POLITECHNIKA WARSZAWSKA

**Wydział Mechatroniki**

**Praca dyplomowa inżynierska**

Ireneusz Szulc

Inteligentny interfejs dotykowy umożliwiający obsługę  
złożonych gestów

Opiekun pracy:

mgr inż. Bogdan Harasymowicz-Boggio

Instytut Automatyki i Robotyki

Warszawa, 2016 r.

Oryginał karty pracy

# Streszczenie w języku polskim (1 strona A4)

# Streszczenie w języku angielskim (1 strona A4)

# Życiorys

# Spis treści

Spis treści

[Streszczenie w języku polskim (1 strona A4) 2](#_Toc440980428)

[Streszczenie w języku angielskim (1 strona A4) 2](#_Toc440980429)

[Życiorys 2](#_Toc440980430)

[Spis treści 2](#_Toc440980431)

[1. Wstęp 2](#_Toc440980432)

[1.1. Cel pracy 2](#_Toc440980433)

[1.2. Założenia projektu 2](#_Toc440980434)

[1.3. Zastosowanie 3](#_Toc440980435)

[1.4. Przegląd istniejących rozwiązań 4](#_Toc440980436)

[2. Wstęp teoretyczny 5](#_Toc440980437)

[2.2. Metody minimalnoodległościowe 5](#_Toc440980438)

[2.1. Klasyfikatory 6](#_Toc440980439)

[2.1.1. Kody łańcuchowe Freemana 6](#_Toc440980440)

[2.1.2. Sieci neuronowe 6](#_Toc440980441)

[2.1.3. Supported Vector Machine 6](#_Toc440980442)

[2.3. Metody porównywania histogramów - metryki 6](#_Toc440980443)

[2.3.1. Współczynnik korelacji Pearsona 6](#_Toc440980444)

[2.4. System operacyjny Android 7](#_Toc440980445)

[2.5. Biblioteka OpenCV 7](#_Toc440980446)

[2.3. Słownik używanych pojęć 7](#_Toc440980447)

[3. Algorytm rozpoznawania pojedynczych konturów 8](#_Toc440980448)

[3.1. Założenia algorytmu 8](#_Toc440980449)

[3.2. Wybór metody 10](#_Toc440980450)

[3.3. Akwizycja konturu 11](#_Toc440980451)

[3.4. Przetwarzanie wstępne 11](#_Toc440980452)

[3.4.1. Interpolacyjne uzupełnienie listy punktów 11](#_Toc440980453)

[3.4.2. Filtracja 11](#_Toc440980454)

[3.5. Ekstrakcja cech 12](#_Toc440980455)

[3.6. Reprezentacja obiektu 13](#_Toc440980456)

[3.7. Klasyfikacja 14](#_Toc440980457)

[3.7.1. Korelacja histogramów 14](#_Toc440980458)

[3.7.2. Korelacja punktów startowych 14](#_Toc440980459)

[3.7.3. Korelacja długości gestu 15](#_Toc440980460)

[3.7.4. Sumaryczny współczynnik korelacji 16](#_Toc440980461)

[3.8. Przypadek krótkich konturów 17](#_Toc440980462)

[3.9. Schemat algorytmu 17](#_Toc440980463)

[3.10. Przykłady 20](#_Toc440980464)

[4. System rozpoznawania złożonych gestów 21](#_Toc440980465)

[4.1. Opis algorytmu 21](#_Toc440980466)

[4.1.1. Kryteria wyboru 22](#_Toc440980467)

[4.1.2. Dane wejściowe 22](#_Toc440980468)

[4.1.3. Wstępna klasyfikacja 23](#_Toc440980469)

[4.1.4. Wypadkowy współczynnik korelacji 23](#_Toc440980470)

[4.1.5. Wybór najlepszego rozwiązania 23](#_Toc440980471)

[4.2. Schemat algorytmu 24](#_Toc440980472)

[4.3. Przykłady 26](#_Toc440980473)

[5. Inteligencja systemu 28](#_Toc440980474)

[5.1. Problem błędnych wzorców 29](#_Toc440980475)

[5.2. Rejestrowanie bilansu identyfikacji 29](#_Toc440980476)

[5.3. Automatyczne usuwanie błędnych wzorców 30](#_Toc440980477)

[5.4. Automatyczne dodawanie wzorców 30](#_Toc440980478)

[5.4.1. Postępowanie z automatycznie dodanymi wzorcami 30](#_Toc440980479)

[5.5. Optymalizator liczby wzorców 30](#_Toc440980480)

[5.6. Interakcja z użytkownikiem 31](#_Toc440980481)

[6. Implementacja oprogramowania na urządzenie mobilne 32](#_Toc440980482)

[6.1. Urządzenie i środowisko programowania 32](#_Toc440980483)

[6.2. Struktura aplikacji 33](#_Toc440980484)

[6.3. Obsługa aplikacji 36](#_Toc440980485)

[6.3.1. Interfejs użytkownika 36](#_Toc440980486)

[6.4. Wykorzystane metody 38](#_Toc440980487)

[6.4.1. Wykorzystanie biblioteki OpenCV 38](#_Toc440980488)

[6.4.2. Mechanizm serializacji 39](#_Toc440980489)

[6.5. Moduł wprowadzania znaków pisma odręcznego 39](#_Toc440980490)

[6.6. Cechy systemu 40](#_Toc440980491)

[6.7. Testy aplikacji 40](#_Toc440980492)

[6.7.1. Dane statystyczne 40](#_Toc440980493)

[6.7.2. Test uczenia się systemu 41](#_Toc440980494)

[6.7.3. Dobór parametrów 41](#_Toc440980495)

[6.7.4. Testy końcowe działającej aplikacji 42](#_Toc440980496)

[Bibliografia 43](#_Toc440980497)

# 1. Wprowadzenie

## 1.1. Cel pracy

Celem pracy jest zaprojektowanie i stworzenie inteligentnego interfejsu dotykowego umożliwiającego obsługę złożonych gestów. Taki interfejs w formie oprogramowania na urządzenie z ekranem dotykowym powinien być odpowiedzialny za rozpoznawanie sekwencji elementarnych konturów rysowanych przez użytkownika poprzez przeciąganie palca lub rysika po ekranie. Wynik analizy wprowadzonych kombinacji gestów może posłużyć do rozpoznawania znaków odręcznego pisma lub wykonania zdefiniowanych akcji w systemie, na którym pracuje urządzenie. Stworzony system pozwala użytkownikowi na swobodne definiowanie własnych gestów o dowolnej złożoności.

Inteligencja powstałego interfejsu polega na zdolności systemu do samouczenia się, co w praktyce oznacza, że system jest w stanie zwiększać swoją skuteczność w rozpoznawaniu znaków poprzez dostosowywanie się do charakteru pisma odręcznego użytkownika oraz zmianę swojego zachowania wskutek akcji wykonywanych przez użytkownika.

istnieją rozwiązania, ale nie rozpoznają dowolnych, złożonych gestów lub nie są inteligentne, nie są konfigurowalne

## 1.2. Założenia projektu

Założeniem projektu jest, aby interfejs dotykowy miał formę aplikacji działającej na urządzeniu mobilnym wyposażonym w system operacyjny Android. Dodatkowo takie urządzenie powinno posiadać ekran dotykowy, który posłuży za źródło danych wejściowych dla aplikacji. Językiem programowania wykorzystanym do stworzenia całej aplikacji jest język Java.

Interfejs powinien umożliwiać rozpoznawanie znaków pisma odręcznego wprowadzanych przez użytkownika poprzez ruch palca lub rysika po ekranie dotykowym, zamianę ich na tekst w formie cyfrowej i przechowywanie go w celu dalszego wykorzystania. Wprowadzane znaki mogą składać się zarówno z wielu fragmentów pisanych z odrywaniem rysika od ekranu, jak również z pojedynczych konturów.

Dodatkowym założeniem jest, aby system uczył się w trakcie działania. Aplikacja powinna nauczyć się charakteru pisma odręcznego od użytkownika, powinna także zwiększać swoją skuteczność w rozpoznawaniu gestów w miarę dłuższego czasu użytkowania. Implikuje to konieczność przechowywania dodatkowych danych zbieranych w trakcie działania aplikacji w celu dostosowywania się do charakteru pisma użytkownika. Cały system powinien również reagować na polecenia uczące wydawane przez użytkownika i zmieniać sposób swojego działania.

Zakres pracy obejmuje trzy główne zagadnienia:

* Projekt i implementację algorytmu rozpoznawania elementarnych gestów,
* Opracowanie metody rozpoznawania złożonych gestów oraz wprowadzania znaków na podstawie pisma odręcznego,
* Projekt i implementacja oprogramowania na urządzenie mobilne.

Praca skupia się głównie na rozpoznawaniu znaków pisma odręcznego, ale łatwo można rozszerzyć tą funkcjonalność systemu na rozpoznawanie gestów wykonujących określone akcje w systemie (np. skrótowe uruchamianie aplikacji), gdyż zaprojektowany system nie ma ograniczenia rozpoznawania dowolnych gestów o dowolnej złożoności.

## 1.3. Zastosowanie

Opracowany system mógłby znaleźć zastosowanie jako wygodna metoda wprowadzania znaków na urządzeniach z panelem dotykowym. Takie rozwiązanie nie ogranicza się jedynie do wprowadzania tekstu, ale pozwala także na równie szybkie wprowadzanie cyfr, znaków diakrytycznych (polskich liter) oraz znaków specjalnych (takich jak: przecinek, kropka, wykrzyknik, itd.), co może przemawiać na korzyść metody w porównaniu do konwencjonalnych klawiatur ekranowych, gdzie wprowadzanie znaków specjalnych może stać się uciążliwe i czasochłonne. Metoda ta jest bardzo intuicyjna, charakteryzuje się prostotą i mogłaby być chętniej wykorzystywana np. przez ludzi starszych przyzwyczajonych do tradycyjnego wprowadzania tekstu.

Kolejnym polem zastosowań są urządzenia z bardzo małymi ekranami dotykowymi (np. smartwatche), gdzie często wykorzystanie klawiatury ekranowej może być uciążliwe z powodu bardzo małych przycisków, zaś wprowadzenie sterowania urządzeniem poprzez gesty oraz pisania dowolnych znaków metodą pisma odręcznego mogłoby zwiększyć komfort użytkowania takiego urządzenia.

System może posłużyć również jako metoda wprowadzania haseł w postaci sekwencji gestów w celu uzyskania dostępu do określnych zasobów. Przetrzymywanie hasła w postaci wzorca charakteru pisma może być bardzo pomocne w procesie uwierzytelniania użytkowników i przyznawać dostęp jedynie osobie, która ma dokładnie taki sam charakter pisma lub w dokładnie taki sam sposób wykonała określone gesty. System z takim rozwiązaniem nie przyznawałby dostępu osobie, która poznała hasło, lecz wprowadziła je w inny sposób.

Interfejs dotykowy będący przedmiotem pracy można również wykorzystać jako rozszerzenie systemu, na którym pracuje urządzenie do skrótowego uruchamiania aplikacji lub niektórych funkcji systemu. Opracowaną metodę można zaimplementować także na panelach operatorskich HMI z ekranem dotykowym w celu wygodnego i łatwego przeglądania monitorowanych wielkości i wizualizowanych procesów w przemyśle.

