POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Elektronika (EKA)
SPECJALNOŚĆ: Aparatura elektroniczna (EAE)

# PRACA DYPLOMOWA MAGISTERSKA

System weryfikacji mówcy w czasie rzeczywistym

Speaker verification in real-time system

AUTOR:
inż. Adam Matusiak

PROWADZĄCY PRACĘ:
dr hab. inż. Józef Borkowski Prof. PWr

OCENA PRACY:

WROCŁAW 2017

# Rozdział 1

# Przetwarzanie sygnałów w rozpoznawaniu mowy

## 1.1 Wprowadzenie

Duża dynamika wzrostu produkcji danych takiego typu jak nagrania audio czy nagrania video stwarza potrzebę wdrażania nowych i niezawodnych systemów biometrycznych pozwalających na stwierdzenie przy ich pomocy tożsamości osób. Dotychczasowe techniki rozpoznawania (identyfikacji) korzystające z tego typu danych, ze względu na wysoką złożoność obliczeniową realizujących je algorytmów sprawiają, że zadanie rozpoznawania tożsamości staje się operacją długotrwałą. Powodowane jest to tym, iż wspomniane rodzaje medium charakteryzują się wysoką pojemnością informacji, a przy tym sama informacja dotycząca tożsamości osób stanowi jedynie ułamek pojemność takiego pasma. Wspomniana trudność nie stanowi problemu i jest dopuszczalna w aplikacjach typu *offline*, tam gdzie nie są oczekiwane natychmiastowe wyniki oraz gdzie rozmiar danych jest stosunkowo mały. Sprawa jednak komplikuje się gdy w grę wchodzi przetwarzanie całych petabajtów danych lub w aplikacjach czasu rzeczywistego, gdzie ograniczenia czasowe na otrzymanie wyniku są równie istotne jak ich poprawność.

Zdaniem autora pracy główne motywacje dla zastosowania tego typu systemów są dwie. Pierwsza z nich jest związana z administracją państwową jako środek inwigilacji ze strony agencji rządowych (np. jako system przeciwdziałania terroryzmowi lub aparat policyjny) lub dużych korporacji (np. jako system wspomagający sprzedaż produktów). Z punktu widzenia tego typu podmiotów, zainteresowanie systemami biometrycznymi wykorzystującymi tego typu dane wynika z powszechności tej informacji - dla przykładu setki milionów godzin materiału filmowego wraz z dźwiękiem umieszczane rok rocznie na serwisie internetowym youtube [19] mogą stanowić dobry materiał startowy dla tego typu systemu. Kolejnym argumentem przemawiającym na korzyść systemów biometrycznych wykorzystujących obraz oraz dźwięk w tym przypadku są względnie małe koszta infrastruktury monitorującej - sieci mikrofonów oraz kamer, ze względu na ich dostępność oraz małą cenę. Zwłaszcza z tego względu, że może zostać wykorzystana infrastruktura już istniejąca - sieci komórkowe czy internetowe kamery. Bardzo szybki rozwój infrastruktury na wszystkich kontynentach oraz coraz to szybciej zwiększająca się populacja ludzi potęguje ten efekt. Inne metody ustalania tożsamości takie jak systemy biometryczne wykorzystujące odciski palców, testy dna czy skany tęczówki oka wymagają zazwyczaj kooperacji ze strony osoby identyfikowanej - co w niektórych sytuacjach może stanowić problem, szczególnie gdy system cechuje się masowością i ma w zamierzeniu pozyskać informację na temat tożsamości znaczącej części populacji danego obszaru geograficznego.

Z oczywistego względu, iż człowiek nie jest w stanie podołać analizie takiej ilości danych pochodzących z rozbudowanej infrastruktury, potrzebne są systemy automatycznego rozpoznawania mówcy czy też systemy automatycznego rozpoznawania twarzy. Inną możliwością dla realizacji rozbudowanych systemów rozpoznawania jest wykonywanie analizy sygnału mowy już na etapie urządzeń które rejestrują sygnały mowy lub obrazy. Szeroko rozpowszechnione urządzenia z systemami wbudowanymi np. telefony komórkowe, telewizory czy konsole, ze względu na bardzo dynamiczny rozwój możliwości obliczeniowych są w stanie nie tylko rejestrować wspomniane dane, ale także analizować sygnał i przesyłać do systemu scentralizowanego konkretny, matematyczny model mówcy/twarzy. Ogranicza to w sposób znaczący konieczną ilość przesyłanych danych.

Drugą motywacją, z pewnością budzącą mniejsze kontrowersje, jest zastosowanie automatycznych systemów identyfikacji tożsamości w systemach chroniących poufne informacje lub ograniczające dostęp do posiadanego mienia np. jako lokalne punkty dostępu w budynkach lub magazynach. Tego typu systemy muszą cechować się bardzo dużą niezawodnością procesu weryfikacji oraz szybkością jej przeprowadzenia - tak aby czas potrzebny na wykonanie obliczeń niezbędnych do dokonania weryfikacji nie był obciążeniem dla użytkownika. Chociaż systemy analizujące sygnał mowy (np. poprzez połączenie telefoniczne podczas autoryzacji dla sektora bankowego) mogą być implementowane i wykonywane na dużych systemach informatycznych z ogromnymi możliwościami obliczeniowymi, ze względu na opóźnienia w komunikacji lub brak zewnętrznej sieci informatycznej istnieje potrzeba implementacji takiego systemu lokalnie, na urządzeniu wbudowanym i w czasie rzeczywistym. Przykładem takiego systemu jest system autoryzacji dostępu w biurowcach, gdzie wewnętrzna sieć urządzeń rejestrujących i analizujących sygnał mowy w czasie rzeczywistym dokonuje procesu weryfikacji mówcy i podejmuje decyzję o udzieleniu dostępu do zastrzeżonych obszarów.

Z zaprezentowanych powyżej rozważań widać, iż kluczowym zagadnieniem dotyczącym obu typów rozpatrywanych systemów rozpoznawania jest szybkość ich działania. Dla pierwszego przypadku ilość przetwarzanych danych jest ogromna i dlatego ważne jest by stosowane algorytmy były jak najmniej kosztowne obliczeniowo oraz by sprzęt na którym są wykonywane umożliwiał ich jak najszybszą realizację po to, aby była możliwa analiza całości uzyskanego materiału w rozsądnym czasie. W drugim przypadku zaś ta sama cecha umożliwia przeprowadzenie rozwiązania problemu rozpoznawania w czasie rzeczywistym bez zbędnych opóźnień. Rozwój w dziedzinie techniki cyfrowej sprawia, że problem szybkości wykonywania tego zadania jest coraz mniejszy. Dla pierwszej dziedziny możliwe jest to dzięki temu, że duże serwery zaczynają być masowo wyposażane w jednostki graficzne (GPU) czy też układy programowalne (FPGA) co jest połączone ze wzrostem szybkości samych jednostek centralnych (CPU) oraz procesem zrównoleglenia tychże jednostek. Wykorzystanie nowych architektur sprzętowych spowodowane jest szybkich rozwojem w dziedzinie algorytmów heurystycznych, takich jak sieci neuronowe oraz ich wykorzystanie dla metod sztucznej inteligencji. Podobnie jest w dziedzinie systemów wbudowanych. Systemy mikroprocesorowe osiągają wydajności obliczeniowe umożliwiające przetwarzanie w czasie rzeczywistym złożonych algorytmów. I tutaj także możliwe jest stosowanie wysoko wydajnych układów programowalnych FPGA, które umożliwiają wykonywanie zadań realizowanych poprzez sieci neuronowe. Platformy sprzętowe wspomagające wykonywanie algorytmów cyfrowego przetwarzania sygnałów na czele z procesorami sygnałowymi DSP również przyczyniają się do efektywnej implementacji systemów rozpoznawania, szczególnie dla dziedziny rozpoznawania mówcy i rozpoznawania mowy.

Równie istotnym czynnikiem ograniczającym powszechne istnienie tego typu systemów jest ich skuteczność identyfikacji. Dotychczasowe rozwiązania dla systemów weryfikacji

mówcy osiągały błąd sięgający 4% (ERR - *ang. equal error ratio*), (2% w przypadku fuzji systemów) [15]. Szybki rozwój w dziedzinie rozpoznawania wzorca, spowodowany m. in. dynamicznym rozwojem w dziedzinie metod sztucznej inteligencji pozwala redukować błędy - dla problemu identyfikacji mówcy - nawet do 0.4 % (ERR) [16]. Szeroko stosowana miara oceny systemów weryfikacji mówcy - ERR mówi o prawdopodobieństwie błędu systemu dla progu decyzji, dla którego błąd odrzucenia prawdziwego mówcy jest równy błędowi akceptacji mówcy podszywającego się.

W niniejszej pracy podejmowana jest próba przedstawienia architektury oprogramowania pozwalającej na efektywną obliczeniowo aplikację algorytmów realizujących zadanie weryfikacji mówcy na platformach sprzętowych systemów wbudowanych. Projekt zakłada, że przetwarzanie wejściowego sygnału mowy przeprowadzane jest w czasie rzeczywistym i umożliwia otrzymanie decyzji o autoryzacji w czasie nie dłuższym niż oczekiwany przez potencjalnych użytkowników takiego systemu. Platformy sprzętowe, za pomocą których realizowane jest przetwarzanie, przewidziane są jako systemy mikroprocesorowe - ogólnego przeznaczenia (CPU), procesory sygnałowe (DSP) czy mikrokontrolery (MCU) posiadające wsparcie dla języka programowania C++ oraz dla jego najnowszych standardów: C++11, C++14 i C++17. Propozycja architektury nie bazuje na żadnym systemie operacyjnym i może być wykorzystana również w systemach wbudowanych które nie oferują żadnego środowiska uruchomieniowego. Jednak obecność takiego systemu, zwłaszcza systemu operacyjnego czasu rzeczywistego (RTOS), w dużym stopniu może ułatwić implementację konkretnego systemu na urządzeniu.

Projektowana architektura ma w zamierzeniu ułatwiać aplikację różnych technik realizujących weryfikację mówcy proponując narzędzia reprezentujące abstrakcję kolejnych etapów dla klasycznego systemu weryfikacji mówcy.

Chociaż proponowane oprogramowanie przeznaczone jest jedynie dla systemów mikroprocesorowych to może być również wykorzystane jako element systemu przetwarzający wstępnie dane - np. tworzący wektory akustyczne przekazywane dalej do innych systemów np. sieci neuronowej zaimplementowanej na układzie FPGA lub zdalnie w chmurze. Niewykluczone jest też użycie abstrakcji dopasowywania cech do implementacji sieci neuronowej na mikroprocesorze - co może jednak być nieefektywne.

Praca przedstawia jedną przykładową implementację systemu weryfikacji mówcy opartą na zaprojektowanej architekturze. W proponowanym zastosowaniu autor korzysta ze wsparcia systemu operacyjnego linux w dystrybucji debianowej. Platformą sprzętową jest komputer jednopłytkowy Raspberry Pi 3. Powstałe urządzenie w zamierzeniu ma stanowić lokalny punkt dostępu w budynku chroniący zastrzeżony obszar przed nieautoryzowanym dostępem. System funkcjonalnie realizuje weryfikację mówcy z wyświetlanym hasłem, korzystającym z technik MFCC (rozdział 1.6.2) i VQ (rozdział 1.7.2) - zaproponowany w publikacji [8] (cechujący się wartością ERR na poziomie 2%), w odróżnieniu jednak charakteryzującym się przetwarzaniem w czasie rzeczywistym. Wybrano to rozwiązanie ze względu na ambicję zmodyfikowania go w przyszłości na system zaproponowany w [5] - czyli system weryfikacji mówcy w czasie rzeczywistym oparty na metodach MFCC i VQ dodatkowo wykorzystujący tzw. *speaker prunning* (rozdział 1.8.3) w celu wykorzystanie nowoczesnego i wydajnego systemu weryfikacji mówcy w czasie rzeczywistym (cechującym się wartością ERR na poziomie 0.5% oraz bardzo szybką odpowiedzią systemu).

## 1.2 Weryfikacja mówcy

Proces weryfikacji mówcy (*ang. speaker verification*) jest związany z szerszym zagadnieniem - rozpoznawania mówcy (*ang. speaker recognition*), które charakteryzuje ogół metod wykorzystujących dane biometryczne zawarte w sygnale mowy w celu określenia tożsamości.

Sygnał mowy może być rozpatrywany jako cecha biometryczna. Sygnał mowy charakteryzowany jest przez budowę aparatu głosowego człowieka, która jest mniej lub bardziej unikatowa dla każdego człowieka, umożliwiając rozróżnienie badanej jednostki na tle populacji.

Ogólną strukturę problemu rozpoznawania mówcy można rozłożyć na na trzy elementy [1]. Po pierwsze, konieczne jest aby tworzony system dysponował modelem charakterystyk aparatu głosowego człowieka. Model taki może przybrać formę modelu fizykomatematycznego aparatu głosowego człowieka. Otrzymany model musi umożliwiać parametryzację - skojarzenie z konkretną osobą. Model taki tworzony jest poprzez analizę sygnału mowy. Dopiero na tej podstawie możliwe jest porównywanie modelu utworzonego przy użyciu testowanego sygnału z modelem odniesienia. Forma i cel tego porównania definiują podklasę zadania rozpoznawania mówcy.

Weryfikacja mówcy charakteryzuje się wykonaniem dwóch kluczowych porównań - pierwszego pomiędzy modelem utworzonym z poddanego weryfikacji sygnału mowy a pamiętanym modelem osoby, której dotyczy weryfikacja. W odróżnieniu od zadania identyfikacji, podczas weryfikacji mówcy potrzebna jest więc znajomość tożsamości osoby poddanej weryfikacji. Drugie z kolei porównanie dokonywane jest pomiędzy modelem poddanym weryfikacji, a uogólnionym modelem całej populacji (*ang. background model*) lub pewnej jej podgrupy (*ang. cohort model*). Na podstawie relacji tych dwóch odległości podejmowana jest decyzja o autoryzacji.

W przypadku kiedy nie jest możliwa lub pożądana znajomość przez system tożsamości osoby weryfikowanej, przed dokonaniem autoryzacji możliwe jest zastosowanie bardziej złożonego zadania identyfikacji mówcy na otwartym zbiorze (*ang. open-set speaker identification*). Proces ten można uważać jako złożenie zadania weryfikacji mówcy oraz identyfikacji mówcy na zbiorze zamkniętym (*ang. close-set speaker identification*). Polega on na przeprowadzeniu weryfikacji mówcy na modelu uzyskanym z procesu identyfikacji mówcy na zbiorze zamkniętym, która dokonuje porównania z całą dostępną bazą modeli mówców i zwraca ten najbliższy modelowi testowanemu. Zadanie te jest więc obliczeniowo co najmniej tak złożony, jak weryfikacja mówcy (dla bazy, w której znajduje się tylko jeden mówca).

Implementacja systemu weryfikacji mówcy jest nazywana automatycznym systemem rozpoznawania mówcy (*ang. automatic speaker verification system*).

### 1.2.1 Zastosowania

Głównymi obszarami zastosowań systemów weryfikacji mówcy są:

- Usługi bankowe: jako zdalna weryfikacja (np. telefoniczna, audio-video) dająca dostęp do danych dotyczących konta czy potwierdzenia realizacji usług. Stosowana może być także bez wiedzy zainteresowanego do zapobiegania oszustwom np. poprzez sprawdzenie czy osoba nie znajduje się w bazie osób podejrzanych.

- Zastosowania prawne: podobnie jak odciski palców czy badania DNA do weryfikacji tożsamości osób na nagraniach.

- Inwigilacja: do zapobiegania przestępstwom czy też terroryzmowi.
- Ochrona dostępu: jako lokalne punkty dostępu chroniące zasoby fizyczne (biura, magazyny, serwerownie) lub dostęp do wszelkiego rodzaju informacji (internetowe bazy danych).
- *Indeksowanie* wypowiedzi: w towarzystwie technik rozpoznawania mowy, system weryfikacji mówcy pomaga archiwizować zebrane, masowe nagrania audio i stwierdzać przynależność danych wypowiedzi do konkretnego mówcy.

### 1.2.2 Inne systemy biometryczne.

System weryfikacji mówcy może być skojarzony z innymi systemami rozpoznawania biometryk, szczególnie w lokalnych punktach autoryzacji, gdzie mówca znajduje się fizycznie. Może okazać się to konieczne chociażby ze względu na to, że część populacji jest niema. Przykładami podanymi w [1] są:

- Analiza DNA - niechybnie najpewniejsza metoda identyfikacji, jednak trwająca dużo dłużej niż weryfikacja mówcy.
- Analiza kształtu małżowiny usznej - może być użyta w połączeniu z weryfikacją mówcy w połączeniu telefonicznym na odległość np. poprzez użycie czujnika (obraz uzyskany za pomocą kamery lub analiza akustyczna) w telefonie komórkowym. Okazuje się, że kształt małżowiny usznej różni się na tyle w obrębie populacji, iż można zastosować tą technikę jako wsparcie dla systemów weryfikacji mówcy.
- System rozpoznawania twarzy - ze względu na powszechność nagrań typu audio-wideo, system weryfikacji mówcy świetnie nadaje się do współpracy z systemami biometrycznymi wykorzystującymi rozpoznawanie mówcy. Kombinacja ta pozwala stwierdzić tożsamość osób znajdujących się na takich nagraniach.
- Skaner odcisku palca - ze względu na tanie, dedykowane czujniki, rozsądnym jest wyposażenie lokalnego punktu dostępu w system biometryczny oparty na skanie odcisku palca.
- Do innych cech biometrycznych, potencjalnie nadających się do współpracy z systemem rozpoznawania mówcy należą: wygląd i geometria dłoni, obraz tęczówki oka, obraz siatkówki oka, obraz termograficzny ciała, rozkład żył w dłoni, sposób chodu, pismo czy sposób pisania na klawiaturze. Do tego wyróżnia się systemy wielomodalne zawierające szereg podanych metod w jednym systemie.

## 1.3 Relacja pomiędzy rozpoznawaniem mowy, a rozpoznawaniem mówcy

Kluczowe jest odróżnienie procesu rozpoznawania mówcy od systemów rozpoznawania mowy (*ang. speech recognition*). Pomiędzy tymi dwoma rozpatrywanymi dziedzinami z zakresu analizy sygnału mowy występuje dychotomiczny podział. Wynika to z tego, że sygnał mowy jest sygnałem bogatym informacyjnie oraz że jedynie mała część tej informacji posiada znaczenie semantyczne, zaś reszta niesie wiedzę o budowie konkretnego, ludzkiego narządu mowy. W problemie rozpoznawania mowy nie jest istotna tożsamość

osoby wypowiadającej się, a jedynie sens jej wypowiedzi. Zatem reszta sygnału nie zawierająca odczytywanej wiadomości jest redundantna z punktu widzenia tego zagadnienia - cała informacja biometryczna jest niewykorzystywana, co za tym idzie często filtrowana przez zaimplementowany system. Z drugiej strony, w systemach rozpoznawania mówcy, w samym sednie jego zainteresowania, abstrahuje się od treści mowy. Stanowi ona jedynie środek dla dostarczenia informacji o fizjologii aparatu mowy. Dlatego prawdopodobnie system rozpoznawania mówcy usunie treść mowy, a utworzy jedynie model aparatu głosowego. Usprawiedliwia to twierdzenie o rozłączności tych dziedzin ze względu na zainteresowanie informacją zawartą w sygnale mowy.

Okazuje się, że wspomniana wyżej zależność powoduje to, że techniki przetwarzania sygnału stosowane przy analizie obu dziedzin są w zasadzie bardzo podobne i zaczynają się różnić na etapie modelowania mówcy/mowy.

W przypadku rozpoznawania mówcy zależnego od wypowiadanego tekstu czy rozpoznawania mówcy z generowanym tekstem informacja semantyczna wykorzystywana jest jedynie do określenia zakresu badanych głosek czy zapobieganiu problemowi żywotności. Informacja ta zatem nie wpływa na postać stosowanych technik rozpoznawania mówcy, a jedynie na optymalny ich dobór - ujawnia kontekst użycia. Innym przykładem tego typu jest zastosowania technik rozpoznawania treści języka naturalnego m. in. w odmianach omawianych systemów opartych na nagromadzonej wiedzy (*ang. knowledge-based systems*), których zadaniem jest jedynie wzmocnienie procesu weryfikacji oraz zapobieganie wystąpienia problemu żywotności (*ang. liveness issue*) (rozdział 1.4.1).

## 1.4 Klasyfikacja problemu weryfikacji mówcy

### 1.4.1 Weryfikacja mówcy zależna od wypowiadanego tekstu (*ang. text-dependent speaker verification*)

System weryfikacji mówcy który tworzy modele mówców na podstawie jednej, ustalonej frazy zawartej w dostarczonym do niego sygnale mowy jest nazywany systemem weryfikacji mówcy zależnym od wypowiadanego tekstu. System ten oczekuje, że w fazie testowania dostarczona zostanie jako próba testowa wypowiedź o takiej samej treści. Przykładem może być system autoryzacji, oczekujący zawsze na to samo hasło. Problem tego typu jest zdecydowanie łatwiejszy do realizacji i można od niego oczekiwać dużo lepszych wyników w porównaniu do innych rodzajów weryfikacji mówcy. Inną zaletą takiego systemu jest bardzo krótki etap uczenia - wymaga się od weryfikowanego użytkownika dostarczenia jednej lub paru próbek stałego, wypowiadanego hasła. Oczywistym niebezpieczeństwem dla takiego systemu jest tzw. problemem żywotności *(ang. liveness problem)*. Problem żywotności polega na możliwości oszukania działającego systemu weryfikacji mówcy poprzez dostarczenie na wejście spreparowanego sygnału mowy - na przykład wysokiej jakości nagranie weryfikowanego mówcy, edytowane w odpowiedni sposób. Wraz z rozwojem technik audio zmylenie systemu niezabezpieczonego ze względu na ten typ ataku staje się coraz łatwiejsze. Istnieje szereg metod pozwalających na detekcję tego, czy dostarczony fragment mowy pochodzi od prawdziwego mówcy - jednak takie systemy okazują się niewystarczające w przypadkach ochrony cennych danych lub krytycznych zasobów. Dobrą alternatywą dla takiego systemu może być system z wyświetlanym hasłem (1.4.3). Standardem w implementacji tego typu systemów jest zastosowanie modelowania tzw. ukrytymi modelami Markowa (HMM - *ang. Hidden Markov Model*) ze względu na możliwość modelowania sekwencji zdarzeń - z czym mamy do czynienia w rozpatrywanym problemie. Z tego też

powodu weryfikacja mówcy zależna od wypowiadanego tekstu jest najbardziej zbliżonym problemem do rozpoznawania mowy.

