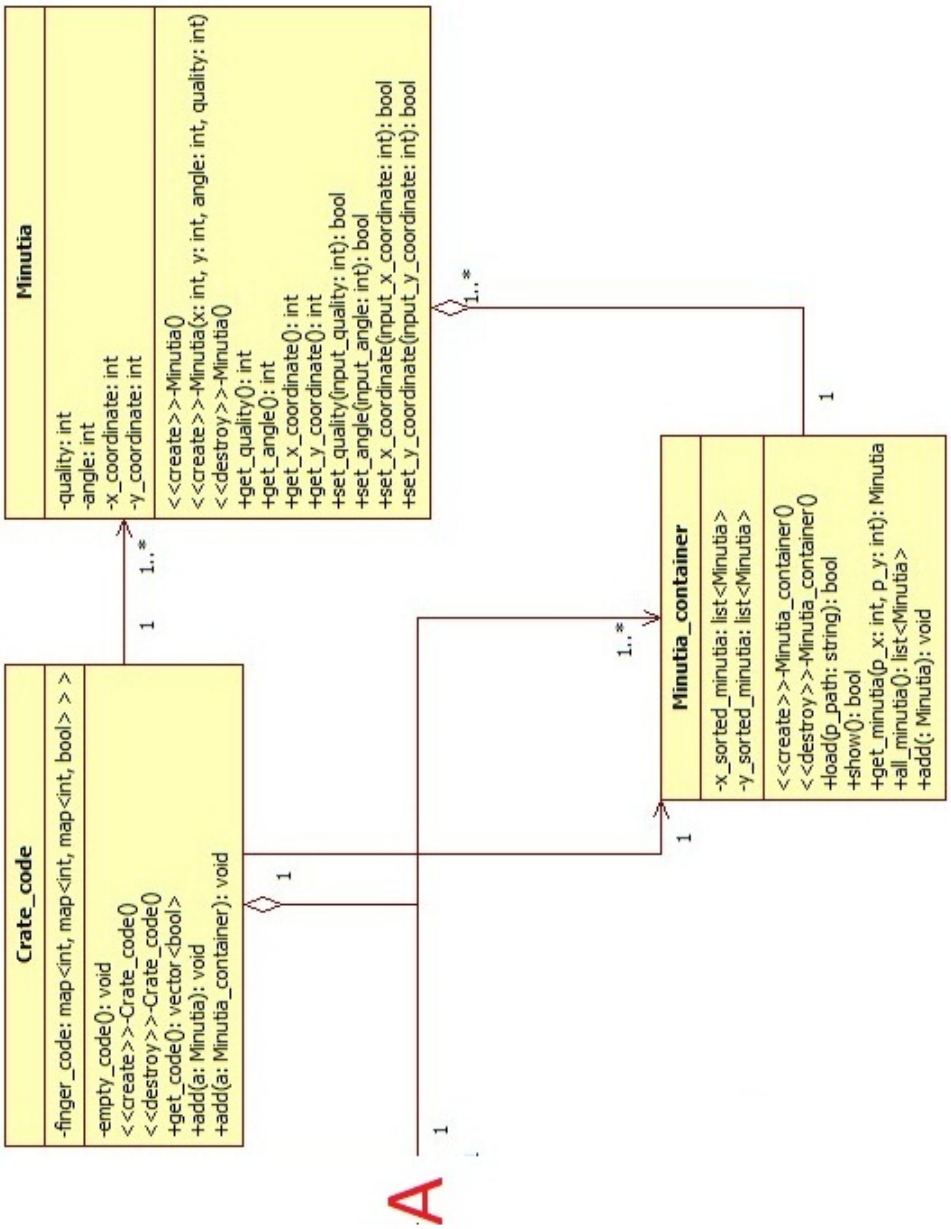
Implementacja algorytmu została wykonana w języku C++ z zastosowaniem standardowych bibliotek tego języka. Stworzony program to aplikacja konsolowa bez interfejsu użytkownika. Brak GUI motywowany jest koniecznością automatycznego wykonywania kodu na całej bazie danych obrazów. Dodatkowo aplikacja stanowi jedynie pomocnicze narzędzie do testowania wymyślonego algorytmu. Aplikacja składa się z trzech części modelu odcisku, czyli logicznego zapisu minucji dla każdego palca, silnika kodującego oraz silnika porównującego kody. Aplikacja jest jednowątkowa natomiast wszystkie funkcje napisane są w taki sposób aby w razie konieczności łatwo przerobić kod na wiele wątków. Aplikacja nie operuje na obrazach. Minucje są wydobywane za pomocą programu MINDTCT[1]. Aplikacja jako wejście przyjmuje pliki .xyt zawierające położenia XY i kąta kolejnych minucji odcisku. Minucje oddzielone są znakiem enter.