## 1.4. Przegląd istniejących rozwiązań

gotowe rozwiązania na rynku, opis, oprogramowanie w smatrfonach

przegląd metod on-line Handwriting Recognition, opis

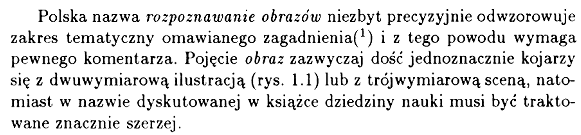
inne prace naukowe, czego użyli

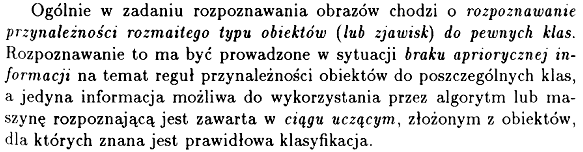
on-line Handwriting Recognition

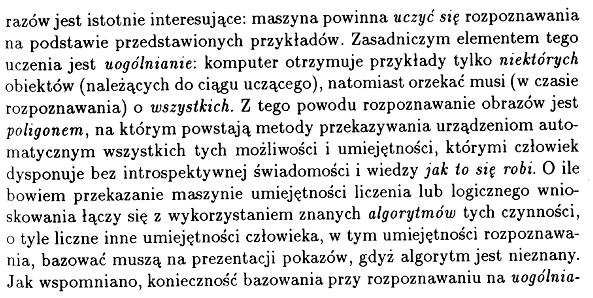
off-line Handwriting Recognition

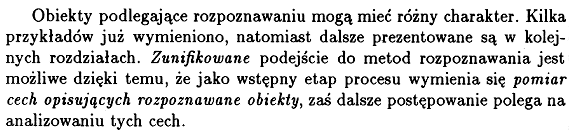
# 2. Wstęp teoretyczny

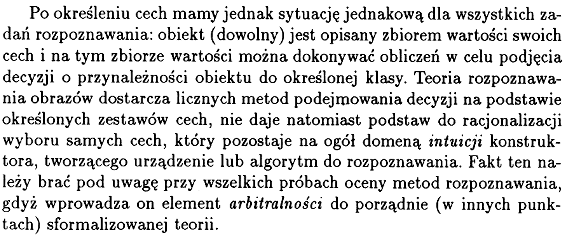
## 1.5. Rozpoznawanie obrazów

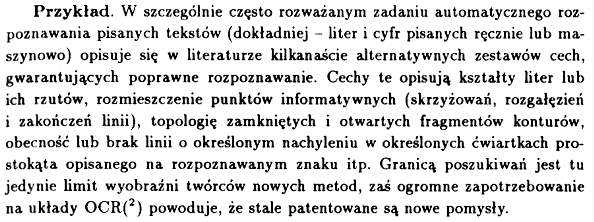


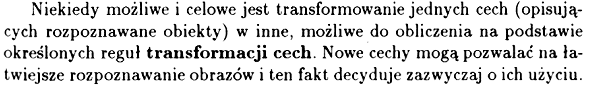






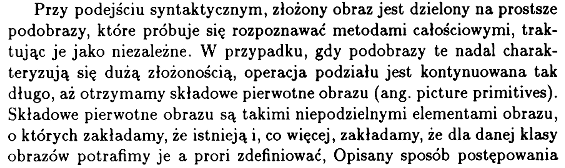


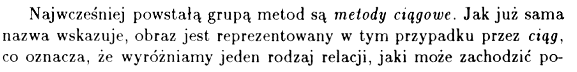




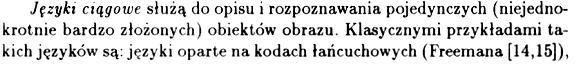
### 1.1.1. Syntaktyczne (strukturalne) rozpoznawanie obrazów











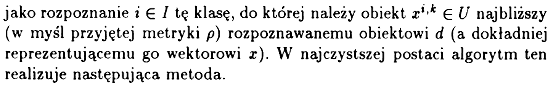
"...syntaktyczne rozpoznawanie obrazów - reprezentowanie obiektu przez ciąg...

[1]

klasyfikacja, przetwarzanie wstępne, etapy, ekstrakcja cech

## 1.6. Metody minimalnoodległościowe





Jest to metoda służąca do klasyﬁkacji według określonych klas. Na podstawie zbioru treningowego, dla którego elementów znane są etykiety klas dokonuje się klasyﬁkacji zbioru danych. Do tego celu potrzebna jest miara odległości między elementami obydwu zbiorów (zależna od rodzaju klasyﬁkowanych obiektów i dziedziny problemu). Dla każdego elementu zbioru testowego znajduje się element zbioru treningowego, do którego odległość jest najmniejsza, a klasyﬁkowanemu elementowi przypisuje się tę etykietę, która pojawia się najczęściej w znalezionym zbiorze. Algorytm można stosować z różnymi rodzajami metryk.

Metoda najbliższego sąsiada

duża wrażliwość na błędy ciągu uczącego

[Tadeusiewicz]

## 1.7. Klasyfikatory

klasyfikacja

### 1.7.1. Kody łańcuchowe Freemana

Normally, the contours created by cvFindContours() are sequences of vertices (i.e.,

points). An alternative representation can be generated by setting the method to

CV\_CHAIN\_CODE. In this case, the resulting contours are stored internally as Freeman chains

[Freeman67] (Figure 8-4). With a Freeman chain, a polygon is represented as a sequence

of steps in one of eight directions; each step is designated by an integer from 0 to 7. Free-

man chains have useful applications in recognition and other contexts.



[2]

zamiana na wektory

### 1.7.2. Sieci neuronowe

### 1.7.3. Supported Vector Machine

inne prace, źródła, jak inni to robili i co stwierdzili

to najprawdopodobniej wyjebać

## 1.8. Metody porównywania histogramów - metryki

Histogramy - co to?

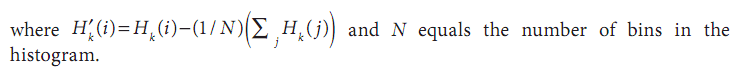
### 1.8.1. Współczynnik korelacji Pearsona

Współczynnik korelacji liniowej Pearsona to współczynnik, który określa poziom zależności liniowej między zmiennymi losowymi i . Niech oraz oznaczają wartości prób losowych tych zmiennych, zaś i wartości średnie z tych prób. Estymator współczynnika korelacji liniowej Pearsona wyraża się wzorem:

Wartość współczynnika korelacji Pearsona zawsze mieści się w przedziale:

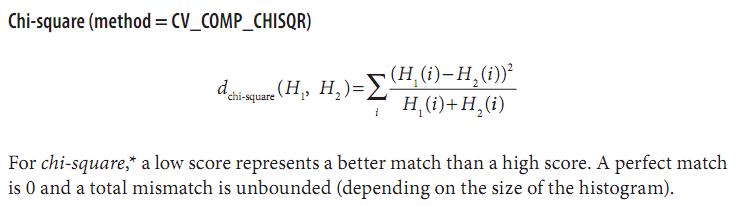
d(H_1,H_2) =  \frac{\sum_I (H_1(I) - \bar{H_1}) (H_2(I) - \bar{H_2})}{\sqrt{\sum_I(H_1(I) - \bar{H_1})^2 \sum_I(H_2(I) - \bar{H_2})^2}}

\bar{H_k} =  \frac{1}{N} \sum _J H_k(J)

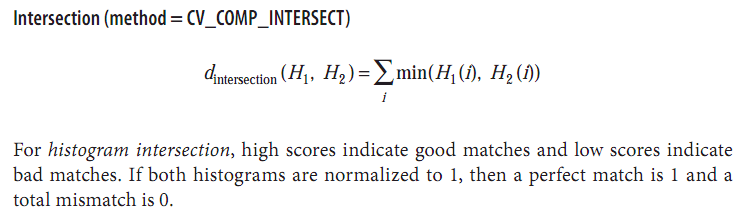


For correlation, a high score represents a better match than a low score. A perfect match is 1 and a maximal mismatch is –1; a value of 0 indicates no correlation (random association). [2]

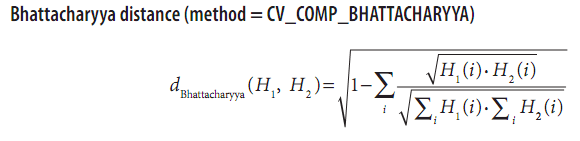
### 1.8.2. Chi-kwadrat



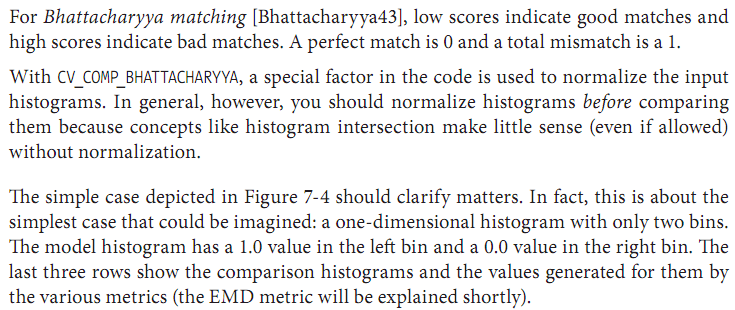
### 1.8.3. Metoda przecięcia

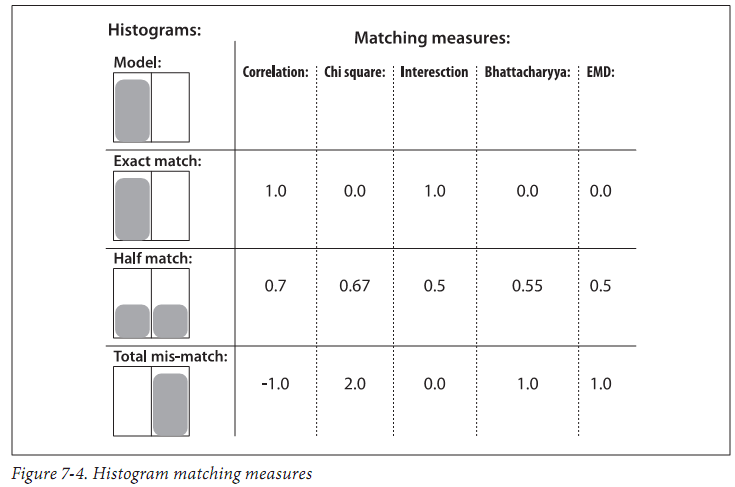


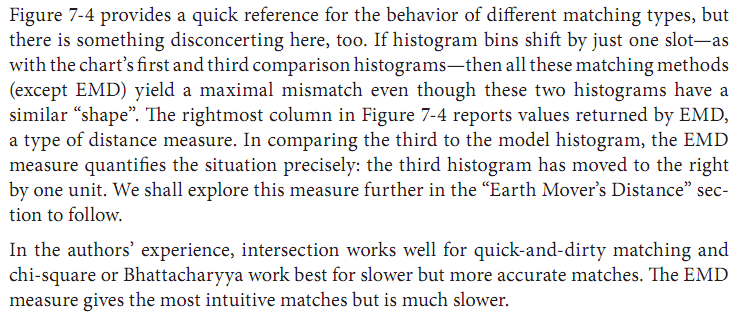
### 1.8.4. Odległość batachararadyaydadsa



Odległość Bhattacharyya jest miarą stosowaną w statystyce do oszacowania różnicy między dwoma rozkładami prawdopodobieństwa.







[2]

## 1.9. System operacyjny Android

łatwość programowania, Android - Linux z JVM, licencja darmowa, wiele urządzeń pracuje )procent)

## 1.10. Biblioteka OpenCV

OpenCV ma implementację na system Android w Javie, licencja darmowa

## 1.11. Słownik używanych pojęć

W pracy stosowane są następujące pojęcia:

**Gest elementarny, gest składowy, gest pojedynczy** - kontur (sekwencja punktów) powstały w wyniku jednorazowego przesunięcia palca lub rysika po ekranie bez odrywania go od ekranu. Istotna jest kolejność punktów, a więc kontur jest ciągiem a nie jedynie zbiorem punktów.

**Gest złożony, gest wieloelementowy** - sekwencja wielu następujących po sobie gestów elementarnych, w której istotna jest kolejność elementów;

**Złożoność gestu -** liczba gestów elementarnych wchodzących w skład gestu złożonego. Przykładem złożonego gestu jest rysowanie kształtu dużej litery "A", która pisana jest zazwyczaj dwuetapowo i składa się z dwóch elementów: kształtu " oraz poziomej kreski  
"-". Zatem złożoność takiego gestu wynosi 2.

