

Politechnika Śląska  
Wydział Automatyki, Elektroniki  
i Informatyki  
Instytut Informatyki

---

*Rozprawa doktorska*

**Komputerowe rozpoznawanie niepłynności mowy  
z zastosowaniem transformaty falkowej  
i sztucznych sieci neuronowych**

*Ireneusz Codello*

*Promotor: dr hab. Wiesława Kuniszyk-Jóźkowiak, prof. nadzw.*

Praca wykonana w Zakładzie Biocybernetyki Instytutu Informatyki  
Uniwersytetu Marii Curie-Skłodowskiej,  
w ramach studiów doktoranckich Politechniki Śląskiej na kierunku Informatyka,  
przedstawiona Radzie Wydziału Automatyki, Elektroniki i Informatyki  
Politechniki Śląskiej, jako rozprawa doktorska.

GLIWICE 2014

Składam serdeczne podziękowania  
Pani dr hab. Wiesława Kuniszyk-Józkowiak, prof. nadzw.  
za zaproponowanie bardzo ciekawej tematyki  
oraz cenne uwagi merytoryczne, bez których ta praca by nie powstała.

## Spis treści

1.	Wstęp .....	6
2.	Cel i tezy pracy .....	9
3.	Zagadnienia niepełności mowy .....	10
4.	Parametryzacja sygnału mowy .....	17
4.1.	Dyskretna transformata Fouriera .....	17
4.2.	Metoda liniowej predykcji .....	19
4.3.	Transformata falkowa .....	21
4.3.1.	Ciągła transformata falkowa .....	23
4.3.2.	Dyskretna transformata falkowa .....	28
4.4.	Konstrukcja falki macierzystej .....	31
4.5.	Skala barkowa .....	34
5.	Grupowanie i klasyfikacja danych z zastosowaniem sztucznych sieci neuronowych ...	36
5.1.	Sieci Kohonena .....	36
5.2.	Perceptron wielowarstwowy .....	42
6.	Algorytmy do automatycznego rozpoznawania niepełności w mowie ciągłej.....	46
6.1.	Parametryzacja sygnału przy użyciu CWT .....	46
6.2.	Tworzenie wektorów wejściowych na podstawie skalogramu CWT .....	47
6.3.	Detekcja początku i końca fonacji .....	48
6.4.	Ocena wyników rozpoznawania .....	49
6.5.	Wybór algorytmów oraz wyniki rozpoznawania .....	49
6.5.1.	Metoda detekcji przedłużeń .....	50
6.5.1.1.	<i>Material dźwiękowy</i> .....	50
6.5.1.2.	<i>Analiza wyników parametryzacji</i> .....	51
6.5.1.3.	<i>Procedura detekcji przedłużeń</i> .....	54
6.5.1.4.	<i>Wyniki rozpoznawania przedłużeń</i> .....	55

6.5.2.	Metoda detekcji powtórzeń głosek .....	57
6.5.2.1.	<i>Material dźwiękowy</i> .....	57
6.5.2.2.	<i>Analiza wyników parametryzacji</i> .....	59
6.5.2.3.	<i>Procedura detekcji powtórzeń głosek</i> .....	62
6.5.2.4.	<i>Wyniki rozpoznawania powtórzeń głosek</i> .....	64
6.5.3.	Metoda detekcji powtórzeń sylab .....	68
6.5.3.1.	<i>Material dźwiękowy</i> .....	68
6.5.3.2.	<i>Analiza wyników parametryzacji</i> .....	70
6.5.3.3.	<i>Procedura detekcji powtórzeń sylab</i> .....	73
6.5.3.4.	<i>Wyniki rozpoznawania powtórzeń sylab</i> .....	74
7.	Program do analizy i rozpoznawania nie płynności mowy .....	78
8.	Podsumowanie .....	79
8.1.	Przegląd użytych metod.....	79
8.2.	Podsumowanie wyników rozpoznawania .....	80
8.3.	Wnioski końcowe.....	83
9.	Bibliografia .....	84
10.	Dodatek A .....	93
10.1.	Ogólny opis programu WaveBlaster.....	93
10.1.1.	A – obszar menu .....	94
10.1.2.	B – obszar przycisków widoczności paneli .....	94
10.1.3.	C – obszar informacji pozycji kursora myszy .....	95
10.1.4.	D – obszar opcji paneli .....	95
10.1.5.	E – obszar paneli.....	96
10.1.6.	F – obszar statusu i wskaźnika postępu .....	97
10.2.	Panel oscylogramu .....	97
10.3.	Panele analizy Fouriera.....	98
10.4.	Panele analizy liniowej predykcji .....	99
10.5.	Panele transformaty falkowej .....	100