[1] Program MINDTCT jest częścią aplikacji NIST Fingerprint Image Software 2 wydaną przez Image Group of the National Institute of Standards and Technology

7.1. Diagram klas



Rysunek 7.1. Diagram klas



Rysunek 7.2. Diagram klas

7.2. Spis klas

1. Minutia
   - Klasa odzwierciedlająca fizyczny obiekt minucja. Stworzona do przechowywania podstawowych danych o minucji.
     — kąt
     — jakość
     — położenie x
     — położenie y

Klasa posiada podstawowe funkcje do zapisu i odczytu przechowywanych wartości.

2. Minutia_container
   - Klasa odzwierciedlająca fizyczny odcisk palca. Jest prostym kontenerem przechowującym Obiekty klasy $Minutia$. Jako kontener zastosowano $std :: list$. Klasa przechowuje minucje posortowane po współrzędnych, zarówno dla x jak i y.
     — lista minucji posortowanych po współrzędnej x
     — lista minucji posortowana po współrzędnej y
     Klasa posiada funkcje do wczytywania listy minucji z pliku $.xyt$, Dodawania pojedyńczego obiektu typu $Minutia$, zwracania minucji po współrzędnych, zwracania wszystkich minucji, oraz mechanizm wyświetlający wszystkie minucje na oknie konsoli(używane do debugowania).
3. Crate_code
   - Klasa przechowująca pojedynczą zakodowaną transformacje na obrazie minucji. Jest fizyczną reprezentacją naiwnego kodu odcisku.
     — Naiwny kod
     Klasa pozwala na dodawanie minucji oraz zbioru minucji do zakodowania. Posiada funkcje zwracającą utworzony kod, kod jest tworzony na bieżąco w momencie dodawania minucji.
4. Finger_code
   - Klasa przechowująca macierz kodów odcisku. Jest fizyczną reprezentacją całego kodu odcisku. Właściwie w funkcjach tej klasy zakodowany jest cały algorytm tworzący kod odcisku.
     — Kod odcisku
     — Liczba segmentów kodu
     — Lista punktów transformacji obrazu(reprezentowana jako minucje do których należy przesunąć obraz)
     Klasa posiada funkcje do zwrócenia kodu odcisku, pobrania rozmiaru segmentu kodu, liczby segmentów, oraz funkcje dokonujące transformacji obrazu.
5. Code_comparer
   - Klasa obliczająca stopień dopasowania odcisków, jest fizyczną reprezentacja funkcji porównującej kody. W zależności od wywoływanej funkcji zwraca odpowiedni współczynnik dopasowania. Działa w dwóch trybach zwracając liczbę minucji dopasowanych i niedopasowanych, lub oblicza współczynnik dopasowania na podstawie podanych wag.
     — najlepszy współczynnik dopasowania
     — liczby dopasowań, niedopasowań, maksymalizujące liczbę porównań 1:1
     — liczby dopasowań, niedopasowań, minimalizujące liczbę porównań 1:0
     — liczby dopasowań, niedopasowań, minimalizujące liczbę porównań 0:1
     — najlepszą liczbę dopasowań, niedopasowań, arbitralnie wybrany wskaźnik maksymalizujący liczbę porównań 1:1
     Klasa posiada silnik porównujący podane kody, oraz metody obliczające dopasowanie dla dwóch trybów działania.
6. Test_compare
   - Klasa do testowania mechanizmu porównywanie poprzez obliczanie współczynnika porównań.
7. Test_cloud_compare

- Klasa do testowania mechanizmu porównywanie poprzez obliczanie liczby dopasowań niedopasowań.