**Punkt startowy konturu** - pierwszy punkt konturu, miejsce początku konturu;

kontur - uszeregowana lista punktów

sumaryczny współczynnik korelacji - średnia ważona z wielu

wypadkowy współczynnik korelacji - średnia arytmetyczna z sumarycznych współczynników korelacji dla złożonego gestu

zkorygowany współczynnik korelacji -

# 2. Algorytm rozpoznawania pojedynczych konturów

Rozdział ten skupia się na opracowaniu i zaimplementowaniu metody rozpoznawania pojedynczych gestów składowych (bez rozważania kontekstu, w jakim zostały wprowadzone).

Ogólny proces rozpoznawania obiektów można podzielić na kilka zasadniczych części:

* przetwarzanie wstępne danych wejściowych,
* ekstrakcja cech,
* utworzenie reprezentacji nieznanego obiektu,
* właściwe rozpoznawanie (klasyfikacja).

Zastosowany algorytm opiera się na jednej z metod minimalnoodległościowych - metodzie najbliższych sąsiadów. Polega ona na bardzo prostym rozumowaniu mówiącym, że wynikiem rozpoznania jest klasa, do której należy obiekt najbliższy (w myśl przyjętej metryki) rozpoznawanemu obiektowi.

W przypadku rozważanego w projekcie zagadnienia rozpoznawania konturów (będących danymi wejściowymi dla algorytmu), obiektami są właśnie poszczególne kontury, które mogą być pogrupowane w klasy. Obiekty te leżą w wielowymiarowej przestrzeni (liczba wymiarów przestrzeni jest równa liczbie cech obiektów). Położenie obiektów jest więc w pełni uwarunkowane cechami obiektów, natomiast metryka określa w tym przypadku stopień podobieństwa konturów (jak bardzo dwa obiekty są do siebie zbliżone). Na tej podstawie można wyznaczyć stopień podobieństwa nowych danych wejściowych (niesklasyfikowanych) do każdego obiektu, który już został sklasyfikowany. Następnie można przyporządkować nieznanemu obiektowi klasę, do której należy najbardziej zbliżony (o minimalnej odległości) znany obiekt. Zasadniczym problemem w tej metodzie staje się zdefiniowanie samej metryki określającej odległość między obiektami (stopień ich podobieństwa). Taka metoda implikuje również konieczność przechowywania wzorców w wewnętrznej bazie aplikacji, które przynależą już do określonych klas i mogą stać się podstawą do identyfikacji nowo wprowadzonych, nieznanych obiektów.

## 2.1. Założenia algorytmu

Aby uzyskać jak największą skuteczność w poprawnym rozpoznawaniu elementarnych gestów i jednocześnie zachować pewną tolerancję danych wejściowych (nowych, nierozpoznanych konturów) na zniekształcenia w stosunku do wzorca, konieczne jest rozważenie problematycznych przypadków oraz określenie wymagań w stosunku do algorytmu. Może to pomóc w uniknięciu błędnego działania algorytmu w przyszłości.

Każdy gest pisany przez użytkownika jest zniekształconym naśladownictwem pewnego abstrakcyjnego idealnego wzorca, do którego dążą w sposób niedoskonały wszystkie konkretne realizacje obiektów z danej klasy. Należy więc pamiętać o tym, że proces identyfikacji konturów powinien być niewrażliwy (jedynie w odpowiednim zakresie) na wybrane transformacje obrazu.

Rozważany algorytm powinien spełniać poniższe założenia wynikające z charakteru wykonywania gestów przez użytkownika:

* uwzględnianie kierunku wykonywania gestu - Program powinien rozróżniać między sobą gesty, które mają ten sam kształt konturu, lecz powstały w wyniku odwróconej sekwencji ruchów na ekranie dotykowym. Właściwość kierunku konturu można odczytać tylko w przypadku metod on-line Handwriting Recognition, gdyż znana jest cała historia i dynamika powstawania konturu. Metoda ta śledzi na bieżąco cały proces powstawania konturu. Natomiast w przypadku metod off-line Handwriting Recognition, nie można stwierdzić, jaki był kierunek powstawania konturu na podstawie samego obrazu (np. z kamery), będącego jedynie końcowym efektem pisania znaku. Znajomość cechy kierunku jest dużą zaletą metody, gdyż wykorzystanie jej może znacznie poprawić skuteczność rozpoznawania algorytmu.
* częściowa niewrażliwość na obrót (jedynie w odpowiednim zakresie) - Użytkownik powinien mieć możliwość wpisania znaku w nieco innej, zniekształconej orientacji, ale tylko w ustalonym, niewielkim zakresie. Pełna niewrażliwość na obrót może prowadzić do błędnego, jednakowego interpretowania niektórych znaków, np. dużych liter "M" i "W", "N" i "Z", lub cyfr "6" i "9", których ogólny kształt może różnić się jedynie orientacją, co mogłoby prowadzić do nierozróżniania takich znaków między sobą.
* częściowa niewrażliwość na przesunięcie (jedynie w odpowiednim zakresie) - Za każdym razem gest wykonywany jest nieco inaczej przez użytkownika, również punkt rozpoczęcia rysowania znajduje się w różnych miejscach. Tutaj również pełna niewrażliwość na przesunięcie konturu w układzie współrzędnych ekranu nie sprawdziłaby się w pewnych przypadkach, np. napisanie znaku "ż" mogłoby być interpretowane przez aplikację jako kombinacja litery "z" i kropki ".". W takiej sytuacji znajomość umiejscowienia kropki nad literą lub w dolnej części ekranu może rozsądzić problematyczną analizę i dać poprawny rezultat. Podobna sytuacja mogłaby wystąpić w przypadku litery "ć" oraz kombinacji litery "c" i przecinka ",".
* częściowa niewrażliwość na skalowanie (jedynie w odpowiednim zakresie) - podobnie jak w poprzednich przypadkach właściwość ta powinna obowiązywać tylko w ustalonym zakresie. Ma to na celu lepsze rozpoznawanie między sobą małych i dużych liter alfabetu, które w niektórych przypadkach charakteru pisma mogą różnić się jedynie wielkością.
* wydobycie ogólnych cech obiektów - Algorytm nie powinien analizować szczegółowych kształtów konturu a powinien w miarę możliwości aproksymować jego cechy geometryczne.

## 2.2. Wybór metody

Do identyfikacji obiektów będących konturami zdecydowano się wykorzystać kody łańcuchowe Freemana oraz reprezentację cech geometrycznych konturu w postaci histogramu obrazującego rozkład tych kodów. Dodatkowo informację o obiekcie rozszerza również informacja o współrzędnych punktu na ekranie, w którym gest zaczął być rysowany oraz liczba pikseli, z których składa się kontur wyrażająca jego długość. Podsumowując, pełna informacja o cechach geometrycznych pojedynczego gestu składowego, która wystarcza do identyfikacji obiektów, reprezentowana jest i przechowywana w trzech elementach:

* histogram kodów łańcuchowych Freemana,
* współrzędne punktu początku konturu,
* całkowita liczba pikseli konturu.

Przewiduje się, że cała aplikacja z kompletną bazą wzorców będzie przechowywać nawet do kilkuset wzorców po długim czasie użytkowania przez użytkownika. Dlatego też od aplikacji oczekuje się, że będzie rozpoznawać znaki w możliwie najkrótszym czasie tak, aby dać użytkownikowi wrażenie płynnego wprowadzania tekstu. Z tego względu kody łańcuchowe Freemana jako metoda reprezentacji obiektu wydają się być dobrym wyborem z perspektywy małej zajętości w pamięci oraz stosunkowo szybkiego algorytmu porównywania wzorców, gdyż dane histogramu charakteryzujące kształt przechowywane mogą być jako tablica jedynie ośmiu liczb zmiennoprzecinkowych. Dokładny opis algorytmu i zasada działania zostały przedstawione w kolejnych rozdziałach.

opis, jakie metody inni wybrali w innych pracach

## 2.3. Akwizycja konturu

Danymi wejściowymi dla algorytmu rozpoznawania elementarnych gestów jest sekwencja punktów, z jakich składa się kontur. Akwizycja tych danych odbywa się dzięki interakcji z systemem, na jakim pracuje urządzenie. Dane o położeniu palca lub rysika na ekranie dotykowym w systemie Android możliwe są do odczytania tylko co pewien, krótki okres czasu. Oznacza to, że gest wykonany przez użytkownika w sposób szybki może być odczytany z mniejszą dokładnością. Każdy gest poddawany jest aproksymacji (w mniejszym lub większym stopniu), a więc z ekranu dotykowego można odczytać zbiór punktów, który reprezentuje linię łamaną otwartą, gdzie odległość między sąsiednimi punktami nie musi być stała (i zwykle nie jest). Po sygnale oderwania palca od ekranu można łatwo stwierdzić, że rysowanie konturu zostało zakończone i można poddać go dalszej analizie.

## 2.4. Przetwarzanie wstępne

### 2.4.1. Interpolacyjne uzupełnienie listy punktów

Zaraz po odczycie, gest składowy poddawany jest przetwarzaniu wstępnemu. Pierwszy krok polega na uzupełnieniu listy punktów otrzymanej ze sterownika ekranu o brakujące piksele tak, aby lista zawierała tyle pikseli, z ilu faktycznie składa się kontur na ekranie, i aby odległość między sąsiednimi punktami była stała (równa jednemu pikselowi). Liczba uzupełnianych pikseli proporcjonalna jest do odległości między punktami w metryce euklidesowej dla każdej pary sąsiednich punktów. Przekształcenie to odwraca proces aproksymacji konturu i ułatwia dalszą analizę gestu. Do wyznaczenia współrzędnych punktów pomiędzy sąsiednimi pikselami wykorzystywana jest interpolacja liniowa współrzędnych punktów będących najbliższymi sąsiadami. Należy zaznaczyć, że współrzędne interpolowanych pikseli zapisywane są jako liczby zmiennoprzecinkowe a nie jako całkowite, aby przechowywać położenie punktów z większą dokładnością.

### 2.4.2. Filtracja

Kolejnym etapem przetwarzania wstępnego jest filtracja konturu otrzymanego z poprzedniego kroku. Filtracja odbywa się poprzez uśrednianie współrzędnych punktów konturu. Współrzędne punktów konturu zastępowane są nowymi współrzędnymi, które są obliczane jako średnia arytmetyczna współrzędnych jego najbliższych sąsiadów ze starej listy punktów. Wartość liczby sąsiadów została wybrana doświadczalnie i w omawianej aplikacji wynosi 20. Oczywiście w wyniku takiego uśrednienia liczba punktów konturu zmniejszy się o wartość .

Filtracja ma na celu wygładzenie konturu oraz redukcję szumów (drgań rysika podczas rysowania oraz kompensacji błędów niedokładności czujnika dotykowego). Przeprowadzone doświadczenia pokazały, że zwiększa to skuteczność w późniejszym wyznaczaniu wektorów kierunków między pikselami i wpływa na wyraźniejszy rozkład łańcuchów Freemana na histogramie, zmniejszając ilość wystąpień błędnie rozpoznanych kierunków, które wynikają z analizy położenia jedynie dwóch sąsiednich punktów znajdujących się bardzo blisko siebie. Liczba sąsiadujących pikseli, z jakich wyznaczany jest nowy uśredniany punkt wyznaczona została eksperymentalnie i może zależeć od rozdzielczości ekranu oraz od gęstości pikseli na ekranie (liczby DPI).

Poniżej przedstawiono ideę przetwarzania wstępnego na przykładzie prostego konturu oraz efekt poszczególnych etapów. Kropki na rysunku reprezentują punkty przechowywane w liście.