### 1.4.2 Weryfikacja mówcy niezależna od wypowiadanego tekstu (*ang. text-independent speaker verification*)

System weryfikacji mówcy dokonywujący weryfikacji mówcy bez względu na zawartość lingwistyczną wypowiedzi nazywa się systemem weryfikacji mówcy niezależnym od wypowiadanego tekstu. Taki system nie przyjmuje żadnych założeń co do treści wypowiedzi, dlatego też w fazie trenowania potrzebuje więcej materiału (różnorodnych wypowiedzi) od modelowanego mówcy. Problem ten jest zdecydowanie trudniejszy do realizacji w porównaniu do wcześniej omawianego. Nie jest możliwe zastosowanie pewnych upraszczających założeń jeżeli chodzi o model fizyczny aparatu głosowego człowieka - nie jest znane pobudzenie modelu - tak jak to czyni się w problemie ze znanym tekstem. Temu zagadnieniu również towarzyszy problem żywotności - jednak nie przyjmuje takiej skali jak we wcześniej omawianym, łatwiej jest skonstruować odpowiednie systemy detekcji spreparowanej wypowiedzi. Ten typ weryfikacji cechuje także uniwersalność użycia - nie jest wymagana współpraca weryfikowanego mówcy, zatem ten typ weryfikacji używany jest w systemach identyfikacji mówcy na otwartym zbiorze (rozdział 1.2). Niejako standardem dla tego typu systemów stała się kombinacja technik ekstrakcji cech - Melowych współczynników cepstralnych (MFCC) i mikstur Gaussowskich (GMM) będących niesekwencyjnym odpowiednikiem techniki HMM. Innym popularnym przykładem techniki modelowania jest kwantyzacja wektorów (VQ) (rozdział 1.7).

### 1.4.3 Weryfikacja mówcy z wyświetlanym hasłem (*ang. text-prompted speaker verification*)

System weryfikacji mówcy, w którym w celu dokonania weryfikacji wyświetlany jest dla użytkownika tekst, który musi wypowiedzieć, nazywany jest systemem weryfikacji mówcy z wyświetlanym hasłem. Jest to problem podobny do problemu weryfikacji zależnym od tekstu z rozszerzonym modelem mówcy o kolejne wypowiadane frazy. W tym przypadku czas trwania sesji treningowej znajduje się pomiędzy długościami obu poprzednich. Tego typu system oferuje ochronę przed problemem żywotności podobną do systemu weryfikacji niezależnej od tekstu.

## 1.5 Sygnał mowy

### 1.5.1 Generacja sygnału mowy.

Sygnał mowy jest sygnałem akustycznym, którego medium stanowi powietrze a informacja w nim zawarta jest rejestrowana jako chwilowe zmiany ciśnienia akustycznego. Źródłem różnicy ciśnień są płuca człowieka, a dalej za pomocą fałd głosowych jest modulowane - powstaje tzw. ton krtaniowy (*ang. glottal pulse*), który dalej filtrowany jest przez charakterystyki budowy aparatu głosowego człowieka. Ton krtaniowy i wspomniane charakterystyki różnią się ze względu na każdego mówcę i ta cecha jest podstawą wykorzystania w systemie biometrycznym mówcy. Tak zaproponowany model produkcji mowy może być przedstawiony jako szeregowe połączenie tych elementów [13]:

$$S_x(f,t) = S_r(f,t) \cdot |H(f,t)|^2 \tag{1.1}$$

gdzie $S_x(f,t)$ reprezentuje transmitancję sygnału mowy, $S_r(f,t)$ transmitancję tonu krtaniowego oraz $H(f,t)$ funkcję transmitancji dróg głosowych. Charakterystyki dróg głosowych są zmienne w czasie i zależą od stanu 4-5 komór rezonansowych znajdujących się w aparacie głosowym człowieka.

Rozpatrywany sygnał głosowy cechuje się pasmem sygnału o szerokości nawet 8 kHz. Z tego powodu spróbkowany sygnał mowy cechuje się stosunkowo wysoką zawartością informacji. Z punktu widzenia rozpoznawania mówcy sygnał mowy zawiera w wysokim stopniu informację redundantną. Z tego powodu stosuje się parametryzację mowy w procesie ekstrakcji cech (rozdział 1.6).

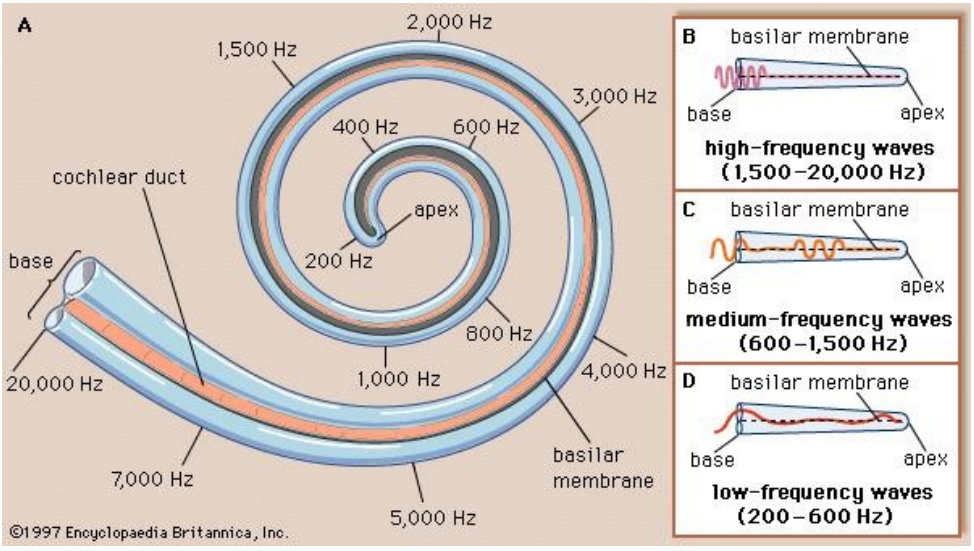
Człowiek dysponuje doskonałymi narzędziami do przeprowadzenia procesów rozpoznawania mowy oraz rozpoznawania mówcy. Analiza oraz zrozumienie mechanizmów powstawania mowy u człowieka dostarcza podstaw do tworzenia modelu generacji mowy i umożliwia budowę takich systemów, jak systemy syntezy mowy. Podobna analiza systemu percepcji mowy, na który składa się aparat słuchowy oraz układ nerwowy związany z dekodowaniem sygnału mowy, daje podstawy do identyfikacji cech, którymi posługuje się ludzki organizm do efektywnego przeprowadzenia zadania rozpoznania mówcy.

Układ produkcji mowy oraz jej percepcji są ze sobą nierozerwalnie związane. Sposób ekstrakcji informacji przez narząd słuchu odzwierciedla fizjologię produkcji mowy - zatem może wskazać najważniejsze cechy sygnału mowy dla naturalnego procesu rozpoznania mówcy. Dlatego wydaje się właściwe prześledzenie związków pomiędzy tymi dwoma elementami.

Ludzki układ percepcji dokonuje rozróżnienia sygnałów audio poprzez rozróżnienie trzech właściwości: wysokości dźwięku, głośności oraz barwy dźwięku - tembru.

### 1.5.2 Aparat słuchowy.

Narząd słuchu człowieka można rozpatrywać jako transduktor, co znaczy, że mapuje zmiany ciśnienia akustycznego w powietrzu na sygnał elektryczny w układzie nerwowym. Na samym początku toru przetwarzania sygnału audio znajduje się małżowina uszna, której zadaniem jest skupienie dźwięku. Sygnał akustyczny wpadający do kanału słuchowego jest filtrowany ze względu na jego fizyczne rozmiary, usuwane są niskie częstotliwości. Zmiany ciśnienia akustycznego zamieniane są na fale mechaniczne w ciele stałym na błonie bębenkowej, a następnie wzmacniane przez układ kosteczek słuchowych - młoteczka, kowadełka i strzemiączka. Strzemiączko łączy się z uchem wewnętrznym poprzez błonę okienka owalnego (*łac. fenestra vestibuli*), które jest wejściem do ślimaka, który z kolej jest częścią narządu Cortiego. Ruch membrany okienka owalnego powoduje ruch płynu nazywanego perylimfą w ślimaku. W taki sposób powstaje fala rozchodząca się w obszarze ślimaka, propaguje się do szczytu tego narządu i następnie kanałem połączonym wraca w kierunku okienka okrągłego. Błona podstawna oddziela oba kanały od siebie i w zależności od długości fali w cieczy jest wyginana w innym obszarze - jest aparatem analizy częstotliwościowej sygnału. Pobudzane są rejony dla konkretnych częstotliwości (rysunek 1.1). Znajdujące się na błonie podstawnej rzęski zagłębione, ustawione w rzędzie wzdłuż ślimaka, zamieniają wychylenie błony na impulsy elektryczne wysyłane dalej przez nerw przedsionkowo-ślimakowy do mózgu. Ze względu na różną sztywność błony podstawnej, narząd Cortiego w różny sposób interpretuje intensywność danej częstotliwości. Inne trzy równoległe rzędy rzęsek dostarczają sprzężenia zwrotnego ze strony mózgu pozwalając na zwiększenie rozdzielczości przeprowadzonej analizy częstotliwościowej. Rzęski zewnętrzne pobudzane są głównie w rejonie największego ugięcia błony i poprzez to pobudzenie 'dostrajają' się do słyszanej częstotliwości.




Rysunek 1.1 Schemat budowy ślimaka [14].

Wykorzystanie wiedzy na temat budowy aparatu słuchowego zaowocowało zdefiniowaniem skal perceptualnych (psychoakustycznych).

### Skala Mela

Skala Mela jest nieliniową skalą częstotliwości powstałą poprzez subiektywne badanie psychoakustyczne. Relację pomiędzy skalą Mela a liniową skalą częstotliwości przybliża się zwykle wzorem:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right), \tag{1.2}$$

gdzie m oznacza wartość częstotliwości w Melach.

### Skala Barka

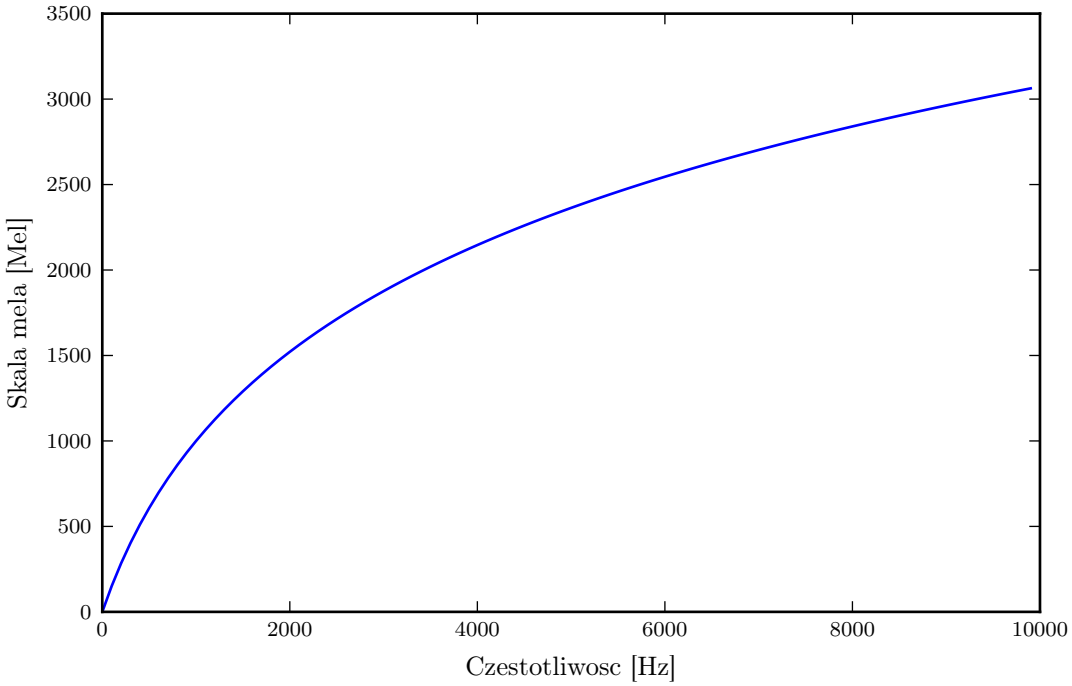
Inną skalą perceptualną opartą na modelu ślimaka jako banku liniowych filtrów w skali częstotliwości jest skala Barka. Wyróżnia ona 24 pasma krytyczne. Zakłada się, że 1 bark odpowiada 100 Melom, zaś funkcją przekształcenia ze skali liniowej jest:

$$B = 13 \arctan\left(0.00076 \cdot f\right) + 4.5 \arctan\left(\frac{f}{7500}^2\right), \tag{1.3}$$

gdzie B oznacza wartość w Barkach.

### Krzywe jednakowej głośności

Ze względu na różną sztywność błony podstawnej, człowiek z inną intensywnością odbiera bodźce dźwiękowe związane z różnymi częstotliwościami. Z pomiarów subiektywnych na większej populacji wyznaczone zostały krzywe jednakowej głośności. Stosowane są one w niektórych technikach ekstrakcji cech (np. 1.6) we wstępnym wzmocnieniu sygnału mowy.



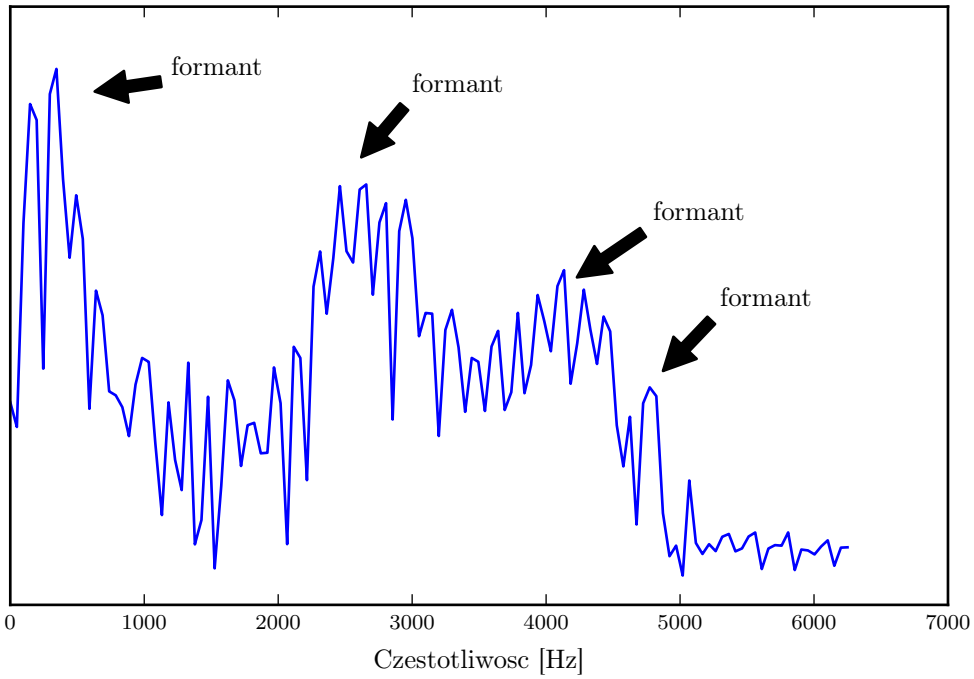

Rysunek 1.2 Relacja pomiędzy skalą Mela i liniową skalą częstotliwości.

### Układ nerwowy

Dziedzina rozpoznawania głosu jest szczególnie zainteresowana naturalną aparaturą w którą wyposażony jest człowiek dla zadania rozpoznawania mowy czy identyfikowania swoich rozmówców. Dotyczy to także całego obszaru metod rozpoznawania sygnałów akustycznych. Trend jest szczególnie widoczny w związku z bardzo dynamicznym rozwojem dziedziny sztucznych sieci neuronowych i sztucznej inteligencji. Tak jak wspomniano wyżej, szereg metod związanych z rozkodowywaniem sygnału mowy - ekstrakcją cech - korzysta z wiedzy anatomicznej przy okazji konstrukcji algorytmów. Niestety w dziedzinie klasyfikacji mówców to podejście nie jest stosowane - prawdopodobnie ze względu na skromną wiedzę o tym, jak funkcjonuje ludzki umysł. Wydaje się jednak, że można przywołać pewną wiedzę na temat tego, jak człowiek interpretuje sygnał mowy poprzez wskazanie pewnych obszarów mózgu odpowiedzialnych za zadania związane z analizą mowy - a także połączenia pomiędzy nimi. Wraz z zastosowaniem rezonansu magnetycznego (MRI) neurologia poczyniła duże postępy i można mieć nadzieję na to, że w najbliższej przyszłości uzyskana wiedza pozwoli na udoskonalenie systemów rozpoznawania głosu.

## 1.5.3 Klasyfikacja języka

Na rysunku 1.3 znajduje się przykładowe widmo pojedynczej ramki sygnału. Obszary koncentracji energii w dziedzinie częstotliwości nazywane są formantami. Analiza zmiany pozycji formantów pomiędzy kolejnymi fonemami okazuje się być bardzo ważna w zadaniu rozpoznawania mówców.




Rysunek 1.3 Przykładowe spektrum pojedynczej ramki sygnału mowy.

### Fonetyka

Fonetyka zajmuje się badaniem dźwięków produkowanych przez aparat mowy człowieka. Elementarnym dźwiękiem rozpatrywanym przez fonetykę jest głoska. Z punktu widzenia lingwistyki jest to najmniejszy segment sygnału mowy. Budowa i funkcjonalność narządu mowy determinują zakres produkowanych głosek. Fonetyka bada podstawowe dźwięki mowy bez rozróżniania ze względu na konkretny język czy znaczenie głoski. Fonem jest najmniejszą jednostką mowy, na podstawie której możliwa jest interpretacja jej znaczenia. To znaczy, że jest semantycznie istotna. Fonem rozpatruje się ze względu na znaczenie w konkretnym języku. Głoska może być realizacją fonemu. Dwie różne głoski mogą stanowić realizację tego samego fonemu - to znaczy nieść tę samą informację semantycznie. Zbiór takich głosek realizujących dany fonem nazywany jest alofonem.

Z punktu widzenia mechanizmów powstawania dźwięków, w ludzkim narządzie mowy można wyróżnić trzy, z których zbudowany jest każdy emitowany dźwięk mowy. Składa się na nie:

- dźwięk rezonujący powstały w wibrującym źródle (np. drgające fałdy głosowe) i rezonujący w przestrzeni rezonansowej na którą składają się drogi oddechowe znajdującą się powyżej krtani,
- dźwięk powstały przez nielaminarny przepływ powietrza,
- dźwięk impulsowy powstały przez energiczne wypuszczenie powietrza z układu oddechowego.

## 1.6 Ekstrakcja cech sygnału mowy

### 1.6.1 Przegląd

Celem ekstrakcji cech z sygnału mowy (*ang. feature extraction*) jest uzyskanie zbioru cech charakteryzujących sygnał ludzkiej mowy za pomocą technik cyfrowego przetwarzania sygnału. Jednocześnie jest to zamiana sygnału, w którym zawarta jest redundantna informacja, na sygnał o niskiej zawartości informacji znaczących dla problemu rozpoznawania mówcy/mowy.

Przedstawione w tym podrozdziale techniki ekstrakcji cech okazują się równie przydatne i powszechnie stosowane zarówno dla technik rozpoznawania mówcy, jak i rozpoznawania mowy. Najpopularniejszą obecnie stosowaną metodą ekstrakcji cech z sygnału mowy jest współczynników cepstrum w dziedzinie częstotliwości Mela - MFCC (*ang. Mel-Frequency Cepstral Coeficcients*).

### 1.6.2 Współczynniki MFCC.

Ekstrakcja cech za pomocą współczynników cepstrum w dziedzinie częstotliwości Mela składa się z dwóch głównych etapów: uzyskania współczynników mocy w dziedzinie częstotliwości Mela na podstawie estymacji widma mocy sygnału mowy (*ang. freqency warping*) oraz obliczenia współczynników cepstrum na podstawie uzyskanych wcześniej współczynników mocy w skali Mela. Wynikiem przeprowadzonej operacji jest uzyskanie wektora cech $x_j$ dla numeru ramki $j$.

#### DSTFT - dyskretna krótkoczasowa transformata Fouriera

Dyskretna krótkoczasowa transformata Fouriera (DSTFT - *ang. Discrete Short-time Fourier Transform*) jest dyskretną wersją ciągłej transformaty Fouriera (STFT - *ang. Short-time Fourier Transform* dla sygnału próbkowanego w czasie oraz dyskretnego w dziedzinie częstotliwości. Jest to metoda analizy czasowo-częstotliwościowej, przeprowadzającej operację dyskretnej transformaty Fouriera (DFT - *ang. Discrete Fourier Transform*) na ramkach sygnału, dla których możemy założyć jego lokalną stacjonarność. Relacja pomiędzy STFT, a DSTFT jest relacją analogiczną do relacji ciągłej transformaty Fouriera, a DFT. Estymacja widma za pomocą metody DSTFT jest podstawą techniki ekstrakcji cech MFCC.

#### Podział sygnału na ramki

Ze względu na własność quasi-stacjonarności sygnału mowy, aby wydobyć informację dotyczącą wypowiadanej głoski konieczne jest rozdzielenie sygnału na ramki. Liczbę próbek dla pojedynczej ramki wybiera się ze względu na konieczność uzyskania lokalnej stacjonarności - w taki sposób, aby było możliwe zbadanie charakterystyki częstotliwości pojedynczej głoski. Średnia długość głoski to 80 ms. Jednak trzeba mieć na względzie fakt, że samogłoski trwają długo w stosunku do przerw pomiędzy nimi (trwających zwykle ok. 5 ms). Zatem aby móc uchwycić krótsze głoski oraz przerwy zwykle ustala się długość ramki na 20 do 30 ms. Jednocześnie długość ramki wyrażona liczbą próbek jest funkcją częstotliwości próbkowania. Do przedstawionych założeń dochodzi zwykle warunek dotyczący użycia algorytmu szybkiej transformaty Fouriera (FFT - *ang. Fast Fourier Transform*) wymagającej aby sygnał był długości równej $2^r$, gdzie $r$ - liczba całkowita. W

przypadku pierwszej oraz ostatniej ramki stosowana jest technika uzupełniania zerami w przypadku braku próbek do zapełnienia całej ramki.

### Aplikacja okna na ramki

W wykorzystywanej technice DSTFT stosowane są zazwyczaj okna inne niż okno prostokątne, o takiej samej długości jak ramki, w celu zredukowania przecieku widma. Zastosowanie okna czasowego odbywa się poprzez przemnożenie wartości próbek ramki przez wartości okna w następujący sposób:

$$h_i = w_i \cdot x_i, \quad i \in 1, ..., N \tag{1.4}$$

gdzie $x_i$ oznacza $i$-tą próbkę sygnału, $h_i$ - zmodyfikowaną wartość próbki użytej dla wyznaczania współczynników DFT, $w_i$ - $i$-tą wartości funkcji okna, $N$ ilość próbek w ramce.