10.6.	Panel widm.....	101
10.7.	Panel sieci Kohonena.....	102
10.8.	Panel komunikatów.....	104
10.9.	Panel wzorców .....	105
10.10.	Zakładka automatycznej detekcji niepłynności .....	106
10.11.	Opcja uruchamiania skryptów .....	107
11.	Dodatek B .....	109

## 1. Wstęp

Analiza mowy jest dziś bardzo ważną gałęzią informatyki - ustna komunikacja z komputerem może być pomocna przy pisaniu dokumentów, tłumaczeniu języków lub po prostu w codziennej pracy przy komputerze. Bardzo praktycznym zastosowaniem jest również rozpoznawanie mówców, płci czy analiza schorzeń narządów traktu głosowego [4] [5] [19] [20] [21] [22] [23] [24] [25] [36] [86] [91]. Z tego powodu to zagadnienie (oraz inne zagadnienia przetwarzania sygnałów np. sonarowych czy sejsmicznych) jest analizowane od wielu lat, co doprowadziło do stworzenia lub zaadoptowania na potrzeby cyfrowego przetwarzania sygnałów (ang. digital signal processing – DSP) wielu algorytmów służących do ekstrakcji danych, takich jak: transformata Fouriera [34] [51] [80], analiza cepstralna [10] [28] [51], liniowa predykcja [7] [38] [40] [42] [47] [51], analiza falkowa [9] [15] [16] [17], filtry [66] i inne... jak i algorytmów rozpoznawania mowy, mówców lub innych cech w sygnale: ukryte łańcuchy Markowa [88] [89] [90] [92], sieci neuronowe [35] [58] [59] [60] [63] [62] [61] [74] [73], logika rozmyta [66], metody analityczne jak analiza obwiedni [43], współczynników korelacji [67] i inne.

Rozpoznawanie zaburzeń w mowie w pewnym sensie jest podobnym problemem – używając algorytmów DSP należy wyekstrahować cechy, które dobrze różnicują fragmenty mowy płynnej od niepłynnej. Jednocześnie zagadnienie to jest na tyle inne, że istnieje dość niewiele prób jego rozwiązania w literaturze polskiej i światowej. Oczywiście niepłynności mowy mogą być różnej natury, więc wydaje się niemożliwe stworzenie jednego algorytmu, który znajdowałby je wszystkie. Należy zatem je pogrupować, a następnie opracowywać algorytm detekcji niezależnie dla każdej grupy.

Odsłuchowe oznaczanie niepłynności przez terapeutę jest obarczone dużym błędem. Trzeba wielokrotnie odsłuchać nagranie, żeby wychwycić wszystkie zaburzenia. Ponieważ percepcja terapeuty znacząco zależy od jego wypoczęcia, poziomu koncentracji w danej chwili oraz od innych subiektywnych czynników – terapeuta ten sam fragment raz może uznać jako płynny, a po ponownym odsłuchaniu jako niepłynny. Automatyczne rozpoznawanie niepłynności, które działa zawsze na podstawie takich samych reguł – jest bardziej obiektywne. Ponadto automatycznie generowane statystyki mowy patologicznej byłoby dużym ułatwieniem dla terapeutów w ich próbach oszacowania postępów terapii. Terapeuta mógłby nagrywać wypowiedzi pacjenta podczas terapii i uruchamiać program

zliczający nie płynności w tych nagraniach, ponieważ robienie tego manualnie jest procesem żmudnym i czasochłonnym. Oprócz statystyk liczby i czasów trwania nie płynności, terapeuta otrzymywałby listę wszystkich nie płynności, dzięki której łatwo mógłby wyszukiwać i odsłuchiwać wyselekcjonowane fragmenty. Narzędzie automatycznie rozpoznające nie płynności:

- wyręczyłoby terapeuta w żmudnym zliczaniu i klasyfikowaniu nie płynności w nagraniach pacjenta,
- zwiększyłoby obiektywizm oceny – w terapii, z sesji na sesję, pokazując statystyki oparte na dokładnie takich samych regułach wyszukiwania (brak subiektywnych „ludzkich” czynników), co mogłoby mieć duże znaczenie dla diagnozy i wyboru ćwiczeń dla pacjenta,
- zaoszczędziłoby czas terapeuty, który mógłby go przeznaczyć chociażby na dłuższe sesje terapeutyczne.