Rysunek ?????. Przykładowy kontur poddany przetwarzaniu wstępnemu: a) punkty odczytane ze sterownika ekranu dotykowego przed przetwarzaniem wstępnym, b) punkty uzyskane w wyniku uzupełnienia i interpolacji, c) lista uśrednionych punktów po etapie filtracji

Dzięki takiemu podejściu z konturu wydobywane są jego ogólne cechy i przybliżony kształt, zaś szczegółowe cechy geometryczne mogą zostać pominięte na skutek filtracji.

## 2.5. Ekstrakcja cech

Do wyodrębnienia cech charakterystycznych obiektu wykorzystano kody łańcuchowe Freemana. W skrócie polega to na tym, że dla każdej pary sąsiednich pikseli z konturu obliczany jest wektor przemieszczenia (różnica współrzędnych) i na tej podstawie wyznaczany jest kąt, jaki tworzy powstały odcinek z niezmienną osią poziomą ekranu (obliczenie za pomocą funkcji ). Następnie wyznaczany jest jeden z 8 kierunków, do których przynależy taki kąt. Szerokość jednego przedziału kątów dla każdego kierunku jest taka sama i jest równa ilorazowi kąta pełnego i liczby wszystkich przedziałów kierunków.



Rysunek ?????. Podział kąta pełnego na 8 przedziałów kierunków do zapisu kodów łańcuchowych

Do zapisania całego konturu w postaci kodów łańcuchowych Freemana potrzebna jest lista liczb całkowitych, z których każda reprezentuje kierunek zmiany współrzędnej punktu w stosunku do poprzedniego piksela i wyraża się liczbą z zakresu od 0 do 7.

Taki sposób reprezentacji ma pożądaną cechę - częściową niewrażliwość na obrót. W niektórych sytuacjach obrót całego wejściowego konturu nawet do 45 stopni może skutkować zapisaniem dokładnie tymi samymi wartościami kodów łańcuchowych (np. w przypadku kształtu linii prostych). Warto zauważyć również, że taki zapis w postaci kodów łańcuchowych jest silnie związany z kierunkiem wykonywania gestu, co jest istotną zaletą, gdyż dzięki temu można łatwo odróżnić wiele gestów, np. pionową linię pisaną w dół lub górę.

## 2.6. Reprezentacja obiektu

Kolejnym etap jest wyznaczenie z listy kodów łańcuchowych Freemana histogramu liczby ich występowania. Polega to oczywiście na zliczeniu ilości wystąpień dla każdego takiego kodu, a następnie podzieleniu tej liczby przez ilość wszystkich kodów występujących w konturze w celu normalizacji histogramu.

Taka reprezentacja obiektu jest bardzo wygodna w dalszej analizie (porównywaniu histogramów). Warto również zauważyć, że takie przedstawienie obiektu ma właściwość całkowitej niewrażliwości na przesunięcia (histogramy kodów łańcuchowych będą dokładnie takie same w wyniku dowolnego przesunięcia konturu) oraz całkowitej niewrażliwości na skalowanie wejściowego konturu (z powodu normalizacji histogramu rozkład procentowy każdego z kierunków będzie również taki sam).

Ze względu na założenie częściowej niewrażliwości na przesunięcie oraz skalowanie w procesie rozpoznawania gestów elementarnych konieczne jest przechowywanie dodatkowych informacji o obiekcie, które mogą być wykorzystane podczas wyznaczania podobieństwa obiektów, a które ograniczą w pewnym stopniu całkowitą niewrażliwość konturu na wybrane przekształcenia geometryczne. Do tego celu razem z histogramem kodów łańcuchowych wyrażającym sam kształt gestu elementarnego dołączana jest również informacja o współrzędnych punktu startowego dla konturu oraz jego długości (wyrażającym wielkość gestu).

## 2.7. Klasyfikacja

Aby dokonać identyfikacji nieznanego gestu elementarnego, należy wyznaczyć współczynnik stwierdzający stopień podobieństwa gestów. W przypadku, gdy istnieje wiele cech obiektów, rozwiązaniem może być wyznaczenie współczynników podobieństwa dla każdej cechy, a następnie obliczenie sumarycznego współczynnika korelacji.

### 2.7.1. Korelacja histogramów

Algorytm identyfikacji pojedynczych gestów wymaga metody porównywania wprowadzonych konturów z konturami przechowywanymi w bazie aplikacji jako wzorce w celu wyznaczenia współczynnika podobieństwa każdego do z nich i wybrania najlepszego dopasowania. Opisywany współczynnik to współczynnik korelacji liniowej Pearsona. Mieści się w zakresie i większa jego wartość oznacza większy stopień podobieństwa porównywanych ciągów liczb. Danymi wejściowymi dla korelacji Pearsona są histogramy kodów łańcuchowych określających kształt gestów. Dzięki temu wynikiem takiego zestawienia histogramów jest liczba określająca jak bardzo kształty gestów są do siebie zbliżone.

Wyznaczenie współczynnika korelacji Pearsona odbywa się poprzez wykorzystanie funkcji z biblioteki OpenCV. Obliczanie trwa bardzo szybko z powodu między innymi bardzo małej ilości danych do przeanalizowania (porównanie 2 tablic o rozmiarze 8) oraz zoptymalizowanego algorytmu zaimplementowanego w bibliotece OpenCV.

### 2.7.2. Korelacja punktów startowych

Aby stwierdzić podobieństwo dotyczące punktów startowych dla konturów, obliczana jest odległość między punktami, następnie dzielona jest przez mniejszy wymiar ekranu (liczbę pikseli szerokości lub wysokości ekranu) tak, aby uzyskać wielkość niezależną od wielkości ekranu urządzenia, na jakim uruchomiona jest aplikacja. Tak uzyskana liczba nazwana jest względną odległością punktów. Wyznaczenie współczynnika korelacji punktów startowych zależne jest również od 2 parametrów wybranych doświadczalnie. Są to współczynniki oraz . Wartość współczynnika określa jaki jest próg tolerancji dla przesunięcia punktów startowych między sobą i nie wpływa ono jeszcze na współczynnik korelacji punktów startowych (uznaje, że rozpoczęcie rysowania gestów odbyło się w tym samym miejscu). Natomiast współczynnik określa jak duża powinna być różnica w odległości punktów między sobą, aby uznać, że nie występuje jakiekolwiek podobieństwo, jeśli chodzi o miejsce rozpoczęcia rysowania. W zakresie względnej odległości punktów między a wartość współczynnika korelacji interpolowana jest w sposób liniowy, co przedstawia się na poniższym wykresie.



Rysunek ?????. Wykres zależności współczynnika korelacji punktów startowych od wartości względnej odległości punktów

Wartości współczynników i zostały wybrane na drodze doświadczalnej i wynoszą odpowiednio: oraz .

### 2.7.3. Korelacja długości gestu

Do wyznaczenia podobieństwa gestów dotyczącego ich wielkości posłużono się współczynnikiem korelacji długości. Do tego celu należy wyznaczyć współczynnik będący względną różnicą długości gestów, wyraża się on wzorem:

gdzie - większa wartość długości konturu spośród porównywanych konturów,   
 - mniejsza wartość długości konturu spośród porównywanych konturów.

Podobnie jak w poprzednim przypadku, współczynnik korelacji długości zależny jest od 2 parametrów wybranych doświadczalnie. Są to współczynniki oraz . Wartość współczynnika określa martwą strefę niewrażliwości na skalowanie (dla algorytm uznaje, że gesty są takiej samej wielkości). Zaś współczynnik określa górną granicę wrażliwości na skalowanie (dla składowe gesty nie wykazują żadnego podobieństwa, jeśli chodzi o wspólną wielkość. W zakresie względnej różnicy długości między a wartość współczynnika korelacji interpolowana jest w sposób liniowy, przebieg jego funkcji przedstawiono na poniższym wykresie:



Rysunek ?????. Wykres zależności współczynnika korelacji długości od wartości względnej różnicy długości

Wartości współczynników i zostały wybrane na drodze doświadczalnej i wynoszą odpowiednio: oraz .

### 2.7.4. Sumaryczny współczynnik korelacji

Jak wcześniej wspomniano, na podstawie współczynników podobieństwa dla każdej cechy obiektu można wyznaczyć sumaryczny współczynnik korelacji określający wypadkową wartość podobieństwa. Taki współczynnik może być prostym iloczynem wszystkich wartości korelacji dla każdej z cech. W takim wypadku sumaryczny wynik jest silnie zależny od każdej z cech, np. gdy jeden ze współczynników składowych jest równy 0, stwierdzony może zostać zupełny brak podobieństwa obiektów.

Przeprowadzone testy pokazały jednak, że iloczyn współczynników korelacji daje stosunkowo słabe rezultaty, a lepsze efekty można osiągnąć, korzystając ze średniej ważonej.

W opisywanej aplikacji wartość sumarycznego współczynnika korelacji wyraża się wzorem na średnią ważoną:

gdzie: - współczynnik korelacji histogramów kodów łańcuchowych Freemana,   
 - współczynnik korelacji punktów startowych, - współczynnik korelacji długości, , , - wagi dla poszczególnych współczynników.

Wartości wag zostały wyznaczone na drodze eksperymentalnej i wynoszą odpowiednio:

, ,

W algorytmie istnieje również minimalny współczynnik korelacji dla pojedynczego gestu, wyrażający wartość współczynnika korelacji, poniżej której nie stwierdza się żadnego podobieństwa między elementarnymi gestami. Wynosi on .

## 2.8. Przypadek krótkich konturów

Istnieje przypadek typu gestu elementarnego wykonywanego przez użytkownika, dla którego powinno się zdefiniować inne, wyjątkowe zachowanie algorytmu, gdyż wyżej opisany algorytm może nie sprawdzić się dla tego przypadku. Jest to przypadek wprowadzania przez użytkownika kropki - pojedynczego kliknięcia na ekranie dotykowym. Sytuacja ta może mieć miejsce np. przy wprowadzaniu liter "i", "j", "ż", znaków interpunkcyjnych kropki, dwukropka lub znaku zapytania.

Wyżej opisany algorytm może niepoprawnie rozpoznawać kropki, gdyż wyznaczenie kodów łańcuchowych Freemana na podstawie jednego piksela może być niemożliwe, lub w przypadku bardzo małej liczby punktów może zwracać losowe kierunki między sąsiednimi punktami, tym samym zwracając prawie losowy rozkład na histogramie. Poza tym filtracja w trakcie przetwarzania wstępnego może być niemożliwa z powodu zbyt małej liczby pikseli uniemożliwiającej uśrednianie punktów.

Dlatego też w przypadku wpisania kropki algorytm zachowuje się nieco inaczej. Wprowadzenie kropki jest bardzo łatwe do wykrycia dzięki informacji o długości konturu. Jeśli liczba pikseli, z jakich składa się kontur jest mniejsza lub równa pewnej liczbie granicznej (w aplikacji wynosi ona 20), generowany jest histogram o rozkładzie równomiernym. Również w przypadku korelacji długości, gdy dwa kontury mają długość mniejszą lub równą liczbie granicznej, automatycznie rozpoznawane są jako gest kropki, ich korelacja histogramów oraz korelacji długości wynosi 1 (sposób obliczania korelacji punktów startowych pozostaje bez zmian). Takie rozwiązanie znacznie poprawia skuteczność w rozpoznawaniu tego typu gestów.

## 2.9. Schemat algorytmu

Należy pamiętać, że opisywany algorytm stosowany może być jedynie do rozpoznawania podstawowego gestu, kiedy w bazie aplikacji znajdują się już zapamiętane wzorce. W kolejnych rozdziałach opisywana jest metoda, która jest modyfikacją tego algorytmu i rozszerza go na rozpoznawanie gestów złożonych (składających się z wielu elementarnych gestów).

Podsumowaniem zasady działania algorytmu rozpoznawania pojedynczych gestów jest schemat blokowy przedstawiony poniżej.



Rysunek ?????. Schemat blokowy algorytmu rozpoznawania pojedynczych gestów.