Najpopularniejszymi oknami stosowanymi w tym przypadku są okna czasowe Hanna, Hamminga oraz okna Gaussowskie [1].

### Estymacja widma

Po zastosowaniu okien czasowych dla kolejnych ramek sygnału, kolejnym etapem procedury obliczania współczynników MFCC jest znalezienie widma ramki za pomocą dyskretnej transformaty fouriera (DFT). Kolejne współczynniki uzyskanego widma zdefiniowane są następująco:

$$H_k = \sum_{n=0}^{N-1} h_n exp(\frac{-2j\pi kn}{N}) \tag{1.5}$$

W omawianej metodzie korzysta się jedynie z informacji o amplitudzie uzyskanego widma. Kolejne współczynniki widma amplitudowego zdefiniowane są w następujący sposób:

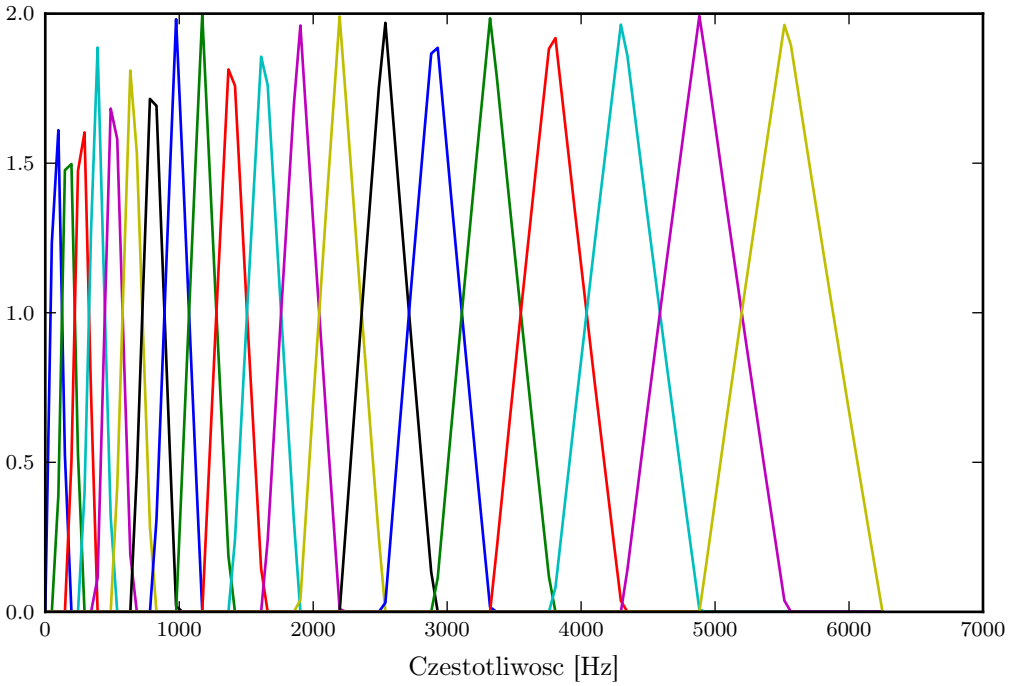
$$M_k = |H_k| = \sqrt{|H_k|^2} \tag{1.6}$$

### Przejście do dziedziny częstotliwości Mela

Przejście do dziedziny częstotliwości Mela (*ang. frequency warping*) uzasadnione jest nieliniową percepcją układu słuchowego człowieka. Ponieważ procedura DFT oblicza widmo z liniową skalą częstotliwości, potrzebna jest nieliniowa transformacja do dziedziny częstotliwości Mela. Jest to proces imitujący cechę sygnału akustycznego - wysokości tonu. Rozpatrywana operacja polega na zastosowaniu banku filtrów trójkątnych na uzyskanych amplitudach widma:

$$\tilde{H}_j = \sum_{k=1}^{K} H_k \cdot F_k, \quad k \in 1...N, \quad j \in 1...M \tag{1.7}$$

gdzie $\tilde{H}_j$ oznacza j-ty współczynnik amplitudy widma w dziedzinie częstotliwości Mela, $F_k$ współczynniki filtru, zaś $M$ oznacza liczbę filtrów. W wyniku tej operacji otrzymuje się wektor współczynników częstotliwości w skali Mela o długości równej ilości użytych filtrów. Najczęściej stosowanym filtrem jest filtr o trójkątnej charakterystyce częstotliwościowej. Każdy kolejny filtr zaczyna się w środku pasma wcześniejszego. Ponieważ przetwarzany jest sygnał składający się z próbek o wartościach rzeczywistych, pokryte pasmo częstotliwości zawiera próbki od 0-wej do próbki oznaczonej numerem N/2. Liczba filtrów oraz ich rozmieszczenie w dziedzinie częstotliwości ustalane są na jeden z kilku sposobów.




Rysunek 1.4 Przykładowe 24 trójkątne filtry dla sygnału o częstotliwości próbkowania *fs=12500 Hz.*

Pierwszym sposobem jest skorzystanie z 24 obszarów krytycznych zdefiniowanych w skali Barka. Każdy filtr ma niezerowe wartości w zakresie częstotliwości od k-1 częstotliwości krytycznej, aż do k+1 częstotliwości krytycznej. Maksimum zaś przypada na k-tą częstotliwość krytyczną. Drugim i najpopularniejszym podejściem jest zastosowanie ustalonej ilości filtrów i rozmieszczenie ich równomiernie w dziedzinie częstotliwości Mela (rysunek 1.6.2).

Stosowane są również filtry trapezowe oraz kosinusowe, a także zmiana wzmocnienia filtrów ze względu na psychoakustyczną krzywą jednakowej głośności.

### Uzyskanie współczynników głośności

Wartości otrzymane w procesie przejścia do dziedziny częstotliwości Mela z prążków reprezentujących widmową gęstość mocy sygnału reprezentują natężenie dźwięku w danym momencie czasu. Ponieważ zadaniem ekstrakcji cech przy pomocy współczynników MFCC jest naśladowanie ludzkiego układu percepcji konieczne jest przekształcenie wartości reprezentujących energię na głośność (*ang. magnitude warping*). Zgodnie z definicją *głośność* jest funkcją zarówno *wysokości tonu* jak i natężenia dźwięku $I$. Relację z *wysokością tonu* modeluje się za pomocą etapu *equalizacji* sygnału. Zatem pozostaje jedynie zdefiniować współczynniki głośności $C_k$ [1] dla każdej ramki sygnału:

$$\tilde{C_k} = 10 \cdot \left( \frac{|\tilde{H_k}|^2}{I_0} \right) \tag{1.8}$$

Ponieważ normalizacja współczynników widma mocy do natężenia odniesienia $I_0$ nie

wpływa na efekt identyfikacji mówcy dlatego w praktyce stosuje się dla wygody postać:

$$C_k = log\left(|\tilde{H}_k|^2\right) \tag{1.9}$$

Przy użyciu tych współczynników otrzymuje się cepstrum sygnału.

### Cepstrum

Funkcja cepstrum mocy dla ciągłego przekształcenia Fouriera zdefiniowana jest następująco [1]:

$$\tilde{h_{pc}} = \left[\frac{1}{2\pi}\int_{-\pi}^{\pi} log\left(|H(\omega)|^2\right) e^{i\omega t}\right]^2 \tag{1.10}$$

gdzie $|H(\omega)|^2$ oznacza funkcję widma gęstości mocy sygnału oryginalnego. Jest to odwrotne przekształcenie Fouriera zastosowane na logarytmie funkcji gęstości mocy sygnału.

Pojęcie cepstrum pierwotnie zostało wprowadzone do badań nad echem sygnałów akustycznych [12]. Taki sygnał opisywany jest przez wyrażenie

$$x(t) = s(t) + \alpha s(t-\tau) \tag{1.11}$$

gdzie $\alpha$ jest współczynnikiem osłabienia sygnału echa przesuniętego w czasie o $\tau$ względem sygnału oryginalnego. Reprezentacja częstotliwościowa zaś przyjmuje w takim wypadku postać

$$|S(f)|^2\left[1+\alpha^2+2\alpha cos(2\pi f\tau)\right]. \tag{1.12}$$

Oznacza to, że widmo sygnału oryginalnego $S(f)$ modulowane jest przez zmienny w częstotliwości komponent $[1+\alpha^2+2\alpha cos(2\pi f\tau)]$. Znajomość tego w jaki sposób modulowana jest obwiednia widma pozwala na określenie współczynników $\alpha$ oraz $\tau$ definiujących sygnał echa. Ponieważ problem taki jest doskonale opisany w dziedzinie czasu przy analizie sygnałów zmodulowanych, narzucającym się rozwiązniem jest skorzystanie z tych znanych już technik. Korzystając z własności logarytmu możliwe jest rozdzielenie dwóch komponentów - zamianę mnożenia na dodawanie, a poprzez zastosowanie odwrotnego przekształcenia Fouriera otrzymuje się sumę dwóch funckji w dziedzinie tzw. "quefrency":

$$\begin{aligned}\mathcal{F}^{-1}\{log(X(f))\} = \mathcal{F}^{-1}\{log(S(f))\} + \mathcal{F}^{-1}\{log(1+\alpha^2)\} + \\ + \mathcal{F}^{-1}\{log(1+\frac{2a}{1+a^2}cos(\omega\tau))\}.\end{aligned} \tag{1.13}$$

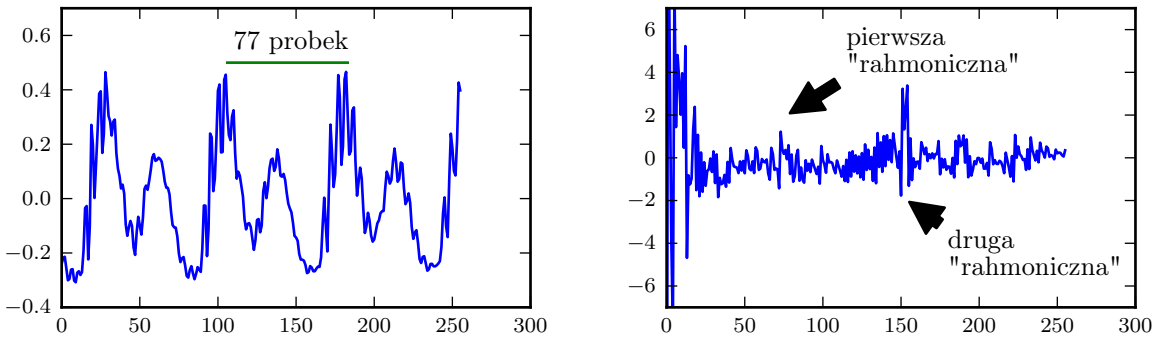
Ostatni składnik powyższej sumy objawia się w postaci widocznego w cepstrum skupionego impulsu którego położenie w skali cepstrum określa przesunięcie $\tau$ zaś jego amplituda jest związana z czynnikiem osłabienia $\alpha$.

Okazuje się, że problem wykrycia wysokości tonu (*ang. pitch detection*) sygnału mowy jest podobny do problemu analizy echa sygnału akustycznego. Zaproponowana została więc wersja wykorzystująca cepstrum dla analizy STDFT [11]. Prosty model generacji sygnału mowy zakłada, że jest on wynikiem splotu odpowiedzi impulsowej dróg głosowych $h_1$ oraz quasi-okresowym sygnałem impulsowym $h_2$ (*ang. quasi-periodic pulse train*) - wymuszeniem generowanym przez głośnię:

$$h(t) = h_1(t) * h_2(t) \tag{1.14}$$

Z własności ciągłego przekształcenia Fouriera wynika, że powyższe równanie w dziedzinie częstotliwości przybiera postać iloczynu transformat $H_1(\omega)\cdot H_2(\omega)$. W dziedzinie cepstrum rozdziela się natomiast na sumę składników:

$$\tilde{h}(t) = log(H_1(\omega)) + log(H_2(\omega)) \tag{1.15}$$




Rysunek 1.5 Ramka sygnału wypowiadanej samogłoski 'i' oraz jej cepstrum.

Poprzez obliczenie cepstrum sygnału mowy otrzymujemy zatem rozdzielenie komponentów pochodzących od funkcji $h_1(t)$ i $h_2(t)$. Proces ten nazywany jest również dekonwolucją (*ang. homomorphic deconvolution*) [12] i zobrazowany jest na rysunku 1.5. Niskie współczynniki cepstrum opisują charakterystykę aparatu głosowego, zaś widoczne w okolicach próbki 77 i 150 piki stanowią harmoniczne w dziedzinie cepstrum (*ang. rahmonics*) i świadczą o okresie sygnału wymuszenia głośni ($h_2$), którą można odczytać z pierwszego wykresu jako odległość w próbkach pomiędzy najwyższymi szczytami - 77 próbek.

Należy zwrócić uwagę, że wykorzystywany w tej metodzie jest logarytm z liczb rzeczywistych oraz współczynniki mocy widma. Spowodowane jest to tym, że podczas procesu rozpoznawania mówcy nie interesuje nas rekonstrukcja sygnału, ani nie korzysta się z informacji zachowanej w fazie sygnału. Z tego też powodu klasycznie w ostatnim etapie obliczania współczynników MFCC zamiast odwrotnej dyskretnej transformaty Fouriera (IDFT) stosuje się dyskretną transformację kosinusową (DCT). Kolejną korzyścią jest otrzymanie w wyniku przeprowadzenia tej transformacji współczynników będącymi liczbami rzeczywistymi. W niniejszej pracy korzysta się z następującej definicji dyskretnej transformaty kosinusowej:

$$H_k = \sum_{n=0}^{N-1} h_n cos\left(\frac{\pi(2n+1)k}{2N}\right) \tag{1.16}$$

gdzie

$$a_k = \begin{cases} 1/N & \text{dla} \quad k = 0 \\ 2/N & \forall \quad k > 0. \end{cases} \tag{1.17}$$

Uzyskane współczynniki MFCC noszą nazwę wektora akustycznego (*ang. acoustic vector*) reprezentującego cechy pojedynczej ramki i są w procesie weryfikacji mówcy dalej wykorzystywane w tzw. procesie dopasowywania cech opisanym w sekcji 1.7.

W wektorze cech zwykle stosuje się od 12 do 15 najniższych współczynników uzyskanych z DCT. Zwykle uzupełnia się go dodatkowymi współczynnikami $\Delta$ oraz $\Delta^2$ zawierającymi dodatkową informację o dynamice sygnału.

### Współczynniki Delta cepstrum oraz Delta-Delta cepstrum

Duże znaczenie w procesie rozpoznawania mówcy ma informacja temporalna zawarta w sygnale mowy - związana z dynamiką zmian współczynników kolejnych ramek wektorów cech. Najpopularniejszą metodą na dodanie informacji tego rodzaju jest dołączenie do wektora cech MFCC współczynników *Delta* - $\Delta$ oraz *Delta-Delta* - $\Delta^2$ które kolejno reprezentują pierwszą i drugą pochodną dyskretną dla kolejnych ramek sygnału. Okazuje się, że

dodanie wspomnianych współczynników daje znaczącą poprawę procesu rozpoznawania, szczególnie w systemach rozpoznawania mówcy z hasłem lub zależnego od wypowiadanego tekstu [9]. Najczęściej stosuje się następującą formułę do obliczenia cech delta cepstrum:

$$\Delta_{t,k} = \frac{\sum_{n=1}^{N} n(c_{t+n,k} - c_{t-n,k})}{2\sum_{n=1}^{N} n^2} \tag{1.18}$$

gdzie $N$ przyjmuje typową wartość 2, $C_{t,k}$ oznacza współczynnik cepstrum dla ramki $t$ i numerze współczynnika w ramce $k$. Współczynniki $\Delta - \Delta$ obliczane są w ten sam sposób za wyjątkiem użycia współczynników Delta w miejsce współczynników cepstrum:

$$\Delta^2_{t,k} = \frac{\sum_{n=1}^{N} n(\Delta_{t+n,k} - \Delta_{t-n,k})}{2\sum_{n=1}^{N} n^2} \tag{1.19}$$

Do wektora cech MFCC dodaje się zwykle taką samą liczbę współczynników $\Delta$ oraz $\Delta^2$ co współczynników cepstrum $c_k$. W systemach rozpoznawania mówcy bez znajomości wypowiadanego tekstu proponuje się ograniczyć liczbę współczynników delta w stosunku do współczynników cepstrum i współczynników $\Delta - \Delta$ w stosunku do współczynników $\Delta$[1].

### 1.6.3 LPCC

Współczynniki liniowego kodowania predykcyjnego cepstrum - LPCC (*ang. linear predicting ceptstral coding*) są alternatywną metodą ekstrakcji cech z sygnału mowy. Jest to synteza metody kodowania LPC wraz z zastosowaniem współczynników cepstrum. Liniowe kodowanie predykcyjne (LPC) jest techniką opartą na zastosowaniu modeli autoregresyjnych (AR). Główną różnicą w stosunku do techniki MFCC jest więc sposób aproksymacji widmowej gęstości mocy przy pomocy współczynników $|H_k(\omega)|^2$. Kroki wstępnego przetwarzania poprzedzające estymację, łącznie z aplikacją na ramkę okna czasowego, są identyczne jak w przypadku procedury obliczania współczynników MFCC. Zamiast posługiwać się dyskretnym przekształceniem Fouriera (DFT), korzysta się z modelu autoregresyjnego, kształtując widmo przy pomocy transmitancji $H(z)$ posiadającej same bieguny (*ang. all-pole*) i jednym zerem w punkcie $z = 0$:

$$\tilde{H}(z) = \frac{\tilde{G}}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{1.20}$$

gdzie $a_k = \frac{\alpha_k}{\alpha_0}$, $\tilde{G} = \frac{1}{\alpha_0}$, a współczynniki $\alpha_0, ..., \alpha_p$ oznaczają współczynniki modelu AR, nazywane również w rozpatrywanym przypadku jako współczynniki predykcji - PC (*ang. predictor coefficients*).

W dziedzinie czasu modele liniowej predykcji (LP) określają zależność próbki sygnału $s_n$ poprzez liniową kombinację wcześniejszych $p$ próbek przeskalowanych przez współczynniki predykcji - PC:

$$s_n = -\sum_{k=1}^{p} a_k \cdot s_{n-k} + \tilde{G} \cdot u_n \tag{1.21}$$

W metodach liniowej predykcji zazwyczaj ignoruje się wartość $u_n$ oznaczającą obecną próbkę wejściową - w przypadku mowy oznaczającą sygnał tonu krtaniowego. W ten sposób otrzymuje się wektor współczynników $a_1, ..., a_n$ modelujących danego mówcę. Popularną metodą estymacji tych wartości jest minimalizowanie wartości błędu średniokwadratowego wartości resztowych (rezydua):

$$V = \sum_n e_n^2 = \sum_n \left\{ s_n + \sum_{k=1}^{p} \alpha_k \cdot s_{n-k} \right\}. \tag{1.22}$$

Kryterium minimalizacji

$$\frac{\delta E}{\delta \alpha_i} = 0, \quad i = 1, .., p \tag{1.23}$$

daje w wyniku zależność:

$$\sum_{k=1}^{p} a + k \cdot \sum_{n} s_{n-k} s_{n-i} = -\sum_{n} s_n s_{n-i}, \quad i = 1, ..., p. \tag{1.24}$$

Lewa suma $\sum_n s_{n-k}s_{n-i}$ jest funkcja autokorelacji o opóźnieniu $\tau = n-k$, czyli $r(|n-k|)$, zaś analogicznie prawa jest funkcją autokorelacji o opóźnieniu $\tau = n$ tzn. $r(n)$, gdzie w obu przypadkach $i = 1, ..., p$. Układ tych równań nazywany jest równaniami Yule'a-Walkera (*Yule-Walker Equations*). Otrzymana z tych równań macierz autokorelacji $\boldsymbol{R}$ przyjmuje postać macierzy Toeplitza - posiada te same wartości na poszczególnych przekątnych. Rozwiązanie problemu minimalizacji przedstawia się jako:

$$\boldsymbol{\alpha} = \boldsymbol{R^{-1}r}, \tag{1.25}$$

gdzie wektor $\boldsymbol{r}$ jest wektorem kolumnowym prawych sum równań (1.24) tzn. autokorelacji $r(n)$, zaś wektor $\boldsymbol{\alpha}$ wektorem kolumnowym współczynników $\alpha_1, ..., \alpha_n$. Wspomniana własność macierzy $\boldsymbol{R}$ jako macierzy Toeplitza w dużym stopniu upraszcza procedurę odwracania macierzy poprzez zastosowanie rekursywnego algorytmu *Levinsona-Durbina*. Możliwy do zastosowania algorytm Shura pozwala przyspieszyć obliczenia poprzez paralelizację. Obliczone wartości $\alpha_i$ reprezentują współczynniki LPC. Wraz z wartością:

$$\tilde{G} = r(0) - \sum_{i=1}^{p} a_i r(i) \tag{1.26}$$

umożliwiając aproksymację funkcji gęstości widma przy pomocy równania (1.20).

#### Obliczanie współczynników LPCC

W praktyce [1] jako wektory cech najczęściej używa się współczynników cepstrum obliczonych przy pomocy uzyskanych współczynników LPC. Rekursywny algorytm zaproponowany przez Rabinera i Juanga [10] oblicza współczynniki cepstrum w następujący sposób:

1. $c_0 = log(\tilde{G}^2)$,
2. $c_d = \alpha_d + \sum_{i=1}^{d-1} c_i a_{d-i}$ dla $1 \leqslant d \leqslant p$
3. $c_d = \sum_{i=1} d + \sum_{i=1}^{d-1} c_i a_{d-i}$ dla $d > p$

Rekomendowana wartość $p$ dla obliczania współczynników LPCC powyższym sposobem to zakres od 8 do 16 [1]. Najczęściej stosuje się liczbę 8 do 20 elementów w wektorze cech.

### 1.6.4 Inne metody ekstrakcji cech

Najczęściej stosowanymi metodami ekstrakcji cech są następujące: MFCC, LPCC, LSF, PLP [15]. Dwie pierwsze zostały omówione wcześniej bardziej szczegółowo.