Plan pracy jest następujący:

W rozdziale 2 przedstawiony został cel i tezy pracy.

W rozdziale 3 znajduje się przegląd literatury dotyczącej rozpoznawania nie płynności mowy. Opisane są w nim rodzaje nie płynności badane przez naukowców, stosowane przez nich metody oraz uzyskane wyniki rozpoznawania.

Opis niezbędnych podstaw teoretycznych potrzebnych do zrozumienia użytych algorytmów znajduje się w rozdziale 4 i 5. Jako podstawową metodę parametryzacji, na bazie której budowane są dopiero kolejne algorytmy, przyjęto ciągłą transformatę falkową (CWT). Autor uznał tę metodę za bardziej odpowiednią od transformaty Fouriera czy liniowej predykcji (uzasadnienie w rozdziale 4) przy analizie sygnału o składowych zarówno szybko jak i wolno zmieniających się. Za istotne, na korzyść CWT, autor uznał również elastyczność w doborze skal – przyjęto skalę barkową, jako popularną i sprawdzoną skalę percepcyjną. Ponadto wykorzystano sztuczne sieci neuronowe – sieci Kohonena (jako reduktor wymiaru wektorów uzyskanych z CWT) i perceptron 3-warstwowy (jako klasyfikator).

Sposób użycia w/w algorytmów wraz z doбором ich parametrów oraz uzyskanymi wynikami opisany został w rozdziale 6. Parametry poszczególnych rodzajów nie płynności (ich charakterystyka, granice zmienności, czasy trwania) zostały określone na podstawie obszernego zbioru nie płynnych wypowiedzi osób jękających się i porównaniu ich z płynnymi

wypowiedziami. Dla każdej grupy niepełności został zastosowany inny algorytm. Mają one jednak wiele cech wspólnych – przede wszystkim wstępną parametryzację sygnału metodą ciągłej analizy falkowej.

Rozdział 7 jest opisem programu „WaveBlaster” przy użyciu którego wykonane były prawie wszystkie badania oraz uzyskane zostały prawie wszystkie wyniki, statystyki i wykresy. Jako, że autor tej rozprawy napisał program „WaveBlaster” na potrzeby opisywanych tu badań, w rozdziale zawarto krótki opis jego struktury i algorytmów w nim zawartych. Opis funkcjonalności, wspomagany zrzutami ekranów wykonanych na kolejnych panelach programu, opisano w dodatku. Do pracy dołączono 4 pliki multimedialne prezentujące:

- ogólny opis funkcjonalności programu WaveBlaster,
- procedurę detekcji przedłużeń przy użyciu programu WaveBlaster,
- procedurę detekcji powtórzeń głosek przy użyciu programu WaveBlaster,
- procedurę detekcji powtórzeń sylab przy użyciu programu WaveBlaster.

Rozdział 8 zawiera podsumowanie i uwagi końcowe.

Część badań opisanych w tej pracy została opublikowana w 12 pracach [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17]. Najnowsza z tych prac [13] otrzymała wyróżnienie (Dodatek B) na międzynarodowej konferencji CORES 2013 oraz uzyskała zaproszenie do publikacji pokonferencyjnej w czasopiśmie "Pattern Analysis and Applications" i w wersji rozszerzonej została wysłana do druku.