## 2.10. Przykłady

Aby zilustrować użytą metodę reprezentacji cech geometrycznych obiektu w postaci histogramu kodów łańcuchowych Freemana, poniżej przedstawiono wykresy histogramów dla przykładowych konturów.



Rysunek ?????. Histogram dla gestu litery "a"



Rysnek ?????. Histogram dla gestu składowego litery "o", w przypadku idealnego okręgu rozkład kierunków na histogramie byłby równomierny



Rysunek ?????. Histogram dla gestu litery "I", według histogramu kontur składa się wyłącznie z wektorów skierowanych pionowo w dół  
(kierunek o indeksie 6)



Rysunek ?????. Histogram dla gestu litery "L", na histogramie widoczne jest występowanie wektorów skierowanych w dół (6) oraz w prawo (0)

Aby pokazać jak istotną funkcję pełni filtracja w procesie przetwarzania wstępnego, poniżej przedstawiono dwa histogramy wygenerowane na podstawie tego samego konturu, lecz w jednym z nich nie zastosowano filtracji. Łatwo można zauważyć, że histogram utworzony z konturu niepoddanego filtracji ma większą wrażliwość na zakłócenia.



Rysunek ?????. Zestawienie kształtu konturu (a), histogramu utworzonego z konturu niepoddanego filtracji (b), histogramu utworzonego z wykorzystaniem filtracji (c)

# 3. System rozpoznawania złożonych gestów

Rozdział ten skupia się na opisie zaprojektowanej metody rozpoznawania złożonych gestów, które składają się z wielu pojedynczych fragmentów.

## 3.1. Opis algorytmu

Stworzony algorytm jest modyfikacją metody rozpoznawania pojedynczych konturów z poprzedniego rozdziału. Tu również wykorzystuje się współczynnik korelacji w celu porównywania gestów, natomiast wprowadzone kontury nie są porównywane ze wszystkimi wzorcami, gdyż nie ma takiej potrzeby. Identyfikacja nie polega jedynie na znalezieniu obiektu z maksymalnym współczynnikiem korelacji (tak jak to miało miejsce w poprzednim przypadku), lecz wiąże się to z głębszą analizą wzorców (będących kombinacją pojedynczych konturów o różnej ilości gestów składowych) oraz z koniecznością rozważania wielu możliwości gestów spełniających ustalone kryteria, jak również odrzucania ich na bieżąco w wyniku wprowadzenia kolejnych gestów. W przypadku, gdy zostanie wiele takich możliwości, algorytm musi podjąć decyzję, które z tych rozwiązań będzie najlepsze. Kolejność wprowadzania konturów przez użytkownika jest bardzo istotna dla algorytmu i powinna być dokładnie taka sama, jak w zapamiętanym wzorcu.

Struktura bazy wzorców również musi być inna w przypadku identyfikacji gestów złożonych. Wiąże się to z koniecznością grupowania wielu konturów i przechowywania listy wielu gestów elementarnych dla każdego wzorca będącego gestem złożonym wraz z informacją o przynależności danego obiektu do klasy (w przypadku rozpoznawania znaków sprowadza się to do przechowywania znaku, jaki odpowiada wzorcowi).

### 3.1.1. Kryteria wyboru

Przy wyborze najbardziej zbliżonego wzorca algorytm kieruje się następującymi kryteriami:

1. Próg współczynnika korelacji - Wszystkie pojedyncze gesty pochodzące z wzorca w porównaniu z odpowiadającymi im konturami wejściowych powinny spełniać warunek współczynnika korelacji większego lub równego minimalnemu współczynnikowi korelacji. Jest to warunek konieczny - niespełnienie tego założenia dla przynajmniej jednego gestu elementarnego skutkuje natychmiastowym odrzuceniem wzorca i zaprzestaniem analizowania jego kolejnych elementów.

2. Największy wypadkowy współczynnik korelacji oraz największa złożoność gestu - Najlepsze rozwiązanie spośród wszystkich rozważanych możliwości wybierane jest przy pomocy dwóch kryteriów jednocześnie. Pod uwagę brany jest jak największy współczynnik korelacji dla całego gestu (wypadkowy), jak i liczba konturów wchodzących w skład gestu. Gdyby program nie stosował kryterium największej złożoności gestu, mogłoby się okazać, że np. po wpisaniu litery "Ł" (jako dwa kontury składowe: litery "L" i kreski"), aplikacja rozpoznałaby samą literę "L" oraz zwróciła błąd nierozpoznania kreski. Dlatego też należy w takich przypadkach zwiększyć wpływ złożoności wynik identyfikacji dla wieloelementowych gestów. Oczywiste jest, że nie powinno się wybierać jedynie gestów z największą złożonością, ale również rozważać najlepiej pasujące wzorce.

### 3.1.2. Dane wejściowe

Uruchomienie algorytmu odbywa się po każdorazowym wprowadzeniu gestu elementarnego przez użytkownika. W pierwszym kroku należy pobrać listę konturów wejściowych, które jeszcze nie zostały rozpoznane. Oczywiście wiąże się to z zapamiętywaniem historii wejściowych gestów elementarnych w aplikacji wraz z informacją o dokonaniu ich przeanalizowania lub nie. Takie kontury wraz z zestawem wzorców są danymi wejściowymi dla algorytmu.

Jeśli lista konturów wejściowych jest pusta, algorytm kończy działanie nie zwracając żadnego wyniku. Natomiast jeżeli lista wzorców jest pusta, zwracany jest błąd nierozpoznania żadnego gestu. Do dalszej analizy wymagany jest zatem niepusty zbiór wzorców i lista wejściowych konturów.

### 3.1.3. Wstępna klasyfikacja

W kolejnym kroku każdy wzorzec z bazy aplikacji jest wstępnie klasyfikowany. Sprawdzane jest, czy spełnione jest kryterium przekroczenia progu minimalnego współczynnika korelacji (kryterium nr 1). Współczynnik ten jest wyznaczany metodą z poprzedniego rozdziału o wyznaczaniu podobieństwa pojedynczych gestów poprzez dokonanie porównania pierwszego konturu pochodzącego z wzorca oraz odpowiadającemu mu konturowi wejściowemu. Jeśli kryterium nie jest spełnione, wzorzec jest odrzucany i analizowany jest kolejny. Jeśli natomiast zachodzi warunek współczynnika korelacji większego lub równego ustalonemu minimum, analizowany jest kolejny kontur z gestu (o kolejnym indeksie) w ten sam sposób. Jeżeli okaże się, że nie można dokonać dalszej analizy, gdyż wprowadzona liczba konturów jest mniejsza niż złożoność wzorca a istnieje możliwość, że właśnie ten wzorzec może posłużyć do identyfikacji obiektu, to algorytm przerywa działanie, gdyż nie ma wystarczających danych, aby podjąć decyzję. Algorytm wykona się ponownie w następnym kroku, gdy użytkownik wprowadzi kolejny kontur, wtedy analiza zostanie wznowiona. Wyjątkowym przypadkiem jest jednak przekroczenie czasu przez użytkownika pomiędzy wpisywaniem kolejnych konturów (wpisanie znaku "L" i odczekanie przez użytkownika pewnego czasu nie wstrzyma pracy algorytmu, czekając na brakującą kreskę i rozpoznanie jako literę "Ł", ale pozwoli mu kontynuować jego działanie i odrzucenie możliwości dłuższych gestów).

### 3.1.4. Wypadkowy współczynnik korelacji

Jeśli wszystkie składowe elementy gestu spełniają kryterium nr 1, wzorzec dopisywany jest do tymczasowej listy możliwości. Z wielu współczynników korelacji, które, które zostały wyznaczone dla każdego konturu (tyle razy, ile wynosiła złożoność gestu) należy obliczyć współczynnik, który będzie określał stopień podobieństwa całego gestu złożonego. Współczynnik ten jest równy średniej arytmetycznej współczynników korelacji dla konturów, z których składał się gest. Liczba ta posłuży w kolejnym kroku do wyboru najlepszego rozwiązania.

### 3.1.5. Wybór najlepszego rozwiązania

Efektem zakończonej wstępnej klasyfikacji dla wszystkich wzorców jest lista potencjalnych możliwości zawierająca informacje o wzorcach i przebiegu ich wstępnej klasyfikacji. Jeśli ta lista jest pusta, algorytm zwraca błąd nierozpoznania żadnego gestu oraz zapisuje fakt dokonania analizy ostatnio wpisanego konturu (aby nie był analizowany następnym razem). Jeżeli zaś na liście możliwości znajdują się wzorce, można przystąpić do wyboru najlepszego wzorca. Polega to na prostym znalezieniu wzorca, dla którego liczba wyrażająca ocenę rozwiązania jest największa. Oczywiście konieczne jest zdefiniowanie, czym jest wprowadzona ocena rozwiązania. Jest to liczba, która wyznaczana jest na podstawie wypadkowego współczynnika korelacji oraz złożoności gestu. Ma to na celu jednoczesne rozważanie obu tych cech (zgodnie z kryterium nr 2). Ocena rozwiązania reprezentuje obie cechy w jednej liczbie (co bardzo ułatwia znalezienie jej maksimum) oraz pozwala wpłynąć na proces rozpoznawania poprzez dobór odpowiednich współczynników. Daje to możliwość ustalenia, w jakim stopniu mają być faworyzowane gesty o większej złożoności, a w jakim stopniu wpływ na wybór najlepszego rozwiązania ma mieć sam współczynnik korelacji.

Ocena potencjalnego rozwiązania dla wzorca wyraża się wzorem:

gdzie: - ocena rozwiązania, - wypadkowy współczynnik korelacji, - złożoność gestu (liczba jego elementarnych konturów), - współczynnik dominacji złożoności (im większy jest współczynnik, tym większy wpływ na wybór rozwiązania ma złożoność gestu). Wartość współczynnika została dobrana na drodze empirycznej i wynosi .

Po wyznaczeniu oceny rozwiązania dla każdej rozważanej możliwości wybierany jest wzorzec o maksymalnej wartości tej oceny i zwracany jest jako wynik rozpoznania wprowadzonego gestu (a dokładnie znak, jaki reprezentował wzorzec lub akcja, jaka była mu przypisana). Ostatnio wprowadzone gesty zostają zaznaczone jako przeanalizowane, aby nie zostały przetworzone przy następnej iteracji (liczba zaznaczonych gestów jest równa złożoności rozpoznanego wzorca). Wzorzec zostaje dopisany do listy rozpoznanych gestów, a algorytm uruchamiany jest ponownie, gdyż na liście wejściowej mogą znajdować się jeszcze nierozpoznane gesty. Powtarzanie wykonywania trwa dopóki algorytm zwraca pomyślnie rozpoznane wzorce.

## 3.2. Schemat algorytmu

Poniższe schematy blokowe przedstawiają zasadę działania algorytmu rozpoznawania złożonych gestów.



Rysunek ?????. Schemat blokowy algorytmu rozpoznawania złożonych gestów - część 1.



Rysunek ?????. Schemat blokowy algorytmu rozpoznawania złożonych gestów - część 2.