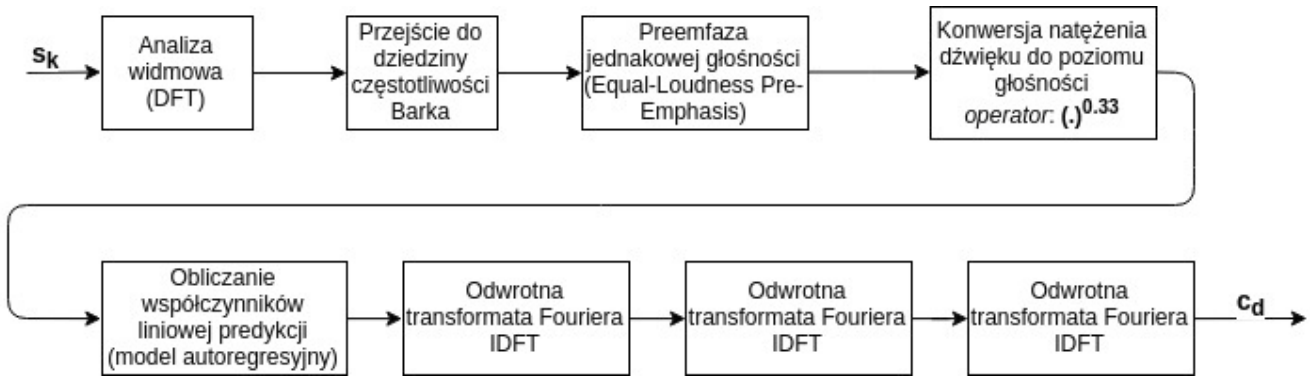
### Metody bazujące na LP

Istnieje szereg innych metod opartych na współczynnikach LP. Zwykle cechuje je nieliniowa transformacja dziedziny czasu/częstotliwości oraz amplitud współczynników mocy widma, która na celu ma przybliżyć sposób funkcjonowania percepcji ludzkiego aparatu słuchowego w konstruowanym systemie.

- PARCORs (*ang. partial correlation coefficients*) - *współczynniki częściowej korelacji* są współczynnikami cech $\kappa_k$ interpretowanymi jako wartość reprezentującą nieredundantną informację o autokorelacji [1]. Wielkości te są obliczane jako wartości pośrednie w algorytmie obliczania współczynników LPC we wcześniej wspomnianym (1.6.3) algorytmie Levinsona-Durbina. Współczynniki te również mają interpretację fizyczną jako modelujące wartości współczynników odbicia w modelu traktu głosowego przedstawionego jako połączenie cylindrów o różnych średnicach, przez które przepływa strumień powietrza z płuc [1]. Z tego też powodu nazywane są także *współczynnikami odbicia* (*ang. reflection coefficients*).

- LAR (*ang. Log Area Ratios*) - wielkości wynikające z modelu cylindrów tak jak w metodzie PARCORs. Współczynniki $g_i$ otrzymuje się jako stosunki logarytmów średnic kolejnych cylindrów modelu. Tak obliczone współczynniki cechują się bardziej równomierną wrażliwością w dziedzinie częstotliwości niż współczynniki odbicia [7].

- PLP (*ang. Perceptual Linear Predictive analysis*) - metoda ekstrakcji cech oparta na współczynnikach predykcji liniowej (LP) inspirująca się metodą MFCC. Zamiast zastosowania skali Mela, w tej metodzie używa się skali Barka (rozdział 1.5.2), stosuje się zmodyfikowaną preemfazę ze względu na krzywą jednakowej głośności (*ang. equal-loudness pre-emphasis*) oraz dokonuje się transformacji widma amplitudowego poprzez podniesienie wartości do potęgi $\frac{1}{3}$ co powoduje wygładzenie widma - zmniejszeniem dynamiki. Jest nazywana konwersją natężenia dźwięku do poziomu głośności (*ang. Intensity to Louadness Conversion*) [2]. Po tych etapach, z tak przekształconego widma uzyskiwane są współczynniki predykcji liniowej (model AR). Tak jak w metodzie MFCC także i tutaj używa się procedury DFT i IDFT. Współczynniki cepstrum uzyskiwane są tak jak w metodzie LPCC, np. przy wykorzystaniu metody Levinsona-Durbina. Na rysunku 1.6 przedstawiono etapy analizy PLP tak jak w pracy źródłowej [6].




Rysunek 1.6 Etapy analizy PLP.

- LSF (*ang. line spectral frequencies*) lub LSP (*ang. line spectral pairs*) - metody reprezentacji sygnału mowy w których zaproponowana jest alternatywna postać

mianownika transmitancji z (1.20). Oznaczając mianownik jako $A(z)$ przyjmowana jest reprezentacja:

$$\begin{aligned} A(z) &= \frac{1}{2}[P(z) + Q(z)] \\ P(z) &= A(z) + z^{p+1}A(z^{-1}) \\ Q(z) &= A(z) - z^{p+1}A(z^{-1}). \end{aligned} \tag{1.27}$$

gdzie wielomian $P(z)$ jest interpretowany fizycznie jako odpowiadający odpowiedzi impulsowej dróg głosowych przy głośni zamkniętej zaś wielomian $Q(z)$ przy otwartej. Pierwiastki wielomianów $P$ i $Q$ leżą na okręgu jednostkowym na płaszczyźnie zespolonej $z$. Technika ta jest nazywana *ang. reverse-filtering.*

### Metody wykorzystujące transformację falkową

Podobnie jak dla krótko-czasowej transformacie Fouriera (STFT) (rozdział 1.6.2), niestacjonarny sygnał mowy można analizować metodami czasowo-częstotliwościowymi z odpowiednio małym oknem dobranym na podstawie czasu trwania pojedynczych głosek. Segmentacja sygnału na takie ramki pozwala zastosować metody estymacji widma dla sygnału stacjonarnego. Metody estymacji widma z zastosowaniem modeli autoregresyjnych (AR) również przyjmują, że działają na sygnale stacjonarnym. Transformacja falkowa dostarcza szeregu ortogonalnych funkcji jądra dla uogólnionego szeregu Fouriera. Wspomniane funkcje powstają poprzez rozciąganie i przemieszczanie funkcji matki dla danej transformacji falkowej. Warto zwrócić uwagę na to, że transformacja Gabora, będącą transformacją falkową z funkcją matką równą iloczynowi jądra transformaty Fouriera $exp(-2\pi jf\tau)$ oraz funkcją Gaussowską $exp(-\pi(\tau - t)^2)$ stanowi przypadek krótko-czasowej transformaty Fouriera (STFT) dla której użyto okna Gaussowkiego. Zatem widać, że istnieje pole do zastosowania analizy czasowo-częstotliwościowej w postaci transformacji falkowej w miejsce STFT czy estymacji za pomocą modeli autoregresyjnych. Odpowiednikiem dyskretnej transformaty Fouriera (DFT) jest dyskretna transformata falkowa (DWT -*ang. discrete wavelet transform*). Przykładami omawianych cech są:

1. MFCWC - *Mel-Frequency Discrete Wavelet Coefficients* - technika oparta na generacji współczynników MFCC - jest praktycznie identyczna w kolejnych krokach za wyjątkiem wykorzystania transformaty cosinusowej (DCT) w zamian za użycie dyskretnej transformaty falkowej (DWT). Okazuje się, że zastosowanie DWT przyczynia się do polepszenia rezultatów w 'agresywnie' hałaśliwym środowisku [1].

2. WOCOR - *Wavelet Octave Coefficients Of Residues* - jest wektorem cech wyekstrahowanych za pomocą transformacji falkowej z funkcją matką związaną z tonem wypowiadanej głoski, co wiąże się z rozmiarem analizowanej ramki. Jest to metoda oparta na współczynnikach liniowej predykcji (LP) rezyduł sygnału w ramce. Rozważana technika jest efektywna dla zastosowania z sygnałem mowy zawierającym języki oparte na intonacji (*ang. pitch-based*) dialektów języka Chińskiego - Mandaryńskiego czy Kantońskiego [1].

### Inne metody ekstrakcji cech

Istnieje szereg metod bazujących na przekształceniu dziedziny czy wartości współczynników funkcji gęstości widmowej sygnału za pomocą banku filtrów o różnych właściwościach. Zamiast banku filtrów Mela (MFCC) lub Barka (PLP) konstruowane są filtry bazujące na innych, bardziej specyficznych modelach ludzkiego narządu słuchu [1].

Większość opisanych wcześniej technik ekstrakcji cech z sygnału mowy są metodami określanymi jako widmowe i krótko-czasowe (*ang. Short-Term Spectral Features*) [15] z ramkami sygnału o długości od 20 do 30 ms.

Cechy temporalno-widmowe (*ang. Spectro-Temporal Features*) [15] są zbiorem wielkości dostarczających informacji na temat tożsamości mówcy zawartej w przejściach pomiędzy formantami (1.5.3) czy modulacją amplitudy współczynników widma w czasie. Jednym ze sposobów na uzyskanie tego typu informacji jest zastosowanie opisanych wcześniej współczynników Delta ($\Delta$) i Delta-Delta ($\Delta^2$) dla cech MFCC (1.6.2). Innymi skutecznymi technikami jest użycie współczynników regresji liniowej czy aproksymacji wielomianami dla kolejnych kilku wektorów cech (zwykle 2 lub 3) [13]. Nowszymi rozwiązaniami jest badanie wolnych modulacji amplitudy i częstotliwości w ramkach zawierających po ok. 10 wektorów cech, jak np. w metodzie TDCT (*Temporal Discrete Cosine Transform*).

Istnieją również cechy wysokiego poziomu, takie jak cechy prozodyczne (*ang. prosodic features*) które mogą identyfikować mówcę za pomocą rytmu, akcentu czy intonacji wypowiedzi. Opisywane w literaturze są również takie zestawy cech które operą nie na charakterystykach takich związanych ze stylem wypowiedzi czy zasobem słownictwa[15].

## 1.7 Metody klasyfikacji sygnału mowy.

Procesy rozpoznawania mówcy, a w szczególności weryfikacji mówcy są podklasą szerszego zagadnienia znanego z techniki jako dopasowania wzorca (*ang. pattern matching*). W ogólności celem przeprowadzonego zadania dopasowania wzorca jest określenie czy w dostarczonej na wejściu systemu sekwencji danych można znaleźć pewien znany wzorzec oraz określenie ilościowej relacji stopnia podobieństwa. W rozpatrywanym w tej pracy zadaniu weryfikacji mówcy oczekuje się, że element systemu odpowiedzialny za proces dopasowania wzorca zwróci wynik reprezentujący miarę podobieństwa wejściowego wektora cech do weryfikowanego modelu mówcy. Takie modele mówcy powstają z wektorów akustycznych dostarczonych jako zestaw danych uczących. W praktycznym systemie weryfikacji mówcy tak skonstruowane modele przechowywane są w postaci zaszyfrowanych danych. Następnie używane są do podjęcia decyzji o tym czy podający się za weryfikowaną osobę otrzyma dostęp do chronionych zasobów.

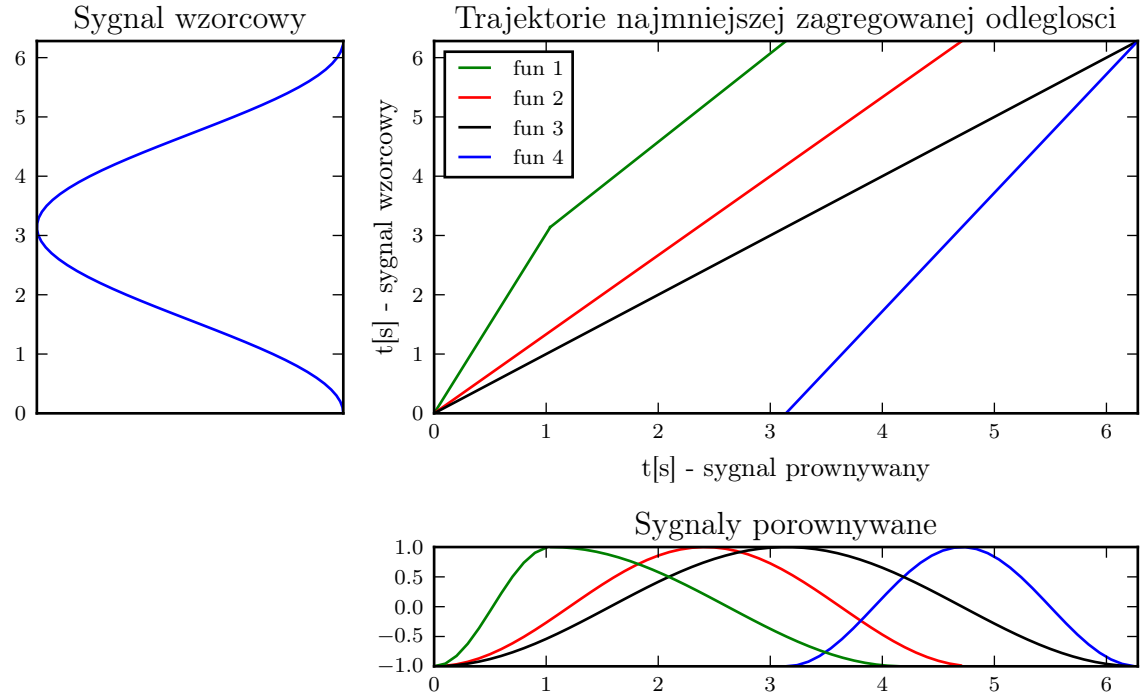
Obecnie dla etapu dopasowania wzorca - w klasycznym podejściu - stosowane są metody [7] [15]: ukryte modele Markowa (*ang. HMM - hidden Markov model*, dynamiczne dopasowanie czasowe (*ang. DTW - Dynamic Time Warping*), kwantyzacja wektorów (*VQ - vector quantization*, mikstury Gaussowskie (*GMM - Gaussian mixture model*), (*SVM - support vector machine*) oraz sztuczne sieci neuronowe (*ANN - artificial neural network*).

Wyróżnia się dwie grupy technik dopasowania wzorca: techniki deterministyczne (*ang. template models*) oraz techniki probabilistyczne (*probabilistic models*) [7].

W grupie technik deterministycznych, w trakcie trwania fazy modelowania (trenowania), z wprowadzonych danych uczących generowane są wektory wzorcowe w sposób zależny od wybranego algorytmu - mogą to być z góry narzucone wartości, uśrednione wartości wektorów czyli centroidy (*ang. centroids*), np:

$$\tilde{x} = \mu = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{1.28}$$

lub takie wzorcowe wektory mogą być ustalone w inny sposób. W tej klasie metod dokonuje się porównania w ten sposób uzyskanych wektorów wzorcowych z wektorami cech dostarczonych jako dane wejściowe w celu przeprowadzenia testowania, przy czym zakłada




Rysunek 1.7 Przykładowe trajektorie dla przebiegów podobnych przebiegów czasowych w technice DTW. Każdy punkt sygnałów może reprezentować pojedynczy wektor cech.

się, że [15] [7] każdy z nich jest nieidealną repliką drugiego. Stopień podobieństwa określany czyli funkcja miary dopasowania (*ang. match function*), różni się w zależności od zastosowanej metody. Do tego typu technik należy niezmodyfikowana metoda kwantyzacji wektorów - VQ, a także dynamiczne dopasowanie czasowe - DTW.

W drugiej grupie - technik probabilistycznych, każdemu mówcy przyporządkowuje się warunkową funkcję gęstości prawdopodobieństwa $p(X|\lambda)$, gdzie $\lambda$ oznacza model mówcy. W fazie uczenia systemu dokonuje się estymacji parametrów takiej funkcji gęstości prawdopodobieństwa na podstawie danych wejściowych - wektorów cech reprezentujących próbkę wypowiedzi mówcy. W fazie testowania najczęściej oszacowuje się prawdopodobieństwo tego czy zbiór lub sekwencja wektorów pochodzących z ekstrakcji testowanej wypowiedzi należy do zarejestrowanego modelu reprezentowanego przez ustaloną wcześniej funkcję gęstości prawdopodobieństwa. Może to odbywać się poprzez obliczenie iloczynu prawdopodobieństw uzyskanych z funkcji gęstości w realizacjach zmiennej losowej reprezentowanych przez wektory wejściowe przy założeniu, że są zdarzeniami niezależnymi. Do zbioru technik probabilistycznych należą m.in. techniki ukrytych model Markova - HMM oraz mikstur Gaussowskich - GMM.

### 1.7.1 Dynamiczne dopasowanie czasowe - DTW.

Dynamiczne dopasowanie czasowe (DTW) jest jedną z najstarszych metod dopasowania wzorca [13] wykorzystywana zarówno w dziedzinie rozpoznawania mowy jak i rozpoznawania mówcy. Metoda ta służy szczególnie w wypadkach kiedy potrzebna jest kompensacja różnic w długości porównywanych sygnałach. Dane wejściowe są traktowane jako sekwencja, a więc ważna jest kolejność ich dostarczania - jest to metoda zależna od czasu (*time dependent*). Z tego względu możliwe jest jej zastosowanie tylko w rozpo-

znawaniu mówcy zależnym od tekstu (*text-dependent*) lub z wyświetlanym hasłem (*time-prompted*). Podczas fazy testowania sekwencja wektorów akustycznych weryfikowanego mówcy $(\tilde{x_1}, .., \tilde{x_N})$ jest porównywana z z sekwencją wzorcową $(\tilde{x_1}, .., \tilde{x_M})$. Wymiary $M$ i $N$ najczęściej różnią się od siebie dla problemu rozpoznawania mowy oraz mówcy, co czyni tę metodę użyteczną dla tego zastosowania. W przypadku gdy $M = N$, problem redukuje się do obliczenia odległości dla pewnej zdefiniowanej metryki $d(x_i, \tilde{x})$ na przestrzeni rozpatrywanych sygnałów. W metodzie dynamicznego dopasowania czasowego wartość dopasowania sygnału wyraża się jako suma:

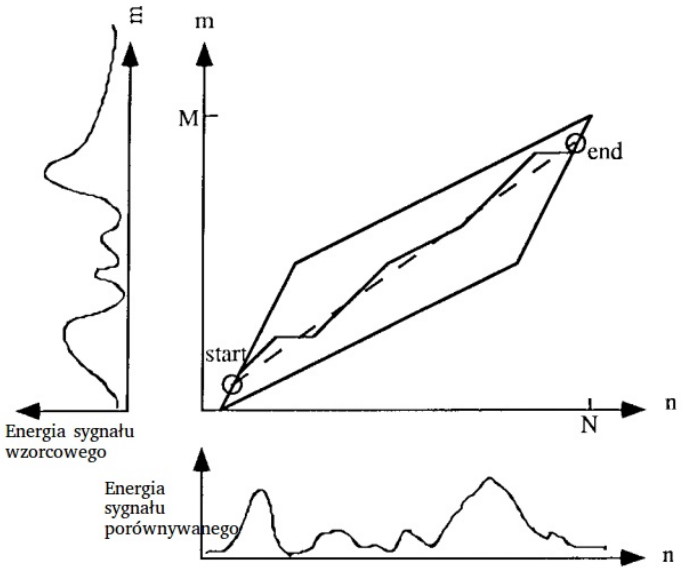
$$z = \sum_{i=1}^{M} d(x_i, x_{\tilde{j(i)}}). \tag{1.29}$$

gdzie: $i \in M, j \in N$, oraz $j(i)$ jest funkcją indeksu $i$.

Opisane przekształcenie dokonuje mapowania: $j(i)$ - które tworzy pary iloczynów próbek sygnału wzorcowego i sygnału porównywanego poprzez zmianę indeksowania kolejnych próbek tego drugiego. Sposób tego przekształcenia jest zależny od wybranego algorytmu realizującego technikę DTW. Najczęściej jest to krzywa szukająca najmniejszej zagregowanej odległości. Założeniem jest wybór takiego sposobu przekształcenia (trajektorii na wykresie) aby minimalizować funkcję miary dopasowania (1.29). Zatem uzyskana droga powinna prezentować sekwencje iloczynów wspomnianych dwóch sygnałów, które w sumie dają najmniejszy wynik. Wyidealizowane (idealnie dobrana droga przy założeniu bardzo dużej ilości próbek) trajektorie reprezentujące takie przekształcenia zaprezentowane są na rysunku 1.7. Wykres reprezentuje iloczyn kartezjański dziedzin obu sygnałów. Na tej przestrzeni rozpięta jest funkcja metryki $d(\tilde{x}(t), x(\tau))$. Porównywane sygnały są przesuniętym i przeskalowanym sygnałem wzorcowym w dziedzinie czasu. Sygnał *fun 3.* jest sygnałem wzorcowym zatem przekształcenie j(i) jest przekształceniem identycznościowym. Sygnał *fun 2.* jest przeskalowany w czasie o wsp. 1.5 i staje się funkcją liniową. *Fun 3.* jest dodatkowo przesunięty względem sygnału wzorcowego w czasie, zaś sygnał *fun 1.* jest w obu swoich połowach przeskalowany przez inny czynnik, z tego powodu jego trajektoria jest linią łamaną na wykresie. W ogólności znalezione przekształcenie nie jest funkcją - algorytm zazwyczaj pozwala na krok w kierunku równoległym do osi czasu sygnału wzorcowego. Dla sygnału uzyskana trajektoria nie jest linią prostą co widać na rysunku 1.8 gdzie sygnałami wzorcowym i porównywanym są bardziej rzeczywiste przebiegi. Na tym rysunku widać również często stosowane ograniczenia dla algorytmu wykrywania drogi - otóż grubszą linią zaznaczony jest obszar dozwolony przez algorytm do prowadzenia trajektorii. Dodatkowo punkty początkowe i końcowe są ustalone "na sztywno". Klasycznym i prostym podejściem algorytmicznym jest wybór punku startowego z ograniczonego obszaru w pobliżu na osi współrzędnych w pobliżu punktu (0,0). Następnie szukanie najmniejszej wartości spośród sąsiadujących punktów o indeksach określonych jako: $(i, j) + \Delta$. gdzie $\Delta \in \{[1, 0], [0, 1], [1, 1]\}$. Ze względu na wspomniane ograniczenia w zastosowaniu podana metoda nie jest już chętnie stosowana w systemach rozpoznawania mówcy. Dodatkową wadą jest duża złożoność obliczeniowa, szczególnie w przypadku dużej ilości sekwencji do sprawdzenia. Jego skuteczność maleje wraz ze wzrostem długości sekwencji. Zaletą tego rozwiązania jest jednak prostota implementacji i w zastosowaniach z detekcją krótkich haseł metoda może okazać się być skuteczna.

### 1.7.2 Kwantyzacja wektorów - VQ.

Inną deterministyczną metodą (*ang. template model*) stosowaną w etapie dopasowania wzorca w problemach rozpoznawania mówcy jest kwantyzacja wektorów (*ang. vector*




Rysunek 1.8 Rezultat działania metody DTW na bardziej skomplikowanej sekwencji [7]. Na głównym wykresie widać trajektorię z zaznaczonym ograniczeniem dla ścieżki algorytmu.