## 2. Cel i tezy pracy

Celem pracy jest opracowanie systemu do automatycznego rozpoznawania niepełności w mowie ciągłej z precyzyjną lokalizacją ich w czasie. Przyjęte zostały następujące tezy:

- 1) Parametryzacja sygnału mowy przy zastosowaniu ciągłej transformaty falkowej pozwala na optymalne przybliżenie własności percepcyjnych słuchu ludzkiego, przy odpowiedniej konstrukcji falki macierzystej. Taki sposób parametryzacji ma bardzo istotne znaczenie przy rozpoznawaniu niepełności, gdyż tworzony system powinien naśladować odbiór zaburzeń przez słuchaczy i samego mówiącego.
- 2) Optymalnym sposobem redukcji nadmiarowości wektorów wejściowych jest zastosowanie sieci Kohonena, pozwala ona bowiem na nieeliminowanie czasu, który w przypadku tych zaburzeń odgrywa istotną rolę. Ponadto odpowiednia modyfikacja algorytmu uczenia tych sieci daje możliwość bardzo dobrego odwzorowania przedłużeń oraz powtórzeń głosek.
- 3) Sieci neuronowe są bardzo dobrymi klasyfikatorami powtórzeń głosek.
- 4) Klasyfikacja powtórzeń sylab jest możliwa przy zastosowaniu odpowiednio dostosowanych algorytmów korelacyjnych.
- 5) Możliwe jest skutecznie rozpoznawanie niepełności w mowie ciągłej z precyzyjną lokalizacją w skali czasu przy zastosowaniu ciągłej transformaty falkowej z konstrukcją skal barkowych.

### 3. Zagadnienia nie płynności mowy

Zaburzenia mowy, czyli fragmenty mowy uznane za nie płynne, mogą być definiowane bardzo różnie – widać to szczególnie dobrze w pracach o charakterze logopedycznym.

Johnson (USA, 1959) [cyt. za [46]] klasyfikował je według: powtórzenia części słów (głosek, sylab), powtórzenia słów, powtórzenia zdań, wtrącenia, przedłużenia głosek, przerwane słowa, niedokończone frazy, poprawki. Vaane i Janssen (Holandia, 1977) [cyt. za [46]] wyróżniali ich więcej: ciche blokady, blokady i artykulacja, blokady i pewne wyuczone zachowania, przedłużenia, wtrącenia głosek, wtrącenia powtórzonych głosek, wtrącenia przedłużonych głosek, wtrącenia oddechowe, szybkie powtórzenia głosek i sylab, szybkie powtórzenia słów, powolne powtórzenia słów, powolne powtórzenia sylab, powolne powtórzenia głosek, powtórzenia zdań, poprawki zdań. Moriyama, Ooaka, Ozawa, Kurishina, Suzuki, Tsuzuki (Japonia, 1995) [cyt. za [46]] wyróżnili ich już aż 18: powtórzenia głosek i sylab, powtórzenia części słów, przedłużenia spółgłosek, przedłużenia samogłosek, nacisk (wybuch), zniekształcenia, przerywania, blokady, zastanawianie się, nienormalne oddychanie, powtórzenia słów i zdań, poprawki, nie dokończone zdania, wtrącenia, pauzy, zmiany tempa, zmiany głośności, wysokości, barwy, mówienie na pozostałym powietrzu.

Bazując na w/w opisie, można zauważyć jak złożona jest analiza nie płynności. Na pewno zależy ona od podejścia samych badaczy (jakie cechy wypowiedzi uznają za istotne i na tyle odmienne, żeby je wydzielić – aby na przykład lepiej dostosować rodzaj terapii). Ogromne znaczenia ma również język wypowiedzi, a raczej jego znajomość – jak inaczej można by rozpoznać niedokończone zdania, wtrącenia lub choćby przedłużania (w języku japońskim są one częste w mowie płynnej). Rozpoznawanie nie płynności jest zatem procesem bardzo złożonym, wymagającym dużej wiedzy o języku. Obecnie wiedza i doświadczenie terapeuty, oceniającego rodzaje nie płynności oraz ich stopień, wydaje się nie możliwa do przełożenia na w pełni automatyczny system informatyczny. Z tego powodu badacze próbują skonstruować programy wspierające pracę terapeuty, analizujące najczęściej występujące nie płynności – reszta analizy (opartej na języku, zwyczajach językowych, zachowaniu pacjenta) nadal jest w rękach logopedy. Ponieważ metody te nie opierają się na tak dokładnej znajomości języka – liczba grup nie płynności jest dużo mniejsza, a rodzaje nie płynności wyodrębniane przez poszczególnych badaczy są do siebie podobne.