8. Config.h
   - Plik nagłówkowy zawierający elementy konfiguracji programu.

```
/*
 * Config.h
 *
 *  Created on: 2011-05-07
 *      Author: czareq
 */

#ifndef Config_H_
#define Config_H_

#include <limits.h>

#define ERROR INT_MAX
#define MIN_SCORE INT_MIN
const double PI = 3.14159265358979323851280895940;

/*
 * code parameters
 */
const int X_CODE_RES = 300;
const int Y_CODE_RES = 300;
const int ANGLE_RES = 300;
const int CODE_SEP = 25;
const int CODE_ANGLE_SEP = 30;

/*
 * warper parameters
 */
const double COVER_STEP = 0.5;
const int X_STEP = (int) (COVER_STEP * CODE_SEP);
const int Y_STEP = (int) (COVER_STEP * CODE_SEP);
const int ANGLE_STEP = 15;
const int ANGLE_STEP_RANGE = 45;

/*
 * comparer parameters
 */
const int MATCH_POINT = 500;
const int DIS_MATCH_POINT = 500;
const int MIS_MATCH_SCORE = 100;
#endif /* Config_H_ */
```

Wydruk 7.1. Przykładowa konfiguracja

7.3. Przebieg procesu weryfikacji

1. Wyodrębnienie listy minucji minucji
   - MINDTCT ekstrakcaj minucji wzorca i próbki
   - Zapis do pliku *.xyt*
2. Tworzenie modeli
   - Wczytanie plików *.xyt*

- Utworzenie obiektu $Minutia\_container$ dla wzorca i dla próbki oraz zapisanie w nich obiektów $Minutia$

3. Tworzenie kodów
   - Utworzenie obiektu $Crate\_code$ dla wzorca
   - Utworzenie obiektu $Finger\_code$ dla próbki
   - Zakodowanie próbki
     — Utworzenie listy wszystkich translacji
     — Utworzenie listy obrotów dla każdej translacji
     — Utworzenie obiektu $Crate\_code$ dla każdej transformacji
     — Dodanie obiektów $Crate\_code$ do obiektu $Finger_code$
4. Weryfikacja
   - Utworzenie obiektu $Code_compare$
   - Pobranie kodu wzorca
   - Pobranie kodu próbki
   - Podział kodu próbki na segmenty
   - Porównanie kodu wzorca z każdym z kodów segmentu
5. Zwrócenie i zapisanie wyniku
   - Odczytanie najlepszego wyniku porównania
   - Zapis do pliku

8. Weryfikacja obiektowo-kodowa

Ocenę metod weryfikacji odcisków należy wystawić przede wszystkim na podstawie ilości błędów popełnianych przez obie metody. W takim przypadku metody obiektowe zostawiają daleko w tyle opisywaną metodę kodową. Błędy EER w granicach 16% -20% świadczą o dużej zawodności metody. W książce *Handbook of fingerprint recognition* [4] prezentowana jest tabela prezentująca błędy EER dla znanych algorytmów porównywania odcisków. W czołówce znajdują się algorytmy o EER poniżej 1%, a najlepszy przedstawiany algorytm (PA15) ma skuteczność na poziomie 0.19%. Tabele zamykają algorytmy o skuteczności 12%- 16%, 23% i 50%. Oznacza to, że moje rozwiązanie mogłoby być porównywane z niektórymi najsłabszymi algorytmami przedstawianymi w książce. Najgorsze komercyjne rozwiązanie ma skuteczność 12.09% co oznacza, że mój algorytm jest niewiele gorszy. Dodatkowo w przypadku prób zapewnienia zerowych błędów fałszywej akceptacji lub fałszywego odrzucenia algorytm staje się kompletnie bezużyteczny. Jednak metoda niekoniecznie musi zostać porzucona warto zauważyć, że algorytm kodujący nie korzysta z żadnych zaawansowanych metod. Jest to tylko wyłącznie proste kodowanie poprzez podzielenie przestrzeni na mniejsze fragmenty i stwierdzanie czy minucja znajduje się w niej czy też nie. Ciężko wyciągać wnioski bez przeprowadzenia eksperymentu, ale gdyby zamiast określać miejsca w którym znajduje się minucja określać gdzie znajduje się z pewnym prawdopodobieństwem to metoda kodowa mogłaby być skuteczniejsza. Jednak takie podejście nie mówi wprost o ilości dopasowanych minucji. I prawdopodobnie również byłoby nieskuteczne dowodzi tego test przeprowadzony w poprzednim rozdziale. Porównywano tam nie tylko bezpośrednio odpowiadające kratki kodu ale też i sąsiednie. W obecnym podejściu algorytm podejmuje binarne decyzje co do dopasowania minucji. Wersja z kodowaniem prawdopodobieństwa położenia minucji w danym obszarze stwierdza dopasowanie minucji wraz z prawdopodobieństwem. Reasumując niełatwe jest wtedy stwierdzenie ile minucji dopasowano a ile nie. Podejście takie utrudniałoby badanie podobieństwa i niepodobieństwa obrazów na podstawie liczby dopasowanych i niedopasowanych minucji, a właśnie takie założenie przyjmuje ta praca. Innym sposobem poprawy jest znalezienie sposobu na kompensacje nieliniowych odkształceń opuszka palca w procesie pobierania odcisku. W aktualnym rozwiązaniu stosowane są przekształcenia(obroty i przesunięcia) dla całego obrazu. Nie stosowane są lokalne transformacje obrazu. Nie jasny jest też sposób realizacji takich przekształceń. Możliwym problemem wprowadzenia tego pomysłu jest nieskończony zbiór przekształceń jaki można wykonać na obrazie. Mówiąc prościej i tak należałoby arbitralnie wybrać jakąś grupę przekształceń bez pewności, że wśród nich znajduje się przekształcenie kompensujące zniekształcenia w obrazie. Tak naprawdę bardzo trudno jest znaleźć dobre wyjście z tego problemu, bo nie wiadomo co jest poszukiwane. Dobrym kierunkiem rozwoju metody jest, wspomniane powyżej, kompletne odejście od liczby dopasowanych minucji. Algorytm stwierdzi jedynie ważone prawdopodobieństwo dopasowania obrazu. Rozwiązanie takie różni