## 3.3. Przykłady

Aby zilustrować działanie programu, przeanalizujmy pewien przypadek, kiedy w bazie aplikacji są już zdefiniowane pewne wzorce:

Tabela 1. Zbiór przykładowych wzorców w bazie aplikacji

|  |  |  |  |
| --- | --- | --- | --- |
| **Wzorzec - gest złożony** | **Gest składowy nr 1** | **Gest składowy nr 2** | **Gest składowy nr 3** |
| mała litera "l", złożoność 1 |  | brak | brak |
| duża litera "I"  złożoność 1 |  | brak | brak |
| mała litera "ł",  złożoność 2 |  |  | brak |
| mała litera "i",  złożoność 2 |  |  | brak |
| mała litera "f",  złożoność 2 |  |  | brak |
| mała litera "k",  złożoność 3 |  |  |  |

Rozważmy zachowanie programu, gdy użytkownik wprowadza znak małej litery "l". Zakończenie rysowania konturu (oderwanie palca lub rysika od ekranu) powoduje wykonanie algorytmu, który jako dane wejściowe dostaje listę zawierającą jeden kontur. Program przegląda bazę wszystkich wzorców i wstępnie klasyfikuje je, porównując wprowadzony kontur z pierwszym gestem składowym. W wyniku takiego porównania otrzymany współczynnik korelacji będzie większy od minimalnego współczynnika dla wszystkich wzorców oprócz litery "f". Zatem ten wzorzec zostanie natychmiast odrzucony. Z pozostałych pięciu wzorców, trzy z nich mają złożoność większą niż 1, lecz nie można dokonać porównania kolejnych gestów składowych (o indeksie 2), gdyż nie istnieje kolejny kontur wejściowy. Z tego powodu algorytm nie ma wystarczających danych, aby podjąć decyzję i kończy swoje działanie. Jeśli użytkownik nie wpisze kolejnych konturów i czas oczekiwania na kolejny kontur minie, algorytm zostanie wywołany ponownie. Tym razem nie będzie brał pod uwagę dłuższych wzorców i odrzuci trzy wzorce o złożoności większej niż 1 (litery "ł", "i" oraz "k"), które wcześniej były brane pod uwagę. Do listy potencjalnych rozwiązań zostaną dodane 2 wzorce: mała litera "l" oraz duża litera "I". W kolejnym kroku zostanie wybrane najlepsze rozwiązanie z listy możliwości. Z racji tego, że oba wzorce mają jednakową złożoność, wybrany zostanie wzorzec z większym współczynnikiem korelacji, którym najprawdopodobniej będzie litera "l". W ten sposób znak wpisany przez użytkownika został poprawnie rozpoznany. Po pomyślnej identyfikacji algorytm uruchomi się ponownie w poszukiwaniu kolejnych gestów, lecz zostanie od razu zakończony, gdyż lista nierozpoznanych konturów będzie pusta.

Rozważmy teraz przypadek, kiedy użytkownik wprowadza do programu znak małej litery "ł". Robi to poprzez wykreślenie dwóch konturów na ekranie. Wprowadzenie pierwszego konturu (podobnego do litery "l") skończy się brakiem podjęcia akcji, oczekując na więcej danych wejściowych (tak, jak to opisano rozważając poprzedni przypadek). Natomiast wprowadzenie kolejnego konturu przez użytkownika (brakującej kreski nad literą "ł") skutkuje wznowieniem analizy wzorców i przeprowadzeniem kolejnego porównania gestów składowych nr 2 z drugim wprowadzonym konturem dla każdego rozważanego wzorca. Po takim porównaniu odrzucone zostaną wzorce małej litery "i" oraz litery "k" z powodu zbyt małego współczynnika korelacji. Zatem na liście potencjalnych rozwiązań znajdą się trzy wzorce: małej litery "l", dużej litery "I" oraz małej litery "ł". Algorytm nie musi oczekiwać na kolejne kontury, gdyż nie rozważa już bardziej złożonych gestów. Następnie każde z tych rozwiązań zostanie poddane ocenie. W związku z wprowadzoną zasadą obliczania oceny rozwiązania, faworyzowane będą gesty z większą złożonością, a więc to litera "ł" zostanie najprawdopodobniej rozpoznana (pod warunkiem występowania zbliżonych wartości współczynników korelacji dla innych wzorców).

# 4. Inteligencja systemu

W tym rozdziale poruszone zostanie zagadnienie zdolności systemu do samouczenia się oraz dostosowywania się do zmiennych warunków. Stworzony system został zaprojektowany tak, aby podejmował interakcję z użytkownikiem w pewnym sytuacjach, oraz aby sam zmieniał swoje zachowanie na skutek poleceń wydawanych przez użytkownika.

Zachowanie systemu zostało zaprojektowane tak, aby aplikacja z początkowo pustą bazą wzorców w niedługim czasie automatycznie uzupełniła bazę wzorcami, dostosowała się do charakteru pisma odręcznego użytkownika i uzyskała pełną funkcjonalność z zadowalającą skutecznością w rozpoznawaniu znaków.

System można nazwać inteligentnym również dlatego, że podczas identyfikacji gestów rozważa wiele możliwości, ocenia rozwiązania i wybiera to, które według niego jest najlepsze, a zatem w pewnym sensie dokonuje procesu wnioskowania.

## 4.1. Problem błędnych wzorców

Zastosowana w rozpoznawaniu gestów metoda najbliższego sąsiada ma pewne wady. Jedną z nich jest wrażliwość na błędne dane w ciągu uczącym. Jeśli jeden obiekt zostanie źle zaklasyfikowany i zapamiętany jako wzorzec, może to prowadzić do błędnego klasyfikowania całego jego otoczenia, przyczynić się do niepoprawnego podejmowania decyzji w pewnym obszarze przestrzeni cech. Jest to poważny problem, skutkiem zapamiętania błędnego wzorca może być znaczny spadek skuteczności dla pewnych gestów. Trudne jest również zlokalizowanie takiej próbki i usunięcie przyczyny błędów.

Błędny wzorzec może się pojawić w wyniku pomyłki użytkownika lub np. zbyt dużych zakłóceń i zniekształceń konturu. Stworzony inteligentny interfejs dotykowy podejmuje próby wykrywania takich wzorców i usuwania ich.

## 4.2. Rejestrowanie bilansu identyfikacji

Aby móc stwierdzić, które wzorce w bazie aplikacji są najczęściej podstawą do identyfikacji innych gestów, a które prowadzą do błędnych decyzji, potrzebne są dodatkowe informacje. W tym celu dla każdego wzorca rejestrowane jest każde jego poprawne rozpoznanie (gdy w procesie rozpoznania wzorzec ten był najlepszym dopasowaniem) oraz każde błędne rozpoznanie. Informacje te są przechowywane w dwóch liczbach całkowitych, co pozwala również na łatwe wyznaczenie sumy ilości wszystkich rozpoznań oraz obliczenie tzw. bilansu identyfikacji, który wyraża się jako różnica ilości poprawnych oraz niepoprawnych rozpoznań. Ujemna wartość takiego bilansu pozwala wnioskować, że wzorzec może być nieprawidłowy i powinien zostać usunięty.

Stwierdzanie poprawnego lub błędnego rozpoznania dla wzorca odbywa się automatycznie. Każda identyfikacja wzorca zawsze początkowo uważana jest za poprawną, a liczba poprawnych rozpoznań zwiększana o 1. Użytkownik oprócz wpisywania znaków ma do dyspozycji również możliwość wykonania akcji poprzez naciśnięcie przycisków "Cofnij" oraz "Popraw". Wybór jednego z dwóch przycisków oznacza, że wzorzec posłużył do błędnej identyfikacji, a więc cofa to wykonaną początkowo klasyfikację rozpoznania jako poprawne i liczba poprawnych rozpoznań zmniejsza jest o 1, zaś liczba błędnych rozpoznań jest inkrementowana. Takie podejście pozwala automatycznie aktualizować dodatkowe informacje o wzorcach i w łatwy sposób stwierdzić, jaka jest przydatność danego wzorca.

## 4.3. Automatyczne usuwanie błędnych wzorców

Usuwanie błędnych wzorców w aplikacji odbywa się automatycznie. Jako błędny wzorzec uznawany jest ten, dla którego wartość bilansu identyfikacji przekracza pewien ustalony próg. Próg ten został wybrany doświadczalnie a jego wartość wynosi . W praktyce oznacza to, że z bazy aplikacji usuwane są te wzorce, których liczba błędnych rozpoznań jest większa o 3 od liczby poprawnych rozpoznań. Sprawdzenie takie odbywa się po każdym usunięciu znaku lub poprawieniu go przez użytkownika. Wymaga to również przechowywania informacji o tym, poprzez który wzorzec został rozpoznany każdy z wprowadzonych znaków.

## 4.4. Automatyczne dodawanie wzorców

Aby system uczył się w trakcie działania, i aby zwiększał swoją skuteczność w rozpoznawaniu, zastosowano automatyczne dodawanie wzorców. Polega to na tym, że nowo wprowadzony gest, który został poprawnie rozpoznany, automatycznie staje się nowym wzorcem. Aby lista wzorców nie rozrastała się zbyt szybko i niepotrzebnie, proces automatycznego dodawania wzorców odbywa się tylko dla gestów, które zostały rozpoznane ze stosunkowo niskim współczynnikiem korelacji. Wymaga to zdefiniowania wartości progu decydującego o dodaniu nowego wzorca lub nie. Wartość tego progu wynosi w aplikacji .

### 4.4.1. Postępowanie z automatycznie dodanymi wzorcami

Podejście przedstawione w poprzednim akapicie wymaga nieco innego zachowania aplikacji w przypadku poprawiania lub usuwania gestów przez użytkownika. Źródłem błędów może stać się niepoprawne rozpoznanie wzorca, a następnie dodanie kolejnego gestu (również błędnego) w wyniku automatycznego dodawania wzorców. Aby tego uniknąć, należy zapamiętywać informację o tym, które wzorce zostały dodane automatycznie i usuwać je natychmiastowo w przypadku poprawienia lub usunięcia wpisanego gestu przez użytkownika.

## 4.5. Optymalizator liczby wzorców

Wprowadzenie automatycznego dodawania wzorców mogłoby wpłynąć na znaczny rozrost bazy wzorców a w efekcie przyczynić się do spadku wydajności całej aplikacji z powodu konieczności wykonywania większej liczby obliczeń oraz przechowywania większej liczby danych.

Aby temu zapobiec wprowadzono mechanizm, który czuwa nad ograniczaniem liczebności wzorców dla każdej klasy obiektów. W przypadku rozpoznawania znaków, klasami są poszczególne znaki. Do każdej klasy przynależeć może wiele obiektów (każdy znak może posiadać wiele wzorców). Zatem maksymalna liczba wzorców przynależących do jednego znaku jest ograniczona. W aplikacji to ograniczenie wynosi . Pojawia się problem podjęcia decyzji, które wzorce usuwać, a które pozostawiać w bazie. Otóż usuwane są te nadmiarowe wzorce, które mają najmniejszy bilans porównań. Zaś spośród tych wzorców, dla których liczba ta jest równa, pozostawiane są wzorce nowsze (można to stwierdzić, gdyż zapamiętywany jest czas dodania wzorca).

Efektem takiego podejścia jest usuwanie nadmiarowych obiektów ciągu uczącego, a więc wzorców, które są rzadko podstawą do poprawnego rozpoznania gestu, lub które stają się przyczyną błędnych rozpoznań. Wprowadzony mechanizm optymalizacji liczby wzorców zawsze pozwala na dodanie kolejnych wzorców w trakcie działania aplikacji (nawet, gdy przekroczy to maksymalną liczbę wzorców dla klasy). Samo usuwanie nadmiarowych obiektów odbywa się dopiero przy wyjściu z aplikacji, kiedy to zapisywana jest obecna lista wzorców. Pozostawianie w bazie próbek najlepszych (lub najnowszych) sprawia, że system uczy się na bieżąco i dostosowuje się do aktualnego pisma użytkownika. Nawet w przypadku zmiany użytkownika korzystającego z aplikacji interfejs dotykowy powinien po pewnym czasie przestawić się na nowy charakter pisma.

## 4.6. Interakcja z użytkownikiem

Użytkownik oprócz możliwości wprowadzania konturów na ekranie ma do dyspozycji dwa przyciski pozwalające wydawać pewnego rodzaju polecenia uczące dla aplikacji. Są to przyciski "Cofnij" oraz "Popraw". Przycisk "Cofnij" usuwa ostatnio rozpoznany gest i zapisuje fakt błędnego rozpoznania gestu w odpowiednim wzorcu, ale nie dodaje nowego wzorca do bazy. Zaś przycisk "Popraw" zastępuje ostatnio rozpoznany znak nowym i dodaje nowy wzorzec odpowiadający podanemu znakowi (wymaga to wyświetlenia użytkownikowi odpowiedniego ekranu).