Wzrost mocy obliczeniowej komputerów osobistych oraz ich dostępności nastąpił pod koniec XX wieku – zatem stosunkowo niedawno. Dopiero wtedy badacze uzyskali techniczne możliwości, dzięki którym nastąpił m. in. rozwój algorytmów cyfrowego przetwarzania sygnałów oraz rozwój prac nad automatyczną analizą nie płynności mowy.

W tym okresie najbardziej zaawansowane były prace Howell'a i współpracowników. W 1995 roku, Howell i Sackin [31] skupili się na rozpoznawaniu powtórzeń i przedłużeń. W swych badaniach korzystali ze współczynników spektralnych uzyskanych z 19-kanalowego vocodera i ich współczynników autokorelacji oraz z obwiedni sygnału mowy. Fragmenty z mową były klasyfikowane do zbioru płynny/niepłynny przy użyciu sztucznych sieci neuronowych ze skutecznością 82% dla przedłużeń oraz 77% dla powtórzeń. Te same rodzaje nie płynności były przedmiotem ich badań w 1997 (Howell, Sackin, Glenn) [32] [33]. Użyli nagrań 12 dzieci jękających się (w języku angielskim) czytających historię składającą się z 376 słów (90% słów jednosylabowych), które manualnie podzielili na wyrazy. Pozostawili jedynie nie płynności pojawiające się w pojedynczych słowach tj: przedłużenia, powtórzenia części i całych wyrazów oraz niedokończone wyrazy. Wyodrębniali wiele parametrów: długość wyrazów, długość kawałków wyrazów (wyodrębniane były kawałki powyżej i poniżej granicznej wartości energii), ich liczba oraz zmienność ich energii, długość fragmentów ciszy, ich liczba oraz zmienność ich energii, wartości współczynników spektralnych dla w/w fragmentów. Tutaj również klasyfikatorem była sieć neuronowa. Wyniki dla dziesięciu najlepiej rozpoznawających sieci były na poziomie: wyrazy płynne 93%-95%, przedłużenia 32%-62%, powtórzenia 31%-53%.

W 2000 roku, Nöth wraz zespołem [48] przeprowadzili badania na nagraniach uzyskanych od 37 pacjentów. Manualnie oznaczyli oni fonemy oraz początki i końce sylab. Zaprezentowali sumaryczne wyniki rozpoznawania kilku rodzajów nie płynności: powtórzenia (od fonemów po całe wyrażenia), przedłużenia fonemów i wyrazów, niechciane przerwy lub pauzy. Do rozpoznawania nie płynności użyli Ukrytych Modeli Markowa (HMM) trenowanych przez 10 godzin nagraniami mowy 'spontanicznej' (tj. różnorodny tekst) uzyskując wartość 0.99 współczynnika korelacji dla mowy nie płynnej. Autorzy zauważyli również, że ich system miał tendencje do nadmiernego wykrywania zaburzeń mowy – zetem wysokie współczynniki były też uzyskiwane dla mowy płynnej. Badacze w swoje pracy zastanawiali się również, który parametr najlepiej odzwierciedla natężenia jękania uznając, że jest to całkowita długość ciszy w nagraniu połączona z wartością ilości nie płynności przypadającej na jeden wyraz.

W 2003 roku, Czyżewski, Kaczmarek i Kostek [18] uznali, iż automatyczne zliczanie niepełności daje bardziej obiektywne oszacowania i pod tym względem są one lepsze od subiektywnych metod – takich jak odsłuchiwanie nagrań i zliczanie niepełności przez człowieka. Stwierdzili również, iż wystarczy, aby taki system bazował na detekcji przerw, powtórzeń oraz przedłużeń. W swojej pracy do parametryzacji sygnału użyli cepstrum obliczonego z wygładzonego spektrum transformaty cosinusowej, na podstawie którego wyznaczyli 7 parametrów: ton podstawowy, częstotliwość i amplituda pierwszego formantu, częstotliwość i amplituda drugiego formantu oraz częstotliwość i amplituda trzeciego formantu. Następnie przeprowadzili eksperymenty przy użyciu algorytmu klasyfikującego, opartego na teorii zbiorów przybliżonych. W przypadku detekcji przerw, do klasyfikacji użyli dodatkowo sztucznych sieci neuronowych (dla porównania). Do analizy użyli 12 plików (6 z niepełnością i 6 płynnych) otrzymując wynik 91% (11 plików poprawnie sklasyfikowanych) a dla sieci neuronowych otrzymując 78%. Przy detekcji przedłużeń użyli 36 nagrań (każde zawierało jedną samogłoskę), a w kolejnym doświadczeniu z 60 nagrań uzyskując rozpoznawanie odpowiednio 97% (35 poprawnych/ 1 błędny) i 93% (56 poprawnych/ 2 błędne/ 2 brak decyzji). W detekcji powtórzeń użyto 20 nagrań (10 płynnych/10 niepełnych) uzyskując wynik 65% (13 poprawnie sklasyfikowanych/ 6 błędnie/ 1 brak decyzji).