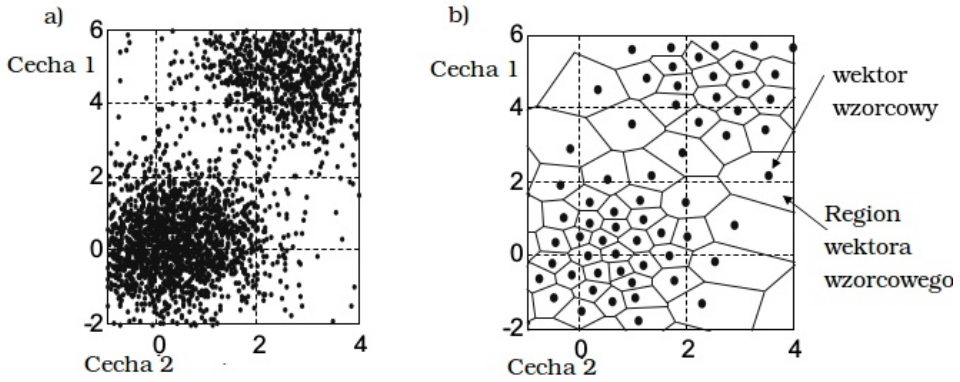
*quantization*). W odróżnieniu do wcześniej omawianej metody DTW, ta technika jest jest niezależna od kolejności analizowanych wektorów cech - tzn. jest niezmienny w czasie (*ang. time-invariant).* Z tego powodu możliwe jest zastosowanie tej techniki dla systemów rozpoznawania mówcy niezależnego od wypowiadanego tekstu (*ang. text-independent*). Technika kwantyzacji wektorów należy do klasy problemów z zakresu technik dopasowania wzorca jaką jest uczenie bez nadzoru (*ang. unsupervised learning*). Tego typu problem nazywany jest także [1] klasteryzacją (*ang. clustering*). Jest to zagadnienie analizy skupień to znaczy dokonywane jest grupowanie przestrzeni elementów na którą składa się większa liczba - w przypadku rozpoznawania mówcy - wektorów cech na mniejszą liczbę regionów. Regiony te reprezentowane są przez pewien ustalony wektor wzorcowy $\tilde{x}$. Zatem każdy punkt rozpatrywanej przestrzeni wektorów cech - $x$, należy do regionu reprezentowanego przez wzorcowy wektor, względem którego ów punkt leży najbliżej w rozumieniu wybranej, zdefiniowanej na tej przestrzeni metryki $d(x, \tilde{x})$. W fazie testowania, przeprowadzana jest analiza przynależności wektorów wejściowych do regionów reprezentujących weryfikowanego mówcę, to znaczy obliczana jest suma ich odległości do najbliższego wektora wzorcowego:

$$z = \sum_{j=1}^{L} \min_{\tilde{x} \in C} \{d(x_j, \tilde{x})\} . \tag{1.30}$$

Funkcja ta wyraża miarę dopasowania wektorów wejściowych do weryfikowanego modelu mówcy. W tej metodzie zbiór wektorów reprezentujących poszczególnych mówców nazywany jest książką kodową (*ang. codebook*), zaś każdy wektor wchodzący w jej skład nazywa się kodem lub hasłem (*ang. codeword*).

Przykładowy proces klasteryzacji w procesie uczenia się został przedstawiony na rysunku 1.7.2. Z puli 5000 wektorów akustycznych cech został stworzony model mówcy reprezentowany przez 64 wektory wzorcowe - tzw. *codebook*.

Metoda ta cechuje się tym, że nie przechowuje informacji związanej z sekwencją wypowiedzi, więc nie jest w stanie wychwycić pewnych cech związanych z konkretnym przej-




Rysunek 1.9 Klasteryzacja przestrzeni [15]. Na rysunku zostały przedstawione tylko 2 arbitralnie wybrane wymiary z wektora cech. Dane uczące składające się z 5000 wektorów zostały przekształcone na model 64 wektorów wzorcowych w procesie kwantyzacji wektorów.

ściem pomiędzy kolejnymi segmentami mowy. Z drugiej strony upraszcza w sposób znaczący implementację systemu.

W tej pracy metoda kwantyzacji wektorów została użyta w przykładowej implementacji systemu weryfikacji mówcy w postaci realizującego nią algorytmu LGB.

### Algorytm LGB.

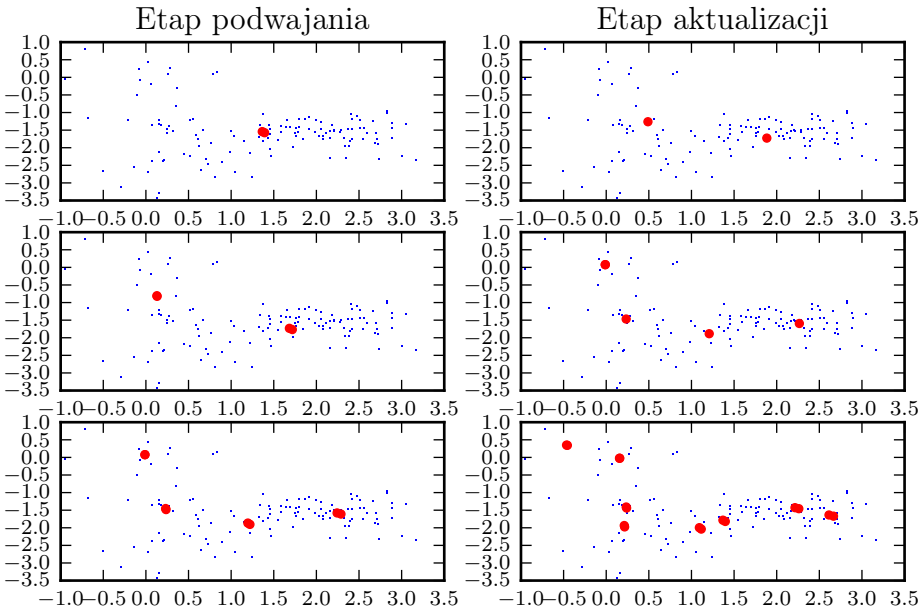
Algorytm Linde, Buzo i Greya - LGB [18] przedstawia sposób uzyskania wektorów wzorcowych (książki kodów - "codebook") reprezentujących mówcę, uzyskanego z danych uczących zawierających dużą liczbę wektorów cech wyekstrahowanych z wypowiedzi modelowanego mówcy. Jest to algorytm szeroko znany i chętnie wykorzystywany [3] w zadanie rozpoznawania mówcy. Zaczynając od jednego punktu, np. uśrednionego wektora cech (jak we wzorze 1.28) produkowane są kolejne centroidy poprzez podział. Ten algorytm więc produkuje książkę kodów o rozmiarze K gdzie K jest potęgą dwójki. W kolejnych etapach algorytmu dokonuje się detekcji najbliższej centroidy dla każdego wektora cech danych trenujących. Ze względu na nowo powstałe regiony aktualizuje się położenie centroid. Po każdej aktualizacji jest prawdopodobne, że kolejne wektory cech zmienią swój region klasteryzacji, a zatem proces powtarza się aż do ustabilizowania się tychże regionów. Następnie powtarza się podwajanie wektorów wzorcowych aż do uzyskania pożądanej ich ilości K. Poniżej podsumowano w sposób szczegółowo kolejne etapy algorytmu, na rysunku 1.7.2 przedstawiono schemat blokowy algorytmu, a na rysunkach 1.10 przedstawiono wizualnie kolejne etapy przetwarzania algorytmu dla dwóch arbitralnie wybranych wymiarów z wektorów cech.

#### Kolejne kroki algorytmu Linde, Buzo i Greya.[3]

1. Utwórz jedno-elementową książkę kodów - zawierającą uśrednioną centroidę jak we wzorze 1.28 dla wszystkich treningowych wektorów cech.

2. Podwój liczbę elementów w książce kodów. Zmodyfikuj ich poszczególne wartości ze względu na wybrany, mały czynnik $\epsilon$ w następujący sposób, dla każdego $\mathbf{k}$ gdzie $k \in \{1, ..., K\}$:

   - $\tilde{\boldsymbol{x}}_{\boldsymbol{k}} = (1 - \epsilon) \cdot \tilde{\boldsymbol{x}}_{\boldsymbol{k}}$,




Rysunek 1.10 Wizualizacja procesu podwajania i aktualizacji pozycji w algortymie LBZ.

- $\tilde{\boldsymbol{x}}_{\boldsymbol{k+1}} = (1+\epsilon) \cdot \tilde{\boldsymbol{x}}_{\boldsymbol{k}}$.

Co oznacza przesunięcie je w przeciwnych kierunkach (w N-wymiarowej przestrzeni).

3. Dokonaj klasyfikacji wszystkich wektorów trenujących - znajdź dla każdego najbliższą centroidę w rozumieniu przyjętej metryki $d(\tilde{\boldsymbol{x}}_{\boldsymbol{k}}, \boldsymbol{x})$.

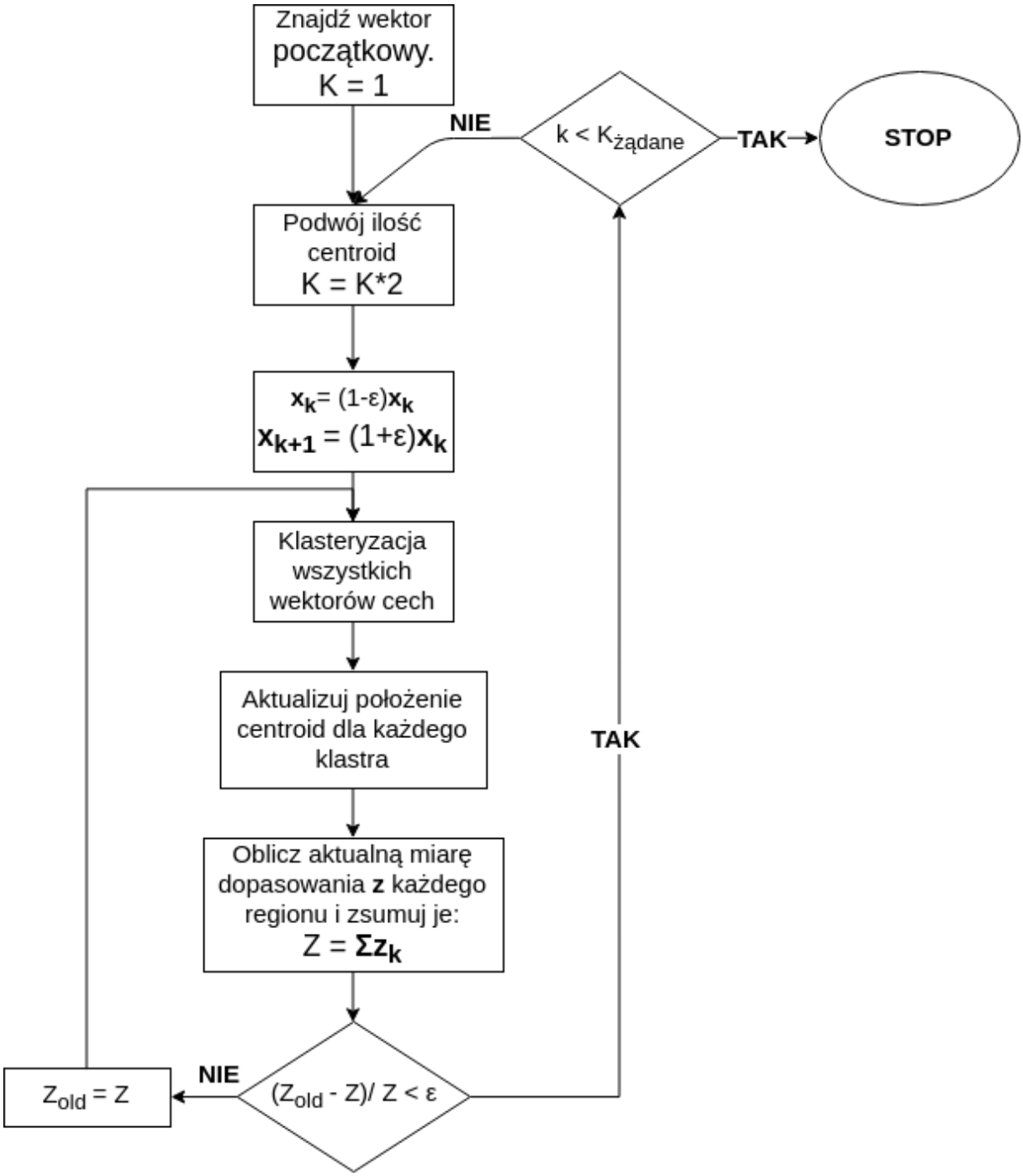
4. Zaktualizuj położenie każdej centroidy jako średniej 1.28 ze wszystkich wektorów cech sklasyfikowane jako leżące najbliżej.

5. Powtarzaj kroki 3 i 4 dopóki położenie nowych centroid nie ustabilizuje się (różnica przesunięcia dwóch ostatnich iteracji jest poniżej ustalonego progu).

6. Powtarzaj kroki 2, 3, 4 i 5 do momentu otrzymania słownika o żądanej długości **K**.

### 1.7.3 Mikstury Gaussowskie - GMM.

Techinka modelowanie miksturami Gaussowskimi *ang. Gaussian mixture modeling* jest metodą stochastyczną rozpoznawania wzorca. GMM jest sposobem modelowania uważanym za swego rodzaju standard w metodach rozpoznawania mówcy niezależnym od wypowiadanego tekstu. Tak jak wspomniano w rozważaniach ogólnych (1.7), w metodach probabilistycznych należy ustalić funkcję gęstości prawdopodobieństwa $p(X|\lambda)$ charakteryzującą danego mówcę. Dla problemu rozpoznawania mówcy niezależnego od tekstu, nie ma żadnej wiedzy apriori, którą można by było wykorzystać do założenia jakiegoś rozkładu. Dlatego też metoda GMM proponuje dokonania aproksymacji tzw. miksturami Gaussa (*ang. Gaussian mixtures*). Okazuje się, że taka reprezentacja funkcji gęstości prawdopodobieństwa jest równie skuteczna co dużo bardziej skomplikowana reprezentacja ukrytymi modelami Markova (HMM) dla problemu rozpoznawania bez znajomości tekstu [4], ze względu na brak przydatności wiedzy o sekwencji wektorów cech. Dla K-wymiarowych




Rysunek 1.11 Schemat blokowy algorytmu Linde, Buzo i Greya. [3]

wektorów cech funkcja gęstości prawdopodobieństwa dla modeli GMM wyraża się jako:

$$p(X|\lambda) = \sum_{i=1}^{M} w_i p_i(X). \tag{1.31}$$

gdzie: $w_i$ są współczynnikami wagowymi ustalanymi w fazie uczenia oraz $\sum_{i=1}^{M} w_i = 1$. M oznacza zaś liczbę wykorzystanych misktur Gaussowskich do aproksymacji funkcji gęstości prawdopodobieństwa dla modelu $\lambda$. Kolejna mikstura Gaussowska przybiera postać

wielowymiarowej funkcji rozkładu Gaussa:

$$p_i(X) = \frac{1}{(2\pi)^{K/2} \cdot |\boldsymbol{\Sigma_i}|^{\frac{1}{2}}} \cdot \exp\left\{-\frac{1}{2}(X - \boldsymbol{\mu_i})'(\boldsymbol{\Sigma_i})^{-1}(X - \boldsymbol{\mu_i})\right\} \tag{1.32}$$

gdzie: $\boldsymbol{\Sigma_i}$ oznacza macierz kowariancji wektora cech X, $\mu_i$ jest wektorem średnim (K-wymiarową wartością oczekiwaną). Pełny opis modelu $\lambda$ zapewnia zbiór parametrów $\lambda = \{w_i, \boldsymbol{\mu_i}, \boldsymbol{\Sigma_i}\}$. Przy pomocy wektorów cech danych trenujących dokonywana jest estymacja parametrów dla poszczególnych mikstur Gaussowskich. W tym celu najczęściej używa się się metod maksymalnego prawdopodobieństwa a posteriori MAP (*maximum a posteriori probability*) oraz algorytmu EM (*ang. expectation-maximization*).

### 1.7.4 Inne metody modelowania mówcy.

- **Sztuczne sieci neuronowe** (*ang. Artificial Neural Network - ANN*) - są metodą uczenia maszynowego wykorzystującą tzw. neurony będące strukturą matematyczną realizującą funkcję nieliniową o wielu wejściach i jednym wyjściu. Takie systemy mają zastosowanie w tworzeniu nowych metod zarówno w dziedzinie parametryzowania mowy (ekstrakcji cech) a także w dziedzinie modelowania mówcy. Tworzone są także systemy złożone z sieci neuronowych realizujące systemy rozpoznawania mówcy od początku do końca. Główną zaletą takich systemów z punktu widzenia wykonania zadania rozpoznawania w czasie rzeczywistym jest możliwość implementacji takiego algorytmu bezpośrednio na sprzęcie za pomocą układów FPGA.

- **Maszyny wektorów wspierających** (*ang. Support Vector Machine - SVM*) - są metodą uczenia maszynowego służącą do rozwiązania zadania uczenia z nadzorowaniem. Stosowana jest z sukcesem dla cech niskiego i wysokiego poziomu. Szczególnie chętnie jest stosowany w zadaniu weryfikacji mówcy. Dla tego problemu metoda SVM w fazie trenowania korzysta z oznaczonych wektorów dla mówcy modelowanego w celu konstrukcji hiperpłaszczyzny rozdzielającej wektory należące do mówcy oraz wektory tła. Problem znalezienia takiej hiperpłaszczyzny jest problemem optymalizacji funkcji decyzyjnej postaci:

$$f(\boldsymbol{X}) = \sum_{i=1}^{N} \alpha_i t_i K(\boldsymbol{X}, \boldsymbol{X_i}) + d \tag{1.33}$$

  gdzie $\boldsymbol{x_i}$ - wektory wspierające, $t_i \in \{-1, 1\}$ - oznaczenie przynależności wektorów, $\alpha$ waga wektora oraz funkcja jądra $K(\boldsymbol{X}, \boldsymbol{X_i})$ określająca odległość od hiperpłaszczyzny $g(X)$. W najprostszym przypadku, klasyfikatora maksymalnego marginesu - (który był zaproponowany w oryginalnej pracy Vapnika w 1979), maksymalizuje się odległość (margines) najbliższych wektorów $\boldsymbol{x_i}$ (czyli *wektorów wspierających* (*ang. support vectors*)) do wspomnianej hiperpłaszczyzny $g(X)$. W wyniku uzyskiwany jest model - granica oddzielająca mówcę od tła.

- **Fuzja** (*ang. fusion*) - dla problemu rozpoznawania mówcy zostały zastosowane systemy korzystające z kombinacji różnych typów wektorów cech sparametryzowanego sygnały mowy. Realizowane są również systemy korzystające z różnych modeli uzyskanych na podstawie tych samych wektorów cech. Decyzja na wyjściu takiego systemu jest podejmowana za pomocą funkcji uwzględniające wyniki wszystkich zastosowanych wewnętrznych systemów. Najlepsze rezultaty otrzymywane są gdy korzysta się z jak najbardziej niezależnych od siebie metod - tzn. uwydatniających

różne właściwości sygnały mowy. Widoczne jest to szczególnie przy fuzji cech wysokiego i niskiego poziomu.

## 1.8 Teoria decyzji

System weryfikacji mówcy uzupełnia zadanie rozpoznawania mówcy o etap weryfikacji - tzn. podjęcia decyzji o autoryzacji osoby podającej się za mówcę znajdującego się w bazie modeli systemu. Każdy system może popełnić dwa rodzaje błędów:

- **Błąd I rodzaju** - pozytywna weryfikacja fałszywego mówcy,
- **Błąd II rodzaju** - negatywna weryfikacja prawdziwego mówcy.

Intuicyjnie można powiedzieć, że błąd rodzaju I jest znacznie poważniejszy od błędu II rodzaju. Relacja częstotliwości tych błędów regulowana jest w większości metod za pomocą progu $\theta$. Pożądane jest zatem zwykle przyjmowanie wartości progu w taki sposób, aby minimalizować ten poważniejszy. Ze względów na wygodę i efektywność wymaga się także odpowiednio małego progu odrzuceń systemu. Z drugiej strony, tak jak opisano zgodnie z definicją błędu ERR (4.1), poziom tych błędów mogą być miarą jakości działania systemu weryfikacji. Główną metodą w omawianych systemach jest testowanie hipotez w ujęciu bayesowskim, co jest przedstawione dalej.

### 1.8.1 Testowanie hipotez w zadaniu weryfikacji mówcy.

Dla testowanego segmentu mowy $Y$ w zadaniu weryfikacji mówcy usiłuje się ustalić czy został on wyprodukowany przez mówcę $S$. Na potrzeby tego zadania można postawić dwie przeciwne wobec siebie hipotezy:

1. $\boldsymbol{H_0}$: sygnał mowy $Y$ pochodzi od mówcy $S$,
2. $\boldsymbol{H_1}$: sygnał mowy $Y$ nie pochodzi od mówcy $S$.

Powszechnie w dziedzinie weryfikacji mówcy stosowane jest kryterium stosunku modeli prawdopodobieństwa $\lambda_S$ (*ang. likelihood ratio*) sformułowane następująco:

$$\lambda_S = \frac{p(Y|H_0)}{p(Y|H_1)} \begin{cases} \geqslant \theta, \text{ przyjmij hipotezę } H_0 \\ < \theta, \text{ odrzuć hipotezę } H_0 \end{cases} \tag{1.34}$$

gdzie funkcje $p(Y|H_0)$ i $p(Y|H_1)$ oznaczają warunkowe funkcje gęstości prawdopodobieństwa dla sygnału Y, $\theta$ oznacza prób przyjęcia.

Mając jednak na uwadze klasycznie podejście od problemu - zawierające etapy ekstrakcji cech/parametryzacji sygnału mowy 1.6 oraz modelowania mówcy 1.7 - oczekujemy, że z sygnału $Y$ zostanie wyekstrahowany zbiór cech $X = \{x_1, ..., x_T\}$ dla kolejnych chwil czasu $t = \{1, ..., T\}$. Z tego powodu 1.34 można zapisać jako:

$$\lambda_S = \frac{p(X|\lambda_S)}{p(X|\lambda_{\tilde{S}})} \begin{cases} \geqslant \theta, \text{ przyjmij hipotezę } H_0 \\ < \theta, \text{ odrzuć hipotezę } H_0 \end{cases} \tag{1.35}$$

taka postać współczynnika uwarunkowana jest tym, że funkcja gęstości prawdopodobieństwa jest scharakteryzowana przez zbiór cech $X$ sparametryzowanej mowy $Y$, a hipoteza

$\boldsymbol{H_0}$ jest reprezentowana przez model mówcy $S$ - $\lambda_S$ oraz hipoteza $\boldsymbol{H_1}$ jest zdefiniowana przez model odniesienia $\lambda_{\tilde{S}}$.