W przypadku, gdy algorytm rozpoznawania złożonych gestów nie rozpozna żadnego obiektu, system podejmie interakcję z użytkownikiem i wyświetli ekran z pytaniem o to, jaki gest został ostatnio wpisany (jaki znak reprezentował i jaką miał złożoność). Podanie i zatwierdzenie danych skutkuje skopiowaniem odpowiedniej liczby wprowadzonych konturów i dopisaniem kolejnego wzorca do wewnętrznej bazy aplikacji. Można powiedzieć, że w ten sposób system uczy się poprzez zdawanie pytań użytkownikowi w sytuacjach, w których jest jeszcze niezdolny do samodzielnego podjęcia decyzji.

Wygląd ekranów widocznych dla użytkownika zostanie przedstawiony w kolejnych rozdziałach.

# 5. Implementacja oprogramowania na urządzenie mobilne

Podczas implementacji oprogramowania zastosowano przedstawioną w poprzednich rozdziałach metodę rozpoznawania gestów złożonych oraz wprowadzono opisane wcześniej inteligentne zachowania systemu.

## 5.1. Urządzenie i środowisko programowania

Urządzeniem, na którym została zainstalowana aplikacja jest smartphone Samsung Galaxy Grand Prime G531F. Posiada ekran dotykowy z wyświetlaczem o rozdzielczości z zagęszczeniem pikseli równym PPI. Urządzenie mobile posiada pamięci RAM, natomiast jednostką obliczeniową jest czterordzeniowy procesor z częstotliwością taktowania . Taki sprzęt pozwala to na wykonywanie dużej ilości obliczeń w krótkim czasie. Systemem operacyjnym, na którym pracuje urządzenie jest system Android w wersji .

Aplikacja została napisana w obiektowym języku programowania Java na platformę Android. Do tego celu posłużono się darmowym środowiskiem programistycznym Android Studio w wersji . Środowisko to pozwala na testowanie aplikacji na wirtualnym urządzeniu z systemem Android oraz na instalację i uruchamianie aplikacji na zewnętrznym urządzeniu podłączonym do komputera przez kabel USB. Przykładowy zrzut ekranu środowiska programistycznego podczas pracy nad aplikacją znajduje się poniżej.



Rysunek ???. Zrzut ekranu środowiska programistycznego Android Studio

## 5.2. Struktura aplikacji

W strukturze powstałej aplikacji można wydzielić pewne moduły odpowiedzialne za poszczególne elementy aplikacji, które z kolei zawierają w sobie klasy lub dzielą się na mniejsze podmoduły. Poniżej przedstawiono hierarchiczną strukturę najważniejszych modułów (paczek Javy) oraz **klas** w aplikacji razem z opisem ich odpowiedzialności za poszczególne elementy.

* moduł logiki aplikacji:
* moduł gestów:
* moduł rozpoznawania:
* **GestureRecognizer** - jest odpowiedzialny za zarządzanie procesem rozpoznawania gestów, uruchamianie algorytmu rozpoznawania złożonych gestów oraz interpretację zwracanych przez niego wartości;
* **Correlator** - wyznacza poszczególne współczynniki korelacji dla pojedynczych gestów;
* **GesturePossibility** - klasa reprezentująca tymczasowe potencjalne rozwiązanie w procesie identyfikacji złożonych gestów. Każdy taki obiekt przechowuje referencje do wzorca oraz wyznaczone współczynniki korelacji dla poszczególnych gestów składowych; **RecognizedGesture** - klasa obiektu będącego rozpoznanym gestem, zawiera informacje o tym, jaki wzorzec posłużył do jego rozpoznania, jakie kontury zostały wprowadzone przez użytkownika, a także czy gest został dodany automatycznie i referencję do niego;
* moduł wzorców:
* **ComplexGesture -** klasa wzorca, przechowuje listę pojedynczych gestów, z których składa się gest złożony, znak lub akcję, jaka jest mu przyporządkowana, identyfikator w bazie (w celu odróżnienia go spośród wielu), czas dodania wzorca oraz liczbę poprawnie i błędnie zaklasyfikowanych gestów na jego podstawie.
* **SamplesContainer** - kontener zawierający listę wzorców, obiekt tej klasy jest serializowany i służy głównie do zapisu danych o wzorcach w systemie plików
* **SamplesCollector -** Optymalizator liczby próbek, jest odpowiedzialny za ograniczanie liczebności wzorców w bazie i usuwanie nadmiarowych próbek.
* moduł pojedynczych gestów:
* **SingleGesture** - reprezentacja wydobytych cech z pojedynczego konturu, zawiera współrzędne punktu startowego, długość oraz histogram kodów łańcuchowych Freemana;
* **Track -** Nieprzetworzony kontur, lista punktów pochodzących z ekranu dotykowego
* **Point -** klasa reprezentacji punktu na płaszczyźnie, posiada dwie współrzędne: x i y;
* **FreemanHistogram -** klasa przedstawiająca rozkład kodów łańcuchowych Freemana na histogramie, jest reprezentowana przez jednowymiarową tablicę. Klasa posiada również metody do obliczania histogramu na podstawie konturu oraz normalizacji histogramu.
* **GestureManager** - klasa zarządza listą wzorców aplikacji, jest odpowiedzialna m.in. za automatyczne dodawanie, usuwanie wzorców oraz odczyt i zapis do bazy aplikacji. Przechowuje listę rozpoznanych i nierozpoznanych gestów.
* **InputGesture** - reprezentuje wpisany przez użytkownika elementarny gest, zawiera informację o konturze oraz flagę oznaczającą, czy obiekt został poddany analizie przez algorytm identyfikacji.
* **TouchPanel** - kontroler panelu dotykowego, monitoruje na bieżąco położenie rysika lub palca na ekranie i tworzy listę punktów konturu. Rejestruje zdarzenia przyciśnięcia, przesunięcia po panelu oraz oderwania od ekranu.
* **Engine** - główny obiekt decyzyjny. Zarządza logiką aplikacji na najwyższym poziomie. Definiuje zachowanie interfejsu oraz zarządza elementami na niższym poziomie.
* **ButtonsManager** - obiekt zarządzający przyciskami oraz wykonujący odpowiednie akcje w wyniku naciśnięcia przez użytkownika.
* **Statistics** - odpowiada za zbieranie danych statystycznych dotyczących procesu rozpoznawania gestów w celu wyznaczenia m.in. skuteczności całego systemu.
* **Graphics** - główny element odpowiedzialny za grafikę aplikacji. Jego zadaniem jest wyświetlanie na ekranie odpowiednich ekranów interfejsu graficznego.
* moduł funkcji systemowych:
* **Files** - służy do odczytu i zapisu w systemie plików;
* **InputManager** - odpowiada za wyświetlanie i obsługę klawiatury ekranowej oraz przygotowanie ekranów formularzy do wprowadzania danych przez użytkownika (liczb lub znaków)
* **Output** - klasa służy do raportowania błędów, wyświetlania komunikatów na ekranie i ich zapisu do dziennika zdarzeń, odpowiedzialna jest za zgłaszanie wystąpienia błędów i wysyłanie ich do środowiska programistycznego w celu zlokalizowania przyczyny niepoprawnego działania aplikacji.
* **TimerManager** - klasa odpowiadająca za odmierzanie czasu rzeczywistego, używana jest do uruchamiania czasomierzy i odmierzania określonego czasu oczekiwania.
* moduł ustawień:
* **Preferences** - klasa odpowiedzialna za odczyt i zapis własnych ustawień użytkownika na urządzeniu (np. lokalizacji pliku z bazą wzorców)
* **Config** - klasa przechowująca wartości większości parametrów przyjętych w aplikacji. W przypadku zaistnienia potrzeby modyfikacji jakiegoś parametru, wystarczy zmienić wartość zmiennej w tej klasie.
* **MainActivity** - główna aktywność aplikacji, odpowiada za uruchomienie aplikacji oraz wykonanie akcji w wyniku komunikatów pochodzących z systemu operacyjnego.

klasa statystyki w logice

## 5.3. Obsługa aplikacji

System posiada dwa główne tryby aplikacji, są to tryb menedżera gestów oraz tryb szybkiego pisania. Tryb menedżera gestów służy do zarządzania bazą wzorców (dodawania nowych wzorców, usuwania, czyszczenia całej listy wzorców) oraz wyświetlania wykresu histogramu łańcuchów Freemana ostatnio wpisanych konturów. Drugi tryb - szybkiego pisania wykorzystywany jest do ciągłego rozpoznawania gestów i zamiany ich na tekst (ciąg znaków). Umożliwia również poprawianie wstawionych znaków, kopiowanie rozpoznanego tekstu do schowka systemowego lub wyczyszczenie całego rozpoznanego tekstu. Przykładowy wygląd interfejsu użytkownika dla każdego trybu zamieszczono na poszczególnych zrzutach ekranu.



Rysunek ???. Zrzut ekranu aplikacji w trybie menedżera gestów



Rysunek ???. Zrzut ekranu aplikacji w trybie szybkiego pisania

### 5.3.1. Interfejs użytkownika

W każdym z dwóch trybów na górze ekranu znajduje się lista przycisków, która różni się dostępnością niektórych przycisków w zależności od trybu.

Przyciski w aplikacji wykonują następujące funkcje:

* **Zakończ** - zapisuje aktualną listę wzorców do pliku i zamyka aplikację;
* **Czyść komunikaty** - Usuwa z ekranu widoczne komunikaty;
* **Szybkie pisanie** / **Manager gestów** - przełącza między trybami aplikacji;
* **Zapisz wzorzec** - wyświetla odpowiedni ekran służący do wprowadzenia znaku, jaki ma być reprezentowany przez gest oraz złożoności zapisywanego gestu (ilości ostatnio wprowadzonych konturów)
* **Lista wzorców** - wyświetla komunikat o liczbie wszystkich wzorców, ich nazwach oraz przyporządkowanym znakom;
* **Lokalizacja wzorców** - wyświetla odpowiedni ekran i pozwala modyfikować ścieżkę położenia pliku zawierającego bazę wzorców;
* **Usuń wzorzec** - pozwala usunąć wybrany gest z bazy poprzez wskazanie jego nazwy;
* **Wyczyść wzorce** - usuwa wszystkie wzorce z listy;
* **Cofnij** - usuwa ostatni gest, zapamiętuje wystąpienie błędnego rozpoznania przez dany wzorzec;
* **Popraw** - usuwa ostatni gest (lub więcej w zależności od złożoności poprawionego gestu), zapamiętuje wystąpienie błędnego rozpoznania przez dany wzorzec oraz wyświetla ekran dodawania nowego wzorca;
* **Kopiuj do schowka** - kopiuje aktualnie rozpoznany łańcuch znaków do systemowego schowka, co pozwala na wykorzystanie wpisanego tekstu w innych aplikacjach na urządzeniu;
* **Wyczyść tekst** - czyści zawartość aktualnie rozpoznanego tekstu.

Oprócz przycisków użytkownik ma oczywiście możliwość pisania po ekranie (w obszarze, gdzie nie występują przyciski). W trakcie rysowania kontur jest na bieżąco aktualizowany i wyświetlany. Po zakończeniu narysowania konturu gestu, wykonuje się zawsze jakaś akcja. W przypadku trybu zarządzania gestami będzie to przetworzenie i wykreślenie histogramu łańcuchów Freemana na ekranie, natomiast w trybie szybkiego pisania będzie to próba dokonania analizy wprowadzonych konturów.

Naciśnięcie niektórych przycisków skutkuje wyświetleniem użytkownikowi ekranu wprowadzania danych za pomocą klawiatury ekranowej. Na przykład wciśnięcie przycisku "Zapisz wzorzec" powoduje wyświetlenie ekranu z rysunku ??? a po wpisaniu znaku i zatwierdzeniu pokazany zostanie ekran wyboru liczby gestów elementarnych (z numeryczną klawiaturą ekranową). Dopiero zatwierdzenie tego ekranu z poprawnymi danymi spowoduje dodanie gestu do bazy wzorców.