się od obecnego tym, że zniekształcenia obrazu są kompensowane nie tylko przez rozmiar kratki kodu ale również przez przynależność do sąsiednich kratek. Jest to jednak temat na kolejne badania.

8.1. Ocena i porównanie metod

Kolejnym etapem pracy było stworzenie mechanizmu porównywania kodów. Rozważano kilka sposobów porównań.

- porównywanie naiwnego kodu
- porównywanie macierzy kodów(transformacje na obrazie próbki)
  - — porównywanie podobieństwa
  - — porównywanie niepodobieństwa
  - — porównywanie podobieństwa i niepodobieństwa jednocześnie(3D)

Dla algorytmu kodującego najskuteczniejsze okazało się porównywanie niepodobieństwa i transformacje obrazu próbki. Dla liczby minucji dopasowanych i niedopasowanych pozyskiwanych od SDK $Neurotechnology$ najskuteczniejszą metodą okazała się metoda 3D porównywania kodów. Skuteczność takiego porównania daje wynik w granicach 0.5% błędnych decyzji. Wynik ten śmiało mógłby rywalizować z przedstawianym algorytmami w książce $Handbookof fingerprintrecognition$ [4]. Niestety nie jest to moje autorskie rozwiązanie a jedynie wykorzystanie komercyjnego SDK w połączeniu z moją metodą. Wynik jednak jest bardzo dobry i taki system mógłby być działającym powszechnie systemem biometrycznym. Konieczne jest jedynie nauczenie klasyfikatora SVM odpowiedniej funkcji separującej. W opisywanej pracy funkcje zostały otrzymane na niezbyt dużych bazach danych. Zwiększenie zbioru uczącego przyczyni się do większego generalizowania zachowania rozkładu ilości punktów dopasowanych i niedopasowanych minucji. Metoda ta nie jest uniwersalna, oznacza to ze proces uczenia klasyfikatora powinien być dostosowywany do oprogramowania, które oblicza liczby dopasowanych i niedopasowanych cech. Jak każda metoda tak i ta posiada wady. Ten sposób klasyfikacji nie daje oczekiwanych rezultatów gdy zbiór punktów dopasowania i zbiór punktów niedopasowania nie jest rozłączny. W przypadku przedstawianego algorytmu kodującego metoda te prowadzi wręcz do pogorszenia wyników.

Bibliografia

[1] Wikimedia Foundation. Angielska i polska wersja wikipedii, 2012.

[2] Chih-Jen Lee, Tai-Ning Yang, Chun-Jung Chen, Allen Y. Chang, Sheng-Hsuan Hsu. Fingerprint identification using local gabor filters. *Department of Computer Science Chinese Culture University Taipei, Taiwan, R.O.C.*

[3] Chia-Hung Lin, Jian-Liung Chen, Zwe-Lee Gaing. Combining biometric fractal pattern and particle swarm optimization-based classifier for fingerprint recognition. *Hindawi Publishing Corporation Mathematical Problems in Engineering*, 2010.

[4] Davide Maltoni, Dario Maio, Anil K.Jain, Salil Prabhakar. *Handbook of Fingerprint Recognition*. Springer, USA 2005.

[5] Marius Tico, Eero Immonen, Pauli Ramo, Pauli Kuosmanen, Jukka Saarinen. Fingerprint recognition using wawelet features. *Digital Media Institute, Tampere University of Technology*.