W ostateczności jako kryterium decyzji używa się logarytmu wspomnianego czynnika $\log(\lambda_S)$ (*ang. log likeligood ratio*). Ostatecznie omawiane kryterium przyjmuje postać:

$$\Lambda(X) = \log(p(X|\lambda_S)) - \log(p(X|\lambda_{\tilde{S}}) \begin{cases} \geqslant \theta, \text{ przyjmij hipotezę } H_0 \\ < \theta, \text{ odrzuć hipotezę } H_0 \end{cases} \tag{1.36}$$

Model odniesienia $\lambda_{\tilde{S}}$ zazwyczaj przybiera jedną z dwóch form:

- Model tła (*ang. background model*) - używa się przekształcenia $\mathcal{F}(p_1(X|\lambda_1), ..., p_N(X|\lambda_N))$ do skonstruowania jednego modelu tła i co za tym idzie funkcji gęstości prawdopodobieństwa $p(X|\lambda_{\tilde{S}})$. Jednym z najbardziej popularnych modeli tego typu jest UBM (*ang. Universal Background Model*). Niewątpliwą korzyścią płynąca z zastosowania tej metody jest tylko jednokrotna operacja obliczenia modelu który stosowany jest dla wszystkich testów.

- Model kohorty (*ang. cohort*) - model odniesienia konstruowany jest za pomocą tylko najbliższych znanych modeli. Możliwy jest wybór $p(X|\lambda_{\tilde{S}})$ np. jako tego cechującego się największym prawdopodobieństwem dla realizacji X lub jako średni model towarzyszących modeli.

Podana metoda wykorzystywana jest z łatwością dla technik GMM i HMM modelowania. Jednak metody deterministyczne, w szczególności technika kwantyzacji wektorów (VQ) wymaga zastosowania dodatkowych, bardziej skomplikowanych technik z zakresu teorii decyzji aby wykorzystać opisywaną metodę.

### 1.8.2 Weryfikacja dla metod deterministycznych.

**Stały próg.**

System weryfikacji mówcy korzystający z podstawowej wersji techniki kwantyzacji wektorów (VQ), z racji tego, iż nie jest modelem probabilistycznym oraz ze względu na prostotę, korzysta zwykle ze stałego progu $\theta$. Metoda jest odpowiednikiem weryfikacji mówcy z jednym mówcą tła. Przyjęty próg nie posiada jednak wymiaru prawdopodobieństwa. W przypadku VQ próg $\theta$ przyjmuje wartość związaną z przyjętą miarą dopasowania $D_{avg}(X, X_\lambda)$. Przykładem takiego systemu jest system weryfikacji mówcy z wyświetlanym hasłem, opisany w [8], na którym to wzoruje się prezentowany w niniejszej pracy przykład systemu. Próg dla takiego przypadku często wyznacza się empirycznie.

**Próg dla modelu kohorty.**

Dla metod deterministycznych możliwe jest także wykorzystanie analogicznego warunku jak w 1.36, z tym, że zastępując warunkową funkcję gęstości prawdopodobieństwa przez funkcję miary dopasowania $D_{avg}(X_\lambda, X_{\tilde{\lambda}_{kohorta}})$, gdzie $X_{\tilde{\lambda}_{kohorta}}$ oznacza model uzyskany ze względu na mówców zawartych w kohorcie. Zatem podobnie jak poprzednio otrzymuje się:

$$d(X) = \log(D_{avg}(X, X_\lambda)) - \log(D(X, X_{\lambda_{kohorta}})) \begin{cases} \geqslant \theta, \text{ przyjmij hipotezę } H_0 \\ < \theta, \text{ odrzuć hipotezę } H_0 \end{cases} \tag{1.37}$$

Dla metody SVM (1.7.4) zaś kryterium to może przyjąć postać:

$$f(X) = \sum_{i\in\{i|t_i=1\}} \alpha_i K(\boldsymbol{X}, \boldsymbol{X_i}) - \sum_{i\in\{i|t_i=1\}} \alpha_i K(\boldsymbol{X}, \boldsymbol{X_i}) + d \begin{cases} \geqslant \theta, \text{ przyjmij hipotezę } H_0 \\ < \theta, \text{ odrzuć hipotezę } H_0 \end{cases} \tag{1.38}$$

### 1.8.3 Przycinanie mówców (*ang. speaker pruning*) .

*Speaker pruning* (tzn. 'przycinanie mówców') jest techniką wspomagającą zadanie identyfikacji mówcy w zastosowaniach czasu rzeczywistego. Możliwe jest też zastosowanie tej metody dla problemu weryfikacji mówcy korzystającego z modelu kohorty, który wymaga analizy więcej niż jednego modelu odniesienia w czasie rzeczywistym czyli tzw. (*ang. unconstrained cohort normalization*). Omawiana technika ta polega na odrzucaniu nieprawdopodobnych mówców już w trakcie nadchodzenia kolejnych wektorów cech, sparametryzowanego sygnału mowy. Najczęściej przyjmuje się pewien krok ilości nowych wektorów które będą wykorzystane do wstępnego odrzucenia kolejnych nieprawdopodobnych modeli. Przykładem algorytmu dla zastosowaniu w 1.37 jest algorytm szybkiego dobierania kohorty (*ang. Fast Cohort Scoring - FCS*) [5]. Algorytm korzysta z wcześniej opisanego algorytmu LGB 1.7.2, klasteryzując zbiór wektorów $X$ na mniejszy $\tilde{X} = LBG(X)$. Niech $\lambda$ oznacza zbiór centroid modelu. Dokonywany jest wybór $K$ najbliższych mówców ze względu na miarę podobieństwa $D_{avg}(\tilde{X}, \lambda)$. Zbiór wybranych modeli jest wspomnianą kohortą. W ostatnim kroku tego algorytmu oblicza się wartość kryterium d(X) z 1.37 jako:

$$d(X) = \frac{D_{avg}(\tilde{X}, \lambda)}{\frac{1}{K}\sum_{k=1}^{K} D_{avg}(\tilde{\tilde{X}}, \lambda_k)} \tag{1.39}$$

# Rozdział 2

# Projekt architektury oprogramowania

W tym rozdziale zostanie przedstawiona architektura oprogramowania systemu weryfikacji mówcy w czasie rzeczywistym w przeznaczeniu dla systemów wbudowanych. W pierwszej kolejności zostanie zaprezentowana ogólna struktura zadania weryfikacji mówcy odpowiednio dla fazy modelowania oraz fazy testowania na której to opiera się cały projekt. Następnie zostanie przedstawiony szczegółowy opis zaproponowanej architektury dla aplikacji algorytmów weryfikacji mówcy w czasie rzeczywistym. Dodatkowo zaprezentowana zostania ogólna implementacja rozwiązania problemu dla języka C++ - zostaną omówione struktury i algorytmy sterujące znajdujące się w systemie które są niezależne od wybranych metod z zakresu weryfikacji mówcy. Przykład konkretnego zastosowania omawianej architektury oraz opis implementacji w pełni funkcjonalnego sytemu weryfikacji mówcy znajduje się w kolejnym rozdziale.

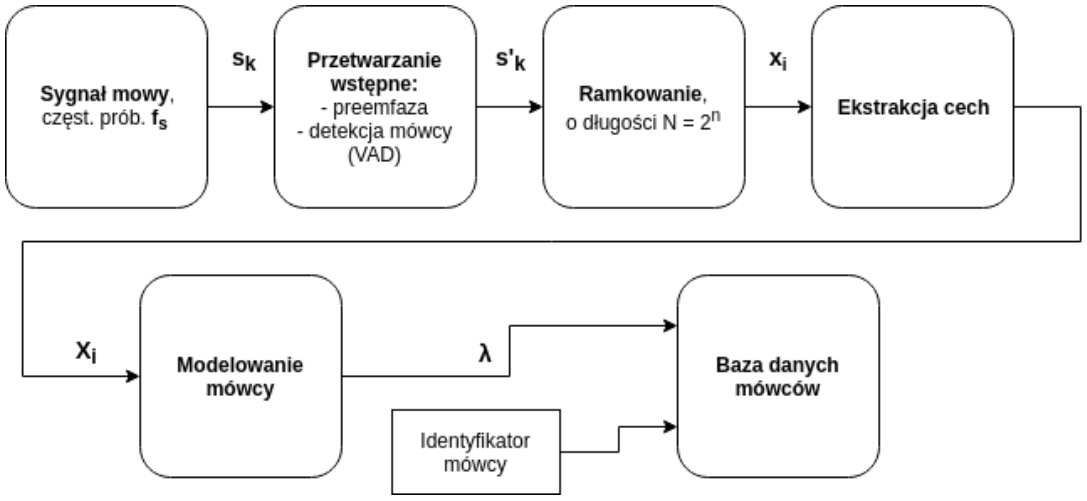
## 2.1 Schemat funkcjonalny systemu weryfikacji mówcy.

W ogólności system weryfikacji mówcy musi sprostać dwóm problemom składającym się na dwa etapy niezależne w czasie: uzyskania modelu mówcy w fazie modelowania (treningu) oraz podjęcie decyzji o weryfikacji mówcy na podstawie wcześniej uzyskanego modelu oraz testowanego sygnału mowy w fazie testowania (weryfikacji). Ponieważ te dwa etapy różnią się na poziomie funkcjonalności zatem struktura wewnętrzna systemów również w pewnym stopniu są różne.

### 2.1.1 Faza modelowania

Aby system mógł podjąć decyzję o weryfikacji mówcy w jego posiadaniu musi znajdować się wcześniej uzyskany model weryfikowanego mówcy, który wykorzystywany jest do porównania z odpowiednio sparametryzowanym, testowym sygnałem mowy. Odbywa się to w etapie modelowania. Ogólna struktura procesu uzyskiwania modelu mówcy z naciskiem na przepływ danych znajduje się na rysunku (**2.1**). Zadanie uzyskania modelu mówcy składa się kolejno z etapów:

1. **Akwizycja sygnału mowy** - wejściem całego sygnału jest spróbkowany z częstotliowścią $f_s$ sygnał mowy $\boldsymbol{s_k}$. Dostarczany jest on z bądź to z pamięci komputera jako całość lub dostarczany jest w czasie rzeczywistym jako kolejne próbki. Dla tego drugiego przypadku na ten etap składa się realizacja sprzętowa akwizycji sygnały akustycznego mowy za pomocą przetwornika elektroakustycznego oraz przetwornika analogowo-akustycznego. Ważnym elementem jest również odpowiednie filtrowanie




Rysunek 2.1 Schemat funkcjonalny systemu weryfikacji mówcy dla etapu modelowania.

analogowe sygnału - usunięcie składowej stałej oraz filtrowanie częstotliwości wyższych niż połowy częstotliwości próbkowania $f_s$ (filtracja antyaliasingowa).

2. **Przetwarzanie wstępne** - na ten etap mogą składać się wszelkiego rodzaju techniki poprawy jakości sygnału takie jak odszumianie. Najczęściej w systemach weryfikacji mówcy stosuje się tzw. preemfazę - czyli wzmacnianie wyższych częstotliwości sygnału oraz systemy detekcji mówcy (*ang. Voice Activity Detection - VAD*) zwłaszcza dla systemów czasu rzeczywistego. W wyniku przeprowadzonych operacji z sygnału $\boldsymbol{s_k}$ otrzymujemy przetworzony sygnał $\boldsymbol{s'_k}$.

3. **Ramkowanie** - zwykle w przypadku ekstrakcji cech niskiego poziomu konieczne jest zaaplikowanie ramek na sygnał mowy tak aby uzyskać chwilowe wektory cech. Dla metod ekstrakcji cech wykorzystujących dyskretne przekształcenie Fouriera (DFT) zwykle dobiera się ramki o długościach będących wielokrotnościami dwójki ($N = 2^n$). W przypadku takich cech jak LPCC nie jest to warunek konieczny, jednak często stosowany. W wyniku operacji ramkowania otrzymywany jest zbiór wektorów $\boldsymbol{x_i}$ o długości N.

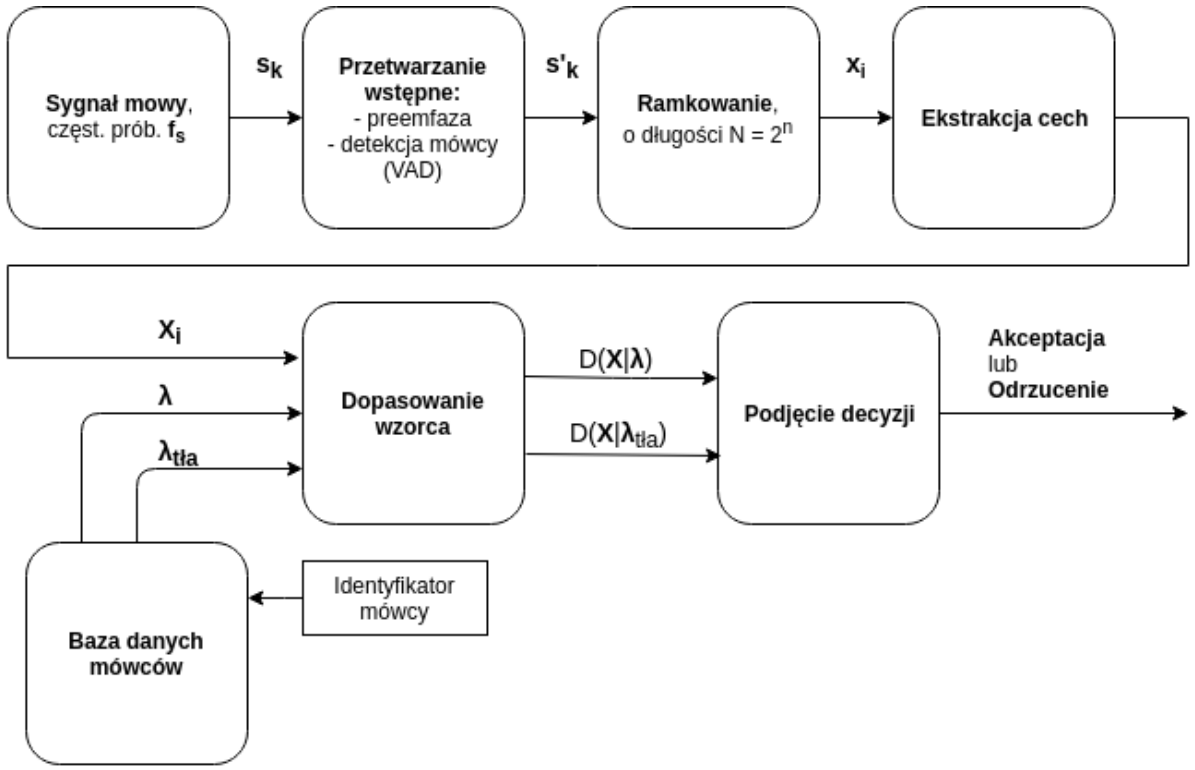
4. **Ekstrakcja cech** - na ten etap składa się przetwarzanie ramek sygnału mowy $\boldsymbol{x_i}$ na wektory cech oznaczone jako $\boldsymbol{X_i}$. Różne techniki wykorzystywane w tym procesie opisano w sekcji **1.6**.

5. **Modelowanie mówcy** - w tej części konstruowany jest model mówcy $\boldsymbol{\lambda}$ na podstawie zbioru wektorów trenujących $\boldsymbol{X_i}$. Różne techniki otrzymywania modeli opisane zostały w sekcji **1.7**. Model reprezentowany jest najczęściej jako wektor współczynników: w VQ są to centroidy znajdujące się w przestrzeni wektorów, zaś w technice GMM są tą współczynniki wagowe $w_i$ kolejnych funkcji aproksymujących funkcję gęstości prawdopodobieństwa.

6. **Zapisanie modelu do bazy danych** - Wartości parametrów charakteryzujących model mówcy $\boldsymbol{\lambda}$ muszą być zapisane na potrzeby fazy weryfikacji. Konieczne jest aby wraz z modelem została zapisana informacja dotycząca tożsamości mówcy - może być to numer identyfikacyjny w postaci danych charakterystycznych dla mówcy. Identyfikator mówcy jest wykorzystywany w fazie weryfikacji razem z modelem.

### 2.1.2 Faza weryfikacji.

Jeżeli system jest w posiadaniu modelu mówcy za który podaje się mówca od którego testowany sygnał mowy pochodzi, system weryfikacji mówcy może przeprowadzić procedurę weryfikacji. W tym celu musi przedstawić informację identyfikującą weryfikowanego mówcę. Ogólna struktura procesu weryfikacji mówcy z naciskiem na przepływ danych znajduje się na rysunku **2.2**. W omawianej procedurze zachowana jest postać poprzedniego algorytmu aż do momentu uzyskania zbioru wyekstrahowanych cech włącznie. Natomiast zachodzą istotne zmiany co do kolejnych etapów:



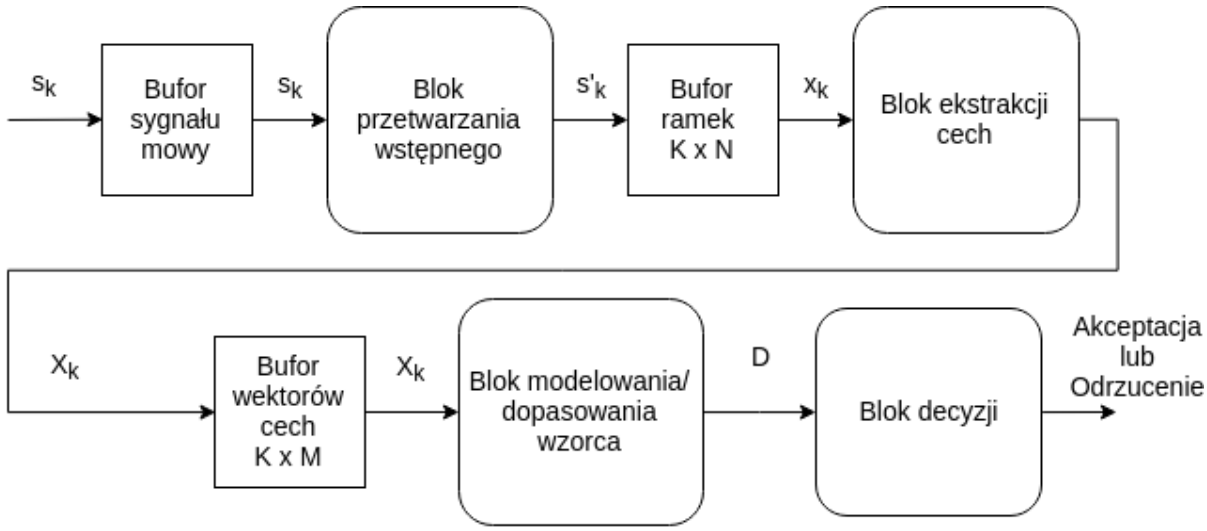

Rysunek 2.2 Schemat funkcjonalny systemu weryfikacji mówcy dla fazy weryfikacji.

1. **Dopasowanie wzorca** - ten etap korzysta ze zbioru cech $X_i$ w celu weryfikacji mówcy. Podczas tego etapu obliczana jest miara dopasowania pomiędzy wektorami $X_i$ a modelami kolejno $\lambda$ i $\lambda_{tla}$. W tym przypadku model oznaczony jako $\lambda_t a$ oznacza albo model tła albo model kohorty (w rozumieniu sekcji **1.8**). Wyjściem procesu jest wartość liczbowa tej miary dla obu porównań na rysunku oznaczone jako $D(\boldsymbol{X}|\boldsymbol{\lambda})$ i $D(\boldsymbol{X}|\boldsymbol{\lambda}_{tla})$. Dla techniki VQ miara dopasowania to wartość funkcji metryki $d(X, C_n)$ gdzie $C - n$ to centroidy modelu. W przypadku techniki GMM miarą dopasowania jest prawdopodobieństwo.

2. **Podjęcie decyzji** - ostatnim etapem razy weryfikacji mówcy jest podjęcie decyzji. Techniki związane z tym zagadnieniem opisano w sekcji **1.8**. Decyzja podejmowana jest na podstawie dwóch liczb dostarczonych z poprzedniego bloku: $D(\boldsymbol{X}|\boldsymbol{\lambda})$ i $D(\boldsymbol{X}|\boldsymbol{\lambda}_{tla})$ oraz ustalonego przez projektanta progu $\theta$. Wyjściem całego systemu jest zmienna binarna przyjmująca wartości: Akceptacja lub Odrzucenie.

## 2.2 Architektura systemu weryfikacji mówcy dla czasu rzeczywistego

Na podstawie przedstawionych schematów funkcjonalnych została zaproponowana architektura oprogramowania dla przetwarzania w czasie rzeczywistym. Ogólny schemat został przedstawiony na rysunku **2.3**. Rozwiązanie oparte jest na użyciu trzech bloków przetwarzania oraz trzech buforów typu fifo (*ang. first in firts out*). Projekt oparty jest na podejściu producent-konsument, tzn. występuje blok produkujący dane przekazywane do kolejki FIFO - nazwany producentem oraz blok odbierający dane z koljki FIFO - nazwany konsumentem. W omawianym rozwiązaniu zawsze występuje tylko jeden producent i jeden konsument. Takie bloki mogą zostać zaimplementowane w zwykłym systemie operacyjnym jako wątki, w systemie czasu rzeczywistego jako 'taski' oraz sekwencyjnie w przypadku braku takiego wsparcia ze strony systemu uruchomieniowego.




Rysunek 2.3 Schemat funkcjonalny architektury systemu weryfikacji dla przetwarzania w czasie rzeczywistym.

### 2.2.1 Bufory FIFO

Głównym zadaniem użytego w systemie buforu typu FIFO (kolejki) jest kontrola przepływu danych. Stan przepełnienia lub wyczerpania kolejki stanowi podstawę do sterowania kontekstem wykonywanego zadania. W systemie występują trzy kolejki przystosowane do różnego typu danych:

- **Bufor sygnału mowy** - jest to bufor FIFO przechowujący kolejne, nadchodzące próbki sygnału mowy. Może być zrealizowany programowo - dodatkowy wątek lub blok programu odpowiedzialny jest za jej napełnianie poprzez wywołanie odpowiednich wywołań systemowych lub za pomocą przerwania. Blok przetwarzani wstępnego jest w tym przypadku konsumentem i odbiera kolejne próbki z kolejki.

- **Bufor ramek** - jest to bufor FIFO przechowujący kolejne ramki sygnału o stałym rozmiarze $N$. Ramki produkowane są przez blok przetwarzania wstępnego i wysyłane do tejże struktury danych. Odbiorcą jest blok ekstrakcji cech.

- **Bufor wektorów cech** - jest podobny do buforu ramek, różni się jedynie mniejszym rozmiarem ramki dla wektora cech o długości $M$.

### Implementacja kolejki FIFO

W zależności od systemu uruchomieniowego kolejki mają postać:

- **System czasu rzeczywistego - RTOS** - w tego typu systemach znajdują się gotowe implementacje kolejek przystosowane do synchronizacji pomiędzy taskami. Przykładem takiej struktury jest *xQueue* w systemie operacyjnym *FreeRTOS* [17].