Rysunek ???. Ekran wyboru znaku dla nowego wzorca Rysunek ???. Ekran wyboru liczby pojedynczych gestów dla nowego wzorca

Wprowadzenie przez użytkownika gestu, który nie został rozpoznany przez algorytm skutkuje tym, że również zostanie wyświetlone podobne okno dodawania nowego gesty, lecz tym razem z pytaniem o to, jaki znak ma reprezentować nierozpoznany gest.

## 5.4. Wykorzystane metody

### 5.4.1. Wykorzystanie biblioteki OpenCV

Do wyznaczania histogramów i obliczania współczynnika korelacji Pearsona dla histogramów wykorzystano funkcje z biblioteki OpenCV służącej do przetwarzania obrazów. Użycie biblioteki wymaga jej inicjalizacji podczas startu aplikacji.

Za wyznaczanie histogramu odpowiedzialna jest funkcja o nazwie "calcHist". Wynik zapisuje w obiektach klasy "Mat" (pochodzących również z biblioteki OpenCV). Natomiast do samego obliczenia współczynnika korelacji używana jest funkcja "compareHist" z pakietu "Imgproc". Jej parametrami są dwa histogramy wejściowe o takich samych rozmiarach oraz numer metody porównywania, jaka ma zostać użyta. Biblioteka OpenCV udostępnia 4 metody porównywania:

* CV\_COMP\_CORREL - korelacja Pearsona,
* CV\_COMP\_CHISQR - metoda Chi-kwadrat,
* CV\_COMP\_INTERSECT - metoda przecięć,
* CV\_COMP\_BHATTACHARYYA - metoda współczynnika Bhattacharyya (lub odległości Hellingera).

Funkcja "compareHist" zwraca w wyniku liczbę zmiennoprzecinkową typu . Jest to współczynnik korelacji, gdyż wybraną metodą porównywania histogramów jest w aplikacji korelacja Pearsona, której to użycie podczas testów dawało najlepsze wyniki w rozpoznawaniu.

### 5.4.2. Mechanizm serializacji

Do zapisu i odczytu bazy wzorców w systemie plików wykorzystano mechanizm serializacji obiektów występujący w języku programowania Java. Serializacja polega na przekształceniu obiektów do postaci strumienia bajtów z zachowaniem aktualnego stanu. Procesem odwrotnym jest deserializacja, polegająca na odczycie strumienia bajtów z pliku i odtworzeniu obiektu wraz z jego stanem. Serializowanym obiektem w programie jest obiekt klasy SamplesContainer. Zawiera on listę wszystkich wzorców jako pole klasy. Mechanizm serializacji wymaga, aby wszystkie klasy, które będą zapisywane (a więc również te, które są agregowane wewnątrz innych obiektów) implementowały interfejs . Dodatkowo zalecane jest, aby serializowane klasy miały zdefiniowane statyczne pole z numerem wersji klasy. W przypadku zmiany struktury klasy, należy zwiększyć numer jej wersji. Pojawienie się niezgodności wersji przy odczycie obiektów uniemożliwi proces deserializacji, wymuszając stosowanie najnowszej wersji.

## 5.5. Moduł wprowadzania znaków pisma odręcznego

Głównym celem implementacji oprogramowania na urządzenie mobilne było utworzenie modułu wprowadzania znaków pisma odręcznego jako alternatywnej metody wpisywania tekstu. System Android pozwala na tworzenie aplikacji działających w tle jako usługi. Zastosowanie takiego podejścia wraz z przyznaniem aplikacji pełnego dostępu do ekranu dotykowego mogłoby rozszerzyć funkcjonalność modułu i pozwolić na bezpośrednie wprowadzanie tekstu za pomocą gestów w innych aplikacjach uruchomionych w systemie. W utworzonym module użytkownik ma do dyspozycji możliwość kopiowania wpisanego tekstu do schowka, co również pozwala na wykorzystanie go w innych aplikacjach.

Projekt skupia się na rozpoznawaniu znaków tekstowych, natomiast łatwo można rozszerzyć tę funkcjonalność o obsługę gestów takich jak: gest usuwania ostatniego znaku (klawisz backspace) bądź też zmiany aktualnego położenia kursora tekstu. W aplikacji możliwe jest natomiast wstawianie białych znaków (spacji lub załamania wiersza) poprzez gesty. Nie istnieje ograniczenie dotyczące długości tekstu przypisanego do gestu. Oznacza to, że jednemu gestowi może zostać przypisanych wiele znaków (np. całe wyrazy), co jeszcze bardziej może zwiększyć wygodę i szybkość pisania poprzez stosowanie inteligentnego interfejsu dotykowego przy wprowadzaniu tekstu.

## 5.6. Cechy systemu

System pozwala na ciągłe wpisywanie tekstu przez użytkownika. Nie jest konieczne odczekanie pewnego czasu po wpisaniu gestu, gdyż analiza odbywa się na bieżąco. Mimo to, można zmienić nieco działanie interfejsu dotykowego poprzez odczekanie po wpisaniu gestu. System zinterpretuje to jako zakończenie pisania znaku i nie będzie rozważał gestów z większą złożonością niż liczba wprowadzonych konturów wejściowych. Takie podejście sprawia, że użytkownik zawsze będzie miał wrażenie, że tekst jest rozpoznawany na bieżąco, bez zbędnych opóźnień.

System jest praktycznie gotowy do użytkowania nawet w przypadku pustej bazy wzorców. Braki wzorców zostaną uzupełnione w wyniku zadawania użytkownikowi pytań o to, jakie znaki zostały wprowadzone. Z czasem takich pytań będzie coraz mniej, skuteczność w rozpoznawaniu powinna wzrastać, a system powinien stawać się coraz bardziej "samodzielny".

## 5.7. Testy aplikacji

### 5.7.1. Dane statystyczne

W celu wyznaczenia skuteczności rozpoznawania gestów (procentowej części poprawnych wyników spośród wszystkich) zaimplementowano w aplikacji osobny moduł do zbierania danych statystycznych. Odpowiednie testy zostały przeprowadzone na urządzeniu mobilnym podłączonym do komputera przez kabel USB w trybie debugowania, co pozwala na przesłanie zbiorczego wyniku w odpowiedniej formie do środowiska programistycznego.

Moduł statystyk zbiera dane w postaci listy zdarzeń. Zdarzeniami są np. poprawne rozpoznanie gestu lub błędna identyfikacja. Lista zdarzeń ma ograniczony rozmiar, w przypadku przepełnienia przechowywane są zawsze najnowsze zdarzenia. Dzięki takiemu podejściu można łatwo odczytać, jaką część spośród wszystkich ostatnich zdarzeń stanowią np. poprawne rozpoznania gestu, a więc jaka jest skuteczność metody wprowadzania znaków.

### 5.7.2. Test uczenia się systemu

Aby zademonstrować, jak działa inteligencja systemu, i czy rzeczywiście prowadzi do poprawienia skuteczności, przeprowadzono pewien eksperyment. Polegał na tym, że usunięto w aplikacji wszystkie wzorce, pozostawiając pustą bazę, a następnie zaczęto wprowadzać kontury znaków w celu rozpoznania ich. Oczywiście początkowo system nie znał znaków i wyświetlał ekran z pytaniem o to, jaki gest został wprowadzony. Jednak w stosunkowo krótkim czasie, system zaczął wykorzystywać zdobytą wiedzę o gestach i poprawił skuteczność. W trakcie trwania testu zapisywane oraz przesyłane były dane statystyczne, które posłużyły do stworzenia poniższego wykresu. Przedstawia on zależność skuteczności rozpoznawania gestów od liczby wszystkich wzorców znajdujących się w bazie aplikacji.



Rysunek ???. Wykres zależności skuteczności rozpoznawania gestów od liczby wszystkich wzorców

Test aplikacji przeprowadzono na wszystkich małych literach alfabetu polskiego. Można zauważyć, że już przy liczbie wzorców równej 45, skuteczność rozpoznawania zaczęła oscylować wokół wartości .

### 5.7.3. Dobór parametrów

W implementacji systemu wiele wartości współczynników zostało dobranych doświadczalnie na skutek przeprowadzenia testów. W początkowej fazie powstawania programu wartości były przyjmowane intuicyjnie, gdyż nie było możliwości przeprowadzenia testów skuteczności, natomiast kiedy system rozpoznawania osiągnął sprawność i pełną funkcjonalność, można było przeprowadzić testy i stwierdzić, jakie wartości współczynników poprawiają parametry interfejsu dotykowego.

Poniższa tabela przedstawia zbiorcze zestawienie najważniejszych wartości parametrów programu istotnych dla algorytmu rozpoznawania gestów.

Tabela ???. Zbiorcze zestawienie wartości parametrów wykorzystanych w aplikacji

|  |  |
| --- | --- |
| **Parametr** | **Wartość** |
| czas braku aktywności użytkownika, po jakim wprowadzanie gestu zostaje uznane za zakończone |  |
| liczba pikseli do uśredniania przy filtrowaniu |  |
| minimalna liczba pikseli konturu (by nie został uznany za kropkę) |  |
| liczba kierunków łańcuchów Freemana |  |
| waga współczynnika korelacji histogramów (przy obliczaniu sumarycznego współczynnika korelacji) |  |
| waga współczynnika korelacji punktów startowych konturów |  |
| waga współczynnika korelacji długości konturów |  |
| minimalny współczynnik korelacji dla pojedynczego gestu |  |
| maksymalna liczba wzorców przechowywana dla jednego znaku |  |
| graniczna wartość współczynnika korelacji dla automatycznego dodawania wzorców (nie występuje dla większych wartości) |  |
| graniczna wartość bilansu identyfikacji, dla której wzorzec zostaje usunięty |  |

### 5.7.4. Testy końcowe działającej aplikacji

Na działającej aplikacji posiadającej stosunkowo dużą liczbę wzorców dodanych przez użytkownika aplikacji również przeprowadzono testy skuteczności. Testy te dotyczyły rozpoznawania małych liter polskich, dużych liter, cyfr oraz niektórych znaków specjalnych. Liczba wszystkich wzorców znajdujących się w bazie aplikacji przed wykonaniem testów wyniosła . Warto dodać, że przy takiej stosunkowo dużej liczbie wzorców, rozmiar pliku przechowującego bazę wzorców wynosił jedynie . W wyniku testu uzyskano średnią skuteczność rozpoznawania gestów równą około . Zmierzony został także czas analizy po wpisaniu konturu. Jest to czas porównywania wpisanego gestu ze wszystkimi wzorcami. Na badanym urządzeniu czas ten mieścił się w przedziale od do , zaś jego średnia wartość wyniosła . Jest to bardzo dobrym wynikiem i można stwierdzić, że opóźnienia wynikające z etapu analizy praktycznie nie występują i są zupełnie niezauważalne z perspektywy użytkownika.

statystyki skuteczności dla każdej litery (znaku), liczba wzorców dla znaku

testy i wyniki różnych metod korelacji w OpenCV (tabelka),

liczba gestów dodanych automatycznie, liczba pytań z powodu nierozpoznania żadnego gestu, liczba usuniętych gestów z powodu złego bilansu,

# 6. Literatura

1. Tadeusiewicz R.: *Rozpoznawanie obrazów*, PWN, Warszawa, 1991.

2. Bradski G., Kaehler A.: *Learning OpenCV*, 2008.

3. Baggio D.: *Mastering OpenCV with Practical Computer Vision Projects*, 2012.

Załączniki

Plakat

na plakacie (posterze) dyplomu powinny znaleźć się m.in. następujące informacje

1. kierunek studiów

2. specjalność

3. rok akademicki ukończenia studiów

zaciągnąć z opisu pracy i streszczenia