- **Wielowątkowy system operacyjny** - ponieważ w bibliotece standardowej nie znajdują się struktury danych umożliwiające bezpieczną pracę w środowisku wielowątkowym potrzebna jest inna implementacja kolejki niż standardowa *stdqueue*. Implementacja może pochodzić z zewnętrznej biblioteki takiej jak *boost* albo *QT*. Jednak łatwo jest zaimplementować omawianą strukturę danych przy pomocy algorytmu wzajemnego wykluczenia (*ang. mutex*) znajdującego się w bibliotece standardowej pod nazwą - *std::mutex*. Poniżej podano przykładową implementację z której korzysta się w przykładowym systemie opisanym w tej pracy.

Listing 2.1 bezpieczna wątkowo kolejka

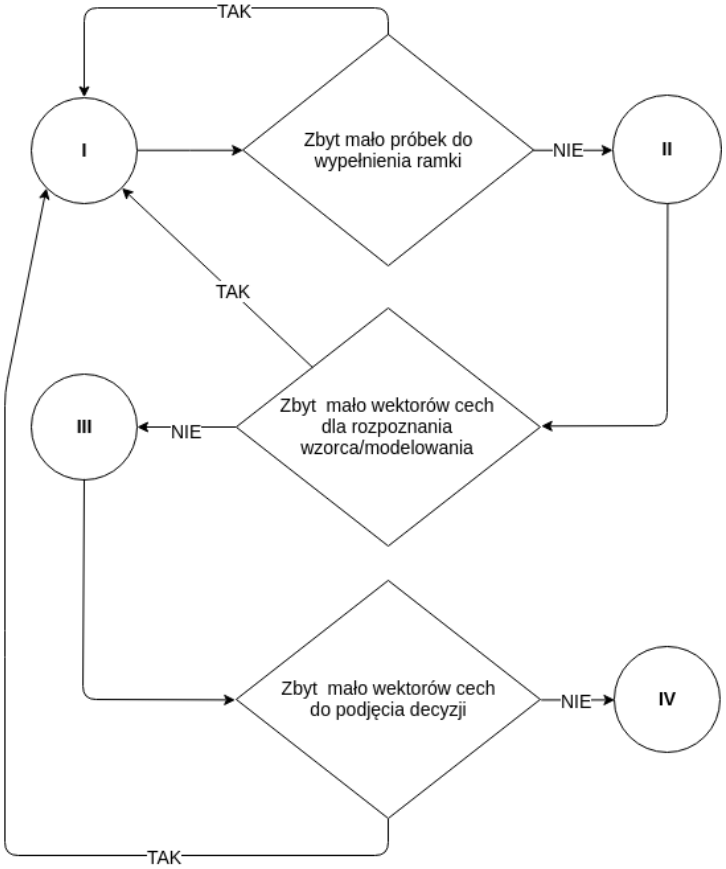
```
template <class T>
class queue
{
  public:
  ...
    T pop()
    {
      //zajecie mutexu
      std::unique_lock<std::mutex> mutex_lock(mutex_);
      while(queue_.empty()) // zablokuj watek jesli pusta
      {
        condition_.wait(mutex_lock);
      }
      auto item = queue_.front(); // pobierz element
      queue_.pop(); //usun obiekt z kolejki
      return item;
    }

    void push(T&& object)
    {
      //zajecie mutexu
      std::unique_lock<std::mutex> mutex_lock(mutex_);
      queue_.push(std::move(item)); // dodaj obiekt na koniec kolejki
      mutex_lock.unlock(); // zwolnij mutex
      condition_.notify_one(); // zaktualizuj zmienna condition_
    }
  ...

  private:
    std::queue<T> queue_;
    mutable std::mutex mutex_;
    std::condition_variable condition_;
};
```

Dla podanej implementacji zachowane semantykę wezwań tak jak w przypadku kolejki *std::queue*: funkcja *pop()* zwraca pierwszy element kolejki, zaś funkcja *push()* umieszcza obiekt na końcu kolejki. Aby zapobiec zjawisku wyścigu użyta została blokada *std::mutex* podczas odczytu i zapisu z kolejki. Z kolei obiekt *std::condition_variable* z biblioteki standardowej realizuje blokadę wątku odczytującego w przypadku kiedy kolejka jest pusta. Wywołanie w linii 25 informuje zablokowany wątek o uzupełnieniu kolejki o jeden element. Pętla w linii 10 zapobiega zwolnieniu wątku w przypadku gdy wystąpi inny powód odblokowania wątku niż powiększenie się kolejki.

- **Brak systemu uruchomieniowego** - w przypadku braku środowiska wielowątkowego nie potrzebne jest zapewnienie kontenera bezpiecznego wątkowo (*ang. thread safe*). Dlatego wystarczy wykorzystać gotową strukturę kolejki z biblioteki standardowej *std::queue*.

### Kolejność wykonywania bloków

W przypadku korzystania ze wsparcia systemu operacyjnego sprawa jest najprostsza - kolejne bloki implementowane są jako niezależne wątki synchronizowane za pomocą kolejek. Jeżeli jednak system nie korzysta ze wsparcia środowiska uruchomieniowego potrzebna jest implementacja odpowiedniej logiki sterującej. Na rysunku **2.4** przedstawiono propozycję algorytmu sterowania wykonaniem w tego typu systemie.




Rysunek 2.4 Algorytm sterownia bez wsparcia systemu operacyjnego. **Blok I** - blok wstępnego przetwarzania. **Blok II** - blok ekstrakcji cech. **Blok III** - blok rozpoznania wzorca/modelowania. **Blok IV** - blok podejmowania decyzji.

### 2.2.2 Bloki przetwarzania

Poniżej opisano trzy bloki przetwarzania znajdujące się w systemie. Bloki realizują funkcjonalności które już zostały opisane na początku tego rozdziału (**2.1**).

- **Blok wstępnego przetwarzania** - realizuje funkcjonalności przetwarzania wstępnego oraz ramkowania z rysunku **2.2**. Wszystkie wykorzystywane funkcje w tym bloku zdefiniowane są w przestrzeni nazw *dsp_utils*.
- **Blok ekstrakcji cech** - realizuje funkcjonalność ekstrakcji cech.
- **Blok rozpoznawania wzorca/modelowania** - realizuje funkcjonalność dopasowania wzorca oraz modelowania.
- **Blok podejmownia decyzji** - realizuje funkcjonalność podejmowania decyzji o weryfikacji.

### 2.2.3 Baza danych mówców

Dany mówca reprezentowana jest jako następująca struktura danych dla systemu weryfikacji mówcy z wyświetlanym hasłem:

Listing 2.2 struktura danych przechowująca informację o mówcy

```
template <int C, int K>
using codebook_array = typename std::array<std::array<double, K>, C>;


class speaker {
  public:
    speaker(const std::string& name)
      : speaker_info{name} {}

    std::string name() {return speaker_info.name;};

    struct Code{
      Code(const std::string& text, codebook_array<C,K>&& code)
        : centroids{code}, text{text} {}
      std::array<std::array<double, K>, C> centroids;
      std::string text;
    };

    void add_code(speaker::Code&& code)
    {
      codebook.push_back(code);
    }

  private:
    std::vector<Code> codebook;

    struct speaker_info{
      speaker_info(const std::string& name)
        : name{name} {}
      std::string name;
    } speaker_info;
};
```

Klasa *speaker* zawiera dwie struktury danych: instancję struktury *speaker_ info* która reprezentuje numer identyfikacyjny mówcy przy pomocy zmiennej łańcuchowej *name* oraz instancji struktury *codebook* przechowującej wektory współczynników charakteryzujących modele (w przypadku metody VQ są to centroidy - stąd nazwa zmiennej *centroids*). Alias do tablicy dwuwymiarowej z której korzysta implementacja to *codebook_ array*.

# Rozdział 3

# System weryfikacji mówcy

W tym rozdziale zostanie opisana implementacja pełnego systemu weryfikacji mówcy.

## 3.1 Platforma uruchomieniowa

Weryfikacja poprawności działania systemu i wstępna implementacja zostały wykonane przy pomocy środowiska obliczeniowego Matlab. Rozwijanie oprogramowania w języku C++ na bazie narzędzi GNU odbywało się na systemie operacyjnym linux w dystrybucji debianowej. Docelową platformą uruchomieniową był komputer jednopłytkowy również z systemem takiego typu.

## 3.2 Wybór technologii

Omawiany system wykonuje parametryzacji sygnału mowy przy pomocy techniki Cepstralnych Współczynników Melowych (MFCC) w sekcji **1.6.2**. Rozpoznanie wzorca jest realizowane przy pomocy techniki kwantyzacji wektorów (VQ) (**sekcja 1.7.2**), zaś proces modelowania mówcy dodatkowo korzysta z algorytmu LGB (**sekcja 1.7.2**).

### 3.2.1 Sposób implementacji algorytmów

Do implementacji algorytmów skorzystano ze znajdującej się w standardzie C++ bibliotek STL (Standard Template Library). W omawianej implementacji wykorzystywane są głównie trzy funkcje pochodzące z biblioteki STL są to:

- **std::generate(container_iter start, container_iter end, object_func lambda)** - funkcja generuje wartość dla każdego elementu pomiędzy iteratorem *start* a iteratorem *end* w jednym kontenerze za pomocą obiektu funkcyjnego *lambda*.
- **std::transform(container1_iter start1, container1_iter end1, container2_iter start2, container3_iter start3, object_func lambda)** - funkcja generuje wartości w kontenerze o początku w *start3* za pomocą obiektu funkcyjnego *lambda* przyjmującego za argumenty kolejno wartości od start1 oraz od start2 aż do iteratora end1 i odpowiedniego w kontenerze nr. 2.
- **std::for_each(container_iter start, container_iter end, object_func lambda)** - produkuje wartości w kontenerze pomiędzy iteratorami start i end za pomocą obiektu funkcyjnego *lambda* przyjmującego za argument wartości z tego kontenera. Funkcja może być zastąpiona pętlą zakresową: **for(auto &&x: container)**.

## 3.3 Implementacja algorytmów wstępnego przetwarzania sygnału mowy

W tym miejscu została opisana implementacja bloku wstępnego przetwarzania opisanego w **sekcji 2.2.2**.

### Ramkowanie

Głównym zadaniem bloku wstępnego przetwarzania w implementowanym systemie jest ramkowanie sygnału mowy. Realizowane jest to za pomocą funkcji *frame_signal*. Funkcja ta przyjmuje jako argument kontener zawierający próbki sygnału *probes* oraz krok przesunięcia kolejnych ramek M. W poniżej implementacji użyta została funkcja *std::copy* umieszczająca w wynikowym wektorze kolejne ramki sygnału.

Listing 3.1 Implementacja funkcji *frame_signal*

```
1  template <int N>
2  std::vector<std::array<double, N>> frame_signal(const std::vector<double> probes, int
3  {
4    std::vector<std::array<double, N>> speech_frames;
5    for(int signal_offset = 0; signal_offset + N < static_cast<int>(probes.size()); sign
6    {
7      auto frame = std::array<double, N>();
8      auto start_pos = probes.cbegin() + signal_offset;
9      std::copy(start_pos, start_pos + N, frame.begin());
10     speech_frames.push_back(std::move(frame));
11   }
12   return speech_frames;
13 }
```

## 3.4 Implementacja ekstrakcji cech MFCC

W tym miejscu została opisana implementacja bloku ekstrakcji cech opisanego w **sekcji 2.2.2**.

Aby utrzymać wektory sparametryzowanego sygnały mowy w postaci współczynników MFCC należy wywołać funkcję *mfcc_extraction* przyjmującą za argumenty częstotliwość próbkowania *samplerate* oraz wektor ramek o długości N z próbkami sygnału mowy *speech_frames*. Funkcja zwraca wektor tablic długości K cech MFCC:

Listing 3.2 Implementacja funkcji mfcc_extraction

```
1
2  std::vector<std::array<double, K>>
3  mfcc_extraction( std::vector<std::array<double, N>>&& speech_frames,
4                   int samplerate)
5  {
6    (...)
```

### Obliczanie widma mocy dla ramek oraz nałożenie filtrów mela

Tak jak opisano w sekcji(**1.6.2**) pierwszym etapem ekstrakcji cech w tym przypadku jest obliczenie widma gęstości mocy. Kolejno w pętli wykonuje się operacje nakładania widma (za pomocą obiektu funkcyjnego *hamming_generator*), obliczenia współczynników

mocy DTF za pomocą algorytmu FFT (funkcja *power_power_frame*). Kolejnym etapem jest nakładanie filtrów mela za pomocą funkcji *mel_frame* :

Listing 3.3 funkcja *mfcc_extraction* - obliczanie widma gęstości mocy

```
(...)
std::vector<std::array<double, K>> mel_coefs_speech_frames;
for(auto &frame: speech_frames)
{
  dsp_utils::window_frame( frame,
                           dsp_utils::Window_type::hamming_generator);
  dsp_utils::power_fft_frame(frame);
  mel_coefs_speech_frames.push_back(mel_frame(frame, samplerate));
}
(...)
```

### Nałożenie logarytmu

W wektorze *mel_coefs_speech_frames* znajdują się na tym etapie funkcje gęstości widma. Kolejnym krokiem jest nałożenie logarytmu dziesiętnego na każdą ramkę. Realizowane jest to za pomocą obiektu funkcyjnego - lambdy:

Listing 3.4 funkcja *mfcc_extraction* - obliczanie logarytmu widma gęstości mocy

```
(...)
for(auto &mel_frame: mel_coefs_speech_frames)
{
  std::for_each( mel_frame.begin(),
                 mel_frame.end(),
                 [](double &val)
                 {
                   val = std::log10(val); }
                 );
}
(...)
```

### Oblicznie cepstrum

Kolejnym krokiem jest obliczenie cepstrum za pomocy dyskretnej transformaty kosinusowej (DCT). Aby zmniejszyć złożoność obliczeniową najpierw tablicowana jest funkcja kosinus. Dużym usprawnieniem jest jej deklaracja jako *const* tak aby tą operację wykonać już w czasie kompilacji. Stablicowana funkcja znajduje się w tablicy *cos_table*. DCT realizowana jest za pomocą obiektu funkcyjnego *dct_frame*.

Listing 3.5 funkcja *mfcc_extraction* - obliczanie cepstrum

```
(...)
std::array<double, 4*K> cos_table;
std::generate( cos_table.begin(),
               cos_table.end(),
               dsp_utils::cos_dct_gen(4*K));

std::vector<std::array<double, K>> mfcc;
for(const auto &mel_frame: mel_coefs_speech_frames)
{
  mfcc.push_back(dsp_utils::dct_frame(mel_frame, cos_table));
}
```

```
12      return mfcc;
13    }
```

W wyniku wykonanych operacji otrzymany zostaje wektor tablic ze współczynnikami MFCC.

### 3.4.1 Funkcje i obiekty funkcyjne realizujące algorytmy

W podrozdziale zostały opisane kolejne zaimplementowane w systemie funkcje i obiekty funkcyjne realizujące algorytmy dla zadania obliczenia współczynników MFCC użytych wcześniej w funkcji *mfcc_extraction*.

- **Funkcja window_frame** aplikująca okno czasowe na ramkę:

```
1 void window_frame( std::array<double, N>& fr,
2                    const Window_type& win_type);
```

- przyjmuje jako argument ramkę sygnału *fr* o rozmiarze N, oraz typ enumerowany *win_type* określający typ zastosowanego okna. W systemie zastosowano okno Hamminga. Aplikacja okna wykonana jest za pomocą funkcji *std::for_each* z użyciem obiektu funkcyjnego **hamming_generator** z określonym operatorem *operator()* jako:

```
1 void hamming_generator::operator()(double& x)
2 {
3   x = x*(0.54 - 0.46*gsl_sf_cos(2.0*M_PI*(index++)/(size -1)));
4 }
```

- **Funkcja power_fft_frame** obliczająca widmową gęstość mocy dla ramki:

```
1   void power_fft_frame(std::array<double, N>& fr);
```

- przyjmuje za argument referencję do pojedynczej ramki sygnału *fr* i modyfikuje ją do postaci współczynników widmowej gęstości mocy.

Najpierw użyty zostaje algortym fft na modyfikowanej ramce:

```
1     gsl_fft_real_radix2_transform(fr.data(), 1, N);
```

Następnie każdy współczynnik DFT zostaje podniesiony do kwadratu.

```
1     std::for_each(fr.begin(),
2                   fr.end(),
3                   [](double &x)
4                   {
5                     x = gsl_pow_2(x);
6                   });
```

A następnie zostają zsumowane wartości rzeczywiste z urojonymi dla kolejnych próbek DFT. Należy zwrócić uwagę, że w wyniku użycia funkcji *gsl_fft_real_radix2_transform* otrzymano tylko połowę widma z wartościami rzeczywistymi rosnącymi kolejno od 0 do N/2 i wartościami urojonymi z malejącymi indeksami od końca wektora fr. Z tego powodu użyty został iterator rekursywny *crbegin*:

```
1     std::transform(fr.cbegin() +1, fr.cbegin() + (N/2),
2                    fr.crbegin(),
3                    fr.begin() +1,
4                    std::plus<double>());
```

W wyniku tych operacji otrzymywany jest wektor współczynników amplitudowego widma mocy.

- **Funkcja mel_frame** aplikująca filtry melowe:

```
std::array<double, K> mel_frame( const std::array<double, N>& fr,
                                 int samplerate);
```

- przyjmuje za argument częstotliwość próbkowania *samplerate* oraz ramkę ze współczynnikami reprezentującymi widmową gęstość mocy o długości N. Funkcja zwraca wektor cech o długości K. Dla stworzenia wektora cech odpowiada funkcja *std::generate* oraz obiekt funkcyjny *mel_frame_generator*:

```
(...)
std::array<double, K> mel_frame;
std::generate(mel_frame.begin(),
              mel_frame.end(),
              mel_frame_generator<K, N>(fr, samplerate));
(...)

```

Operator *operator()* tego obiektu funkcyjnego zdefiniowany jest następująco:

```
double operator()(){

```

Obiekt *mel_frame_generator* oblicza i przechowuje wartości indeksów dla początku filtra (*f_begin*, środku filtra (*f_center*) oraz końcu filtra (*f_end*). Do obliczenia wartości szukanego współczynnika wystarczy użyć niezerowych wartości filtru tzn. tylko od indeksu f_begin do f_end. Ten zakres jest użyty w dla funkcji *for_each* w listingu poniżej. Jako funktor użyty jest kolejny obiekt funkcyjny *triangle_windowed_sum* inicjowany informacją o ilości próbek dla zbocza rosnącego (f_center - f_begin) oraz ilość próbek dla zbocza opadającego (f_end - f_begin) używanego filtru trójkątnego. Dla każdego kolejnego współczynnika melowego indeksy położenia filtru trójkątnego są aktualizowane prywatną funkcją *update_filter_samples*. Opisana część funkcji operatora to:

```
  auto mel_coef = std::for_each(fram.cbegin() + f_begin,
                  fram.cbegin() + f_end,
                  triangle_windowed_sum(f_center-f_begin,
                                        f_end-f_begin));
  update_filter_samples();
  return mel_coef.acc;
};

```

Mnożenie skalarne filtra przez ramkę realizowane jest przez wspomniany obiekt *triangle_windowed_sum*. Kolejne próbki (*val*) są przemnażane za pomocą jego przeciążonego operatora *operator()*. Co krok aktualizowana jest wartość filtra - zmienna *win*. Gdy algorytm dojdzie do indeksu środkowego filtru (warunek $cen == i$) zmienia się przyrost wartości filtra na ujemny, co pozwala na zachowanie takiego samego kodu dla obu zboczy filtru. Listing omawianej funkcji znajduje się poniżej:

```
void operator()(const double& val){
  if(cen == i) del = -1.0/static_cast<float>((end-cen));

```

```
4         acc += win*val;
5         win += del;
6         i++;
7       }
8
```

- **Obiekt funkcyjny cos_dct_gen** generujący funkcję kosinus:

```
1   struct cos_dct_gen
2   {
3     cos_dct_gen(int _K): K{_K}, i{0}
4     {}
5
6     double operator()(){
7       double c = gsl_sf_cos(2*M_PI*static_cast<double>(i)/K);
8       ++i;
9       return c;
10    }
11    int K, i;
12  };
```

- korzystając z bibliotecznej funkcji *gsl_sf_cos* generowana jest jeden okres funkcji kosinus dla K próbek. Tak wygenerowana tablica jest wykorzystywana dalej w algorytmie obliczania DCT. Warto zauważyć, że ze względu na definicję **1.16** potrzebna jest 4 razy większa tablica.

- **Funkcja dct_frame** obliczająca dyskretną transformatę kosinusową:

```
1   std::array<double, K>
2   dct_frame(const std::array<double, K>& mel_frame,
3             const std::array<double, K*4>& cos_table);
```

- przyjmuje jako argument tablicę współczynników melowych*mel_frame* o długości K oraz stablicowaną funkcję kosinus o długości 4*K. Funkcja zwraca tablicę współczynników MFCC.

Funkcja oblicza współczynniki DCT przy pomocy funkcji *std::generate* oraz obiektu funkcyjnego *mfcc_gen* który generuje kolejne współczynniki DCT. Obiekt inicjalizowany jest referencjami do ramki cech i tablicy okresu funkcji kosinus:

```
1   (...)
2    std::array<double, K> mfcc_frame;
3    std::generate(mfcc_frame.begin(),
4                  mfcc_frame.end(),
5                  mfcc_gen<K>(mel_frame, cos_table));
6    (...)
```

Obiekt generuje współczynniki DCT za pomocą przeciążonego operatora *operator()* zgodnie z definicją z rozdziału **1.16**:

```
1       double operator()(){
2         double mfcc;
3         auto k = 0;
4         {// for k=0
5           mfcc = (std::sqrt(K/2.0))*(1.0/K)*mel[k];
6         }
7         for(k=1; k<K; ++k)//exclude mean val
8         {
```

```
            mfcc+=(std::sqrt(K/2.0))*(2.0/K)*mel[k]*cos[(k*(2*n+1))%(4*K)];
          }
          ++n;
          return mfcc;
        };

```

## 3.5 Implementacja rozpoznania wzorca VQ

W tym miejscu została opisana implementacja bloku rozpoznawania wzorca opisanego w **sekcji 2.2.2**.

### 3.5.1 Miara dopasowania modelu do zbioru wektorów

W celu otrzymania wartości miary dopasowania modelu złożnego z C centroid o wymiarze K w przestrzeni wektorów cech należy wywołać funkcję *compute_distortion*:

Listing 3.6 Implementacja funkcji *compute_distortion*

```

    template<int C, int K>
    double
    compute_distortion(const std::array<std::array<double, K>, C>& code,
                       const std::vector<std::array<double, K>>&
                                                          acoustic_vectors)
{
   (...)
```

Funkcja ta zwraca wartość typu double reprezentującą miarę dopasowania do badanego modelu. Jej argumentami jest książka kodów *code* reprezentująca model mówcy za pomocą C centroid K wymiarowych. Drugim argumentem jest zbiór wektorów cech o tym samym wymiarze K.

Implementacja tej funkcji składa się na dwie główne pętle for: pierwsza iteruje po wszystkich wektorach cech *acoustic_vectors* oraz sumuje wartość odchylenia od modelu dla każdego wektora do zwracanej zmiennej *distortion*. Druga zaś pętla for - wewnętrzna - wyszukuje najbliższej centroidy w celu obliczenia miary dopasowania *distortion*. Opisywana reszta implementacji znajduje się tutaj:

Listing 3.7 Ciało funkcji *compute_distortion*

```
    (...)
    double distortion = 0;
    for(const auto& acc_vec: acoustic_vectors)
    {
      auto acc_vec_dist = std::numeric_limits<double>::max();
      for(const auto& centroid: code)
      {
        auto distance = vq::dis_eu<K>(centroid, acc_vec);
        if(distance < acc_vec_dist)
          acc_vec_dist = distance;
      }
      distortion += acc_vec_dist;
    }
    return distortion;
  }
```

W powyższym kodzie używa się funkcji *dis_eu* obliczającej odległość euklidesową pomiędzy dwoma wektorami - *v1* i *v2*. Zaimplementowano funkcję zgodnie z definicją odległości euklidesowej:

Listing 3.8 Funkcja *dis_eu*

```
template<int K>
double dis_eu(const std::array<double, K>& v1, const std::array<double, K>& v2)
{
  double dist =0;

  auto i1 = v1.cbegin();
  auto i2 = v2.cbegin();
  for(;i1 != v1.cend(); ++i1, ++i2)
  {
    dist += std::pow(*i1-*i2, 2);
  }

  return std::sqrt(dist);
}
```

### 3.5.2 Modelowania przy pomocy algorytmu LBG

W celu otrzymania współczynników modelu mówcy - centroid wyprodukowanych przez algorytm LBG (**sekcja 1.7.2**) na podstawie zbioru treningowych wektorów cech należy wywołać funkcję *lbg*:

Listing 3.9 Deklaracja funkcji *lbg*

```
template<int C, int K>
std::array<std::array<double, K>, C>
lbg(const std::vector<std::array<double, K>>& acoustic_vectors);
```

W wyniku wywołania funkcji zwracana jest 'książka kodów' o rozmiarze C. Wejściem jest zbiór wektorów cech o długościach K. Argumentem funkcji są wektory cech o nazwie referencji *acoustic_vectors*.

Implementacja realizuje algorytm znajdujący się na schemacie blokowym(**1.7.2**). Poniżej zostało przedstawione ciało funkcji z opisanymi w komentarzach nazwami kroków algorytmów. Następnie zostanie szczegółowo opisana implementacja każdego kroku z osobna.

Omawiana funkcja składa się z części inicjalizacyjnej oraz dwóch głównych pętli: zewnętrznej pętli *for* która za każdą iteracją podwaja ilość centroid aż do uzyskania wielkości C długości książki kodów. Wewnętrzna pętla *while* ma za zadanie powtarzać kroki algorytmu LBG aktualizujące położenie centroid do momentu niewielkiej zmiany. Sygnalizowane jest to zmienną *distortion_diff_small_enough*. Stała *eps* jest współczynnikiem przesunięcia położeń dwóch wygenerowanych centroid względem centroidy z której zostały stworzone. Jednocześnie jest też współczynnikiem określającym krok poniżej którego algorytm nie poprawia położeń centroid. Kontenery zadeklarowane w listingu(**3.10**) przechowują inormację o:

- **owner_ind** - jest kontenerem przechowującym numer najbliższej centroidy dla każdego wektora cech z *acoustic_vectors*.
- **owner_dist** - jest kontenerem przechowującym odpowiadającą odległość do najbliższej centroidy dla każdego wektora cech.

- **owner_ind_count** - jest kontenerem przechowującym informację o ilości wektorów cech leżących najbliżej dla każdej centroidy.

- **owner_distortion** - jest kontenerem przechowującym sumę miar dopasowania wektorów cech przynależących kolejno do każdej centroidy.

Listing 3.10 Ciało funkcji *lbg*

```
template<int c, int k>
std::array<std::array<double, k>, c>
lbg(const std::vector<std::array<double, k>>& acoustic_vectors);
{
  std::array<std::array<double, k>, c> codebook;

  /* inicjalizacja - obliczenie polozenia pierwszej centroidy */

  (...)

  const double eps = 0.01;

  std::vector<int> owner_ind(acoustic_vectors.size(), 0);
  std::vector<double> owner_dist(acoustic_vectors.size(), 0);
  std::vector<int> owner_ind_count(c, 0);
  std::vector<double> owner_distortion(c, std::numeric_limits<double>::max());

  for(auto code_size= 1; code_size < c;)
  {
    /* podwojenie ilosci centroid w ksiazce kodow */

    (...)


    auto distortion_diff_small_enough = false;
    while(!distortion_diff_small_enough)
    {

      /* znalezienie najblizszych centroid dla kazdego wektora cech */

      (...)

      /* aktualizacja polozenia centroid ze wzgledu na przynaleznosc wektrorow cech */

      (...)

      /* obliczenie sumy miary dopasowania dla kazdego wektora cech. */

      (...)

      /* decyzja o kontynuowaniu dostosownia centroid do wektorow cech */

      (...)
    }
  return codebook;
}
```

### Inicjalizacja - pierwsza centroida.

W tej części funkcji obliczany jest wektor średni - centroida dla wszystkich wektorów cech znajdujących się w *acoustic_vectors*. Wektor sumy poszczególnych wymiarów jest utworzony jako pierwsza centroida w książce kodów *code* za pomocą funkcji *std::transform*, a następnie w drugiej pętli for liczona jest średnia poprzez podzielenie sum przez ilość wektorów cech.

Listing 3.11 Funkcja *lbg* - obliczenie pierwszej centroidy

```
1     std::fill(code.at(0).begin(), code.at(0).end(), 0);
2     for(auto&& a_vec: acoustic_vectors)
3     {
4       std::transform(a_vec.cbegin(), a_vec.cend(),
5                      code.at(0).cbegin(), code.at(0).begin(),
6                      [](double x, double acc){
7                        return x+acc;
8                      });
9     }
10
11    for(auto&& x: code.at(0))
12    {
13      x /= acoustic_vectors.size();
14    }
```

### Podwojenie ilości centroid.

Poniższy fragment kodu kopiuje każdą centroidę modyfikując każdy wymiar o współczynnik *1+esp* oraz dodaje ją do książki kodów *code*. Pozostałe centroidy są modyfikowane o współczynnik *1-esp* w drugiej pętli for.

Listing 3.12 Funkcja *lbg* - podwojenie ilości centroid

```
1     (...)
2
3       for(auto j = 0; j < code_size; ++j)
4       {
5         std::transform(code.at(j).cbegin(),
6                        code.at(j).cend(),
7                        code.at(j+code_size).begin(),
8                        [eps](double x){
9                          return x*(1.0+eps);
10                       });
11        for(auto && x: code.at(j))
12        {
13          x *= 1.0-eps;
14        }
15      }
16      code_size *= 2;
17
18    (...)
```

### Aktualizacja położeń centroid.

Poniższy kod realizuje funkcjonalność detekcji najbliżej centroidy z książki kodów *code* oraz obliczenia wartości miary dopasowania dla każdego wektora cech. Główna pętla for iterująca po każdym z wektorów cech aktualizuje kolejne wartości kontenerów *owner_ind*,

*owner_ dist* i *owner_ ind_ count*. Znalezienie najbliższej centroidy odbywa się poprzez analizę odległości euklidesowej do każdej centroidy przy pomocy funkcji *dis_ eu*.

Listing 3.13 Funkcja *lbg* - znalezienie najbliższej centroidy

```
1       (...)
2
3         std::fill(owner_ind_count.begin(), owner_ind_count.end(), 0);
4         //find owner indexes
5         auto a_vec_index =0;
6         for(auto a_vec_iter = acoustic_vectors.cbegin();
7             a_vec_iter != acoustic_vectors.cend();
8             ++a_vec_iter, ++a_vec_index)
9         {
10          std::vector<double> distances;
11          for(auto iter = code.cbegin();
12              iter != code.cbegin() +code_size;
13              ++iter)
14          {
15            distances.push_back(vq::dis_eu<K>(*a_vec_iter, *iter));
16          }
17          auto min_dist = std::min_element(distances.begin(),
18                                           distances.end());
19          auto cen_index = std::distance(std::begin(distances), min_dist);
20          owner_ind.at(a_vec_index) = cen_index;
21          owner_dist.at(a_vec_index) = distances.at(cen_index);
22          ++owner_ind_count.at(cen_index);
23        }
24
25      (...)
```

Kolejnym krokiem jest aktualizacja centroid poprzez obliczenie średniego wektora dla każdej z nich przez uwzględnienie tylko tych wektorów cech które należą do segmentu tej centroidy. Realizowane jest to w linii 16 listingu **3.14** za pomocą funkcji *std::transform* oraz kolejnej pętli for w której wykonywane jest dzielenie.

Listing 3.14 Funkcja *lbg* - obliczenie ogólnego dopasowania

```
1       (...)
2
3         for(auto cen_iter = code.begin();
4             cen_iter !=  code.begin() +code_size;
5             ++cen_iter)
6         {
7           std::fill(cen_iter->begin(), cen_iter->end(), 0);
8         }
9         a_vec_index =0;
10        for(auto a_vec_iter = acoustic_vectors.cbegin();
11            a_vec_iter != acoustic_vectors.cend();
12            ++a_vec_iter, ++a_vec_index)
13        {
14          auto owner_centroid_begin =
15                                code.at(owner_ind.at(a_vec_index)).begin();
16          std::transform(a_vec_iter->cbegin(), a_vec_iter->cend(),
17                         owner_centroid_begin, owner_centroid_begin,
18                        [](double x, double y){
19                          return x+y;
20                        });
21        }
22        auto cen_index = 0;
```

```
23          for(auto cen_iter = code.begin();
24              cen_iter !=  code.begin() +code_size;
25              ++cen_iter, ++cen_index)
26          {
27            for(auto && c: *cen_iter)
28            {
29              auto count = owner_ind_count.at(cen_index);
30              c /= count!=0?count:1;
31            }
32          }
33
34        (...)
```

### Suma miary dopasowania i decyzja o kontynuowaniu.

Ostatnią operacją wykonywaną w omawianej funkcji jest obliczenie sumy miar dopasowania które zostały już obliczonych we wcześniejszych etapach i znajdujących się w kontenerze *owner_dist*. Sumowanie jest realizowane w pierwszej pętli for w listingu **3.15** co w rezultacie daje wektor *distortion* posiadający sumę miar dopasowania dla każdej centroidy. Kolejna pętla for implementuje logikę decyzji o tym czy kontynuować procedurę aktualizacji centroid. Wykonywane jest to poprzez sprawdzenie czy przesunięcie którejkolwiek centroidy nie było większe niż współczynnik *esp*. Jeżeli ten warunek jest prawdą wtedy procedura odszukiwania najbliższych centroid i aktualizacji tychże jest uruchamiana ponownie poprzez odpowiednie ustalenie stanu zmiennej *distortion_diff_small_enough*.

Listing 3.15 Funkcja *lbg* - decyzja o kontynuowaniu

```
1         (...)
2
3           std::vector<double> distortion(owner_distortion.size(),0);
4           cen_index = 0;
5           for(auto i=0ul; i < acoustic_vectors.size(); ++i)
6           {
7             auto owner_index = owner_ind.at(i);
8             distortion.at(owner_index) += owner_dist.at(i);
9           }
10          for(auto i=0; i < code_size; ++i)
11          {
12            distortion_diff_small_enough = true;
13            auto d_coef =
14              (owner_distortion.at(i)-distortion.at(i))/distortion.at(i);
15            if(d_coef > eps)
16            {
17              distortion_diff_small_enough = false;
18              owner_distortion = std::move(distortion);
19              break;
20            }
21          }
22
23        (...)
```

## 3.6 Implementacja algorytmu decyzji przy pomocy ustalonego progu $\theta$

Została zaimplementowana najprostsza metoda dla deterministycznego modelowania, opisana w sekcji(**1.8.2**).

Listing 3.16 Funkcja *make_ decision* - decyzja o kontynuowaniu

```
1  inline bool make_decision(double distortion, double theta)
2  {
3    return distortion < theta;
4  }
```

W tym miejscu została opisana implementacja bloku podejmowania decyzji opisanego w **sekcji 2.2.2**.

# Rozdział 4

# Testy

## 4.1 Test ERR dla 11 mówców w bazie.

Zaimplementowany system weryfikacji mówcy został poddany statycznemu testowi na współczynnik błędu EER (*ang. Equal Error Rate*). Błąd ten jest zdefiniowany dla takiego systemu z progiem $\theta$, że błąd typu fałszywa akceptacja FAR - (*ang. False Acceptance Rate*) jest równy błędowi typu fałszywa odmowa - FRR (*ang. False Rejection Ratio*).

Błąd FAR jest zdefiniowany następująco:

$$FAR = \frac{n_{fa}}{n_r} \cdot 100\% \tag{4.1}$$

gdzie $n_{fa}$ oznacza ilość przypadków zaakceptowania mówcy, który nie jest mówcą weryfikowanym. $n_r$ oznacza całkowitą ilość prób weryfikacji takiego mówcy.

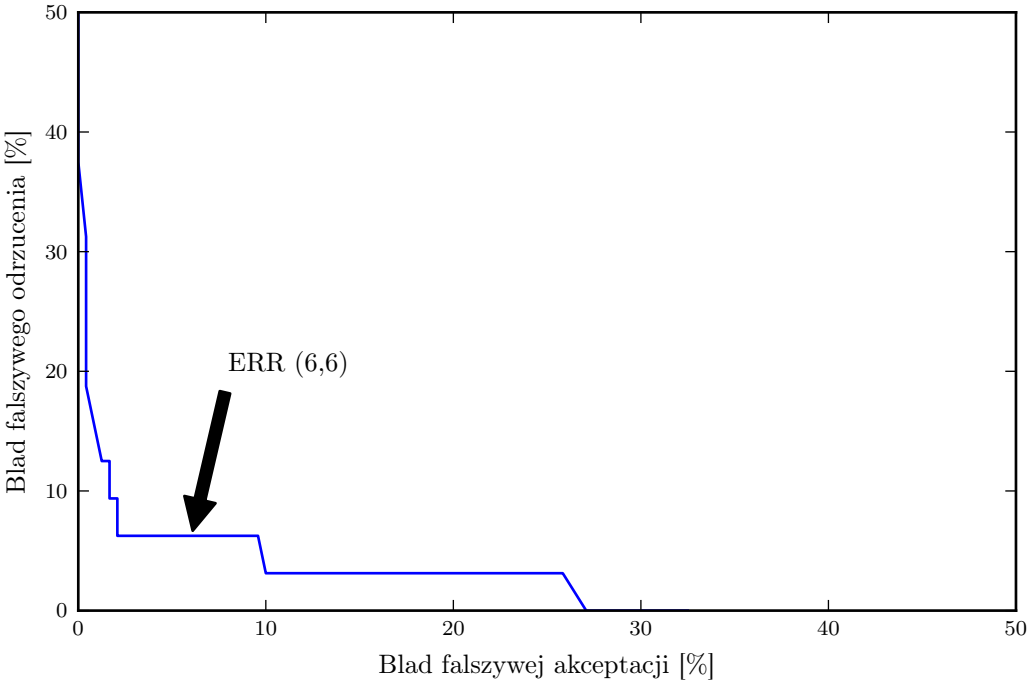
Błąd FRR jest zdefiniowany następująco:

$$FRR = \frac{n_{fr}}{n_a} \cdot 100\% \tag{4.2}$$

gdzie $n_{fr}$ oznacza ilość przypadków odrzucenia mówcy, będącego mówcą weryfikowanym. $n_a$ oznacza całkowitą ilość prób weryfikacji takiego mówcy.

W testowanym systemie zastosowano K=30 współczynników MFCC, ramki sygnału o długości N=256 oraz C=16 centroid dla modelu mówcy przy użyciu techniki VQ. Test składał się z przeanalizowania wpływu zależności stałego współczynnika $\theta$ (sekcja **1.8.2**) z szerokiego zakresu wartości na wielkość błędów FAR i FRR w systemie weryfikacji mówcy. W teście użyto 16 modeli mówców i 16 nagrań testowych angielskiego słowa 'zero' od tych samych mówców w celu oszacowania błędów FRR. Daje to 256 testów weryfikacyjnych dla każdej wartości współczynnika $\theta$. Dla oszacowania błędu FAR użyto 3 modeli mówców pochodzących z tej samej książki kodów wraz z 32 nagraniami pochodzącymi od od tych 3 mówców. Wyniki testu przedstawione są na rysunku (**4.1**). Wynik ERR wyniósł 6%.

Ze względu na małą ilość danych treningowych oraz danych testowych (szczególnie w przypadku szacowania błędu FAR) otrzymane wyniki mogą różnić się w pewnym stopniu od statystycznie istotnej oceny modelu.




Rysunek 4.1 Wykres zależności błędu fałszywych akceptacji i odrzuceń.

# Rozdział 5

# Podsumowanie

Powstały w niniejszej pracy system weryfikacji mówcy osiągnął dobry wynik błędu ERR równy 6% w teście zawierającym ograniczony materiał nagrań. Każde nagranie zostało zarejestrowane w podobnych warunkach z podobnymi zakłóceniami tła oraz zawierało tę samą treść (wyświetlane hasło). Z tych powodów aby otrzymać bardziej wiarygodne wyniki efektywności otrzymanego systemu należy wykonać bardziej rozległe testy, najlepiej przy użyciu dużych baz wypowiedzi mówców powstałych w tym właśnie celu takie jak baza NIST (*National Institute of Standards and Technology*) albo ELRA (European Language Resources Association). Implementacja otrzymanego przykładowego systemu składa się tylko z niezbędnych elementów potrzebnych do jego uruchomienia. Jednak przedstawiona w rozdziale 2 architektura systemu pozwala na szeroką ingerencję jeżeli chodzi o dodawanie do systemu nowych technik poprawiających efektywność systemu. W uruchomionym na potrzeby tej pracy systemie pożądanym ulepszeniem jest wprowadzenie technik detekcji mówcy (VAD) tak aby puste ramki cech MFCC (pomiędzy słowami) nie wpływały na efekt procesu tworzenia modelu mówcy (ze względu na faworyzowanie modeli z centroidami bliżej punktu zerowego). Tak jak wspomniano na wstępie pracy, warto dostarczyć do systemu więcej materiału trenującego (np. parokrotne powtórzenia wszystkich cyfr dla każdego mówcy w przypadku systemu z wyświetlanym hasłem). Kolejnym ulepszeniem systemu mogłoby być w zastosowaniach z przetwarzaniem sygnału mowy w czasie rzeczywistym uwzględnienie podsystemu *speaker pruning* (opisanego w sekcji **1.8.3**) może to okazać się przydatne dla zwiększenia szybkości pracy systemu dla weryfikacji z modelami kohort.

Autor uważa, że zaproponowana architektura i sposób implementacji w języku C++ może być bazą do stworzenia obszernego narzędzia dla prac związanych z konstruowaniem systemów weryfikacji mówcy dla systemów wbudowanych w czasie rzeczywistym. Wydaje się, że jest możliwa efektywna i jasna implementacja w proponowanym środowisku większości technik weryfikacji mówcy omawianych w części teoretycznej tej pracy.

# Spis treści

# Bibliografia


[1] Homayoon Beigi. *Fundamentals of Speaker Recognition*. 1st ed. Springer US, 2011.

[2] Florian Hönig Georg Stemmer Christian Hacker Fabio Brugnara. *Revising Perceptual Linear Prediction (PLP)*. 2005.

[3] *Digital Signal Processing Mini-Project: An Automatic Speaker Recognition System*. dostęp 3.06.2017. URL: `http://minhdo.ece.illinois.edu/teaching/speaker_recognition/`.

[4] Douglas A. Reynolds Thomas F. Quatieri Robert B. Dunn. "Speaker Verification Using Adapted Gaussion Mixture Models." In: *Digital Signal Processing* (2000).

[5] Tomi Kinnunen Evgeny Karpov Pasi Fränti. "Real-Time Speaker Identification and Verification". In: *IEEE trans. speech & audio processing* (2006).

[6] H. Hermansky. "Perceptual linear predictive (PLP) analysis of speech." In: *The Journal of the Acoustical Society of America* (1990).

[7] Jr. Joseph P. Campbell. "Speaker Recognition: A Tutorial". In: *IEEE, vol. 85, no. 9, september 1997* (1997).

[8] F. K. Soong A. E. Rosenberg L. R. Rabiner B. H. Juang. *A Vector Quantization Approach to Speaker Recognition*. AT&T Bell Laboratories Murray Hill, New Jersey 07974. 1992.

[9] Richard M. Stern shitiz Kumar Chanwoo Kim. *Delta-spectral cepstral coefficients for robust speech recognition*. 2004.

[10] Biing-Hwang Juang Lawerence Rabiner. *Fundamentals of speech recognition*. Prentice-Hall International, 1990.

[11] A. Michael Noll. "Cepstrum Pitch Determination". In: *The Journal of the Acoustical Society of America* (1966).

[12] Alan V. Oppenheim and Ronald W. Schafer. "From Frequency to Quefrency: A History of the Cepstrum". In: *IEEE signal processing magazine* (2004).

[13] Roman Rumian Redakcja: Tomasz P. Zieliński Przemysław Korohoda. *Cyfrowe przetwarzanie sygnałów w telekomunikacji*. Wydawnictwo Naukowe PWN, 2014. ISBN: 9788301174453.

[14] *Ślimak rysunek*. dostęp 3.06.2017. URL: `https://www.britannica.com/science/ear/Transmission-of-sound-by-bone-conduction`.

[15] Haizhou Li Tomi Kinnunen. "An Overview of Text-Independent Speaker Recognition: from Features to Supervectors". In: *Speech Communication* (2009).

[16] Lantian Li Yixiang Chen Ying Shi Zhiyuan Tang Dong Wang. "Deep Speaker Feature Learning for Text-independent Speaker Verification". In: (2017).

[17] *xQueue reference*. dostęp 3.06.2017. URL: http://www.freertos.org/Embedded-RTOS-Queues.html.

[18] R. Gray Y. Linde A. Buzo. "An algorithm for vector quantizer design". In: *IEEE Transactions on Communications Vol. 28 pp.84-95* (1980).

[19] *Youtube statictics*. dostęp: 20.06.2017. URL: www.statisticbrain.com/youtube-statistics/.