W latach 2003-2006 Suszyński, Kuniszyk-Józkowiak stworzyli algorytmy automatycznej detekcji czterech rodzajów niepełności: przedłużeń [68], powtórzeń sylab [70], blokad [69] i wtrąceń [71]. Wszystkie metody korzystały z takiego samego algorytmu parametryzacji nagrań dźwiękowych: analiza FFT oknem 23ms, następnie filtracja filtrami 1/3 oktawowymi, zastosowanie filtru korekcyjnego A, normalizacja wektorów. Do analiz przedłużeń autorzy użyli nagrań jedenastu osób jękających się. Badane osoby czytały ten sam tekst składający się z 200 słów oraz omawiały głośno ten sam obrazek (opowiadania te zawierały około 100 słów). Wyodrębniono następujące cechy dystynktywne przydatne w rozpoznawaniu przedłużeń: pasma filtrów 1/3 oktawowych, w których były zlokalizowane maksima poziomu dźwięku; zakres zmian szerokości pasma zawierającego maksimum; zakres zmian średniego poziomu; czas, w którym maksimum było zlokalizowane w danym paśmie i zmieniało się w danym zakresie. Na podstawie tych cech autorzy stworzyli zbiory rozmyte oraz reguły wnioskowania rozmytego przy pomocy których, system przydzielał analizowane fragmenty do zbiorów płynny/niepełny (jednocześnie dokładnie umiejscawiając początek i koniec niepełności). Program rozpoznał 91% przedłużeń. Przy detekcji blokad autorzy analizowali

czasy trwania fragmentów mowy oraz otaczających go przerw, na podstawie których, stworzyli zbiory oraz reguły rozmyte. Analizowane były nagrania wypowiedzi 11 osób, z których zostały wycięte 4-sekundowe fragmenty (każdy zawierający blokadę). Nagrano również płynne odpowiedniki w/w wypowiedzi – łącznie 150 czterosekundowych nagrań (połowa zawierała nie płynności). Fragmenty w całości były oceniane przez system, jako płynne lub nie płynne – uzyskano 95% skuteczność. Przy detekcji powtórzeń głosek użyto algorytmu korelacji, a stosowano go na wektorach spektrogramu FFT (z oknem = 23,22ms) zredukowanego filtrami 1/3 oktawowymi. Szerokość korelowanego fragmentu  $T$  oraz optymalny dystans  $\Delta t$  do drugiego okna, z którym sygnał był korelowany, szacowano na podstawie 125 czterosekundowych fragmentów z nie płynnościami (każdy fragment zawierał jedno lub wiele powtórzeń jednej sylaby) oraz 110 czterosekundowych nagrań wypowiedzi płynnych. Zadawalające rezultaty uzyskano dla  $\Delta t = 93\text{ms}$  i  $116\text{ms}$  (odpowiednio cztero i pięć-krotność okna FFT) oraz dla szerokości korelowanych fragmentów również równych  $T=93\text{ms}$  i  $116\text{ms}$ . Do oceny skuteczności użyto współczynników czułości i przewidywalności [2], które wyniosły około 70%. Przy detekcji wtrąceń autorzy skorzystali z tych samych algorytmów jak przy detekcji powtórzeń sylab. Różnicą było to, iż w programie należało zaznaczyć fragment nagrania, który był uznany jako wtrącenie, a następnie był wyszukiwany w całym nagraniu – wynikało to z faktu iż każdy pacjent może wtrącać różne fonemy czy sylaby, zatem niemożliwe jest założenie z góry co konkretnie ma być uznawane za wtrącenie. W badaniach użyto nagrań wypowiedzi siedmiu osób o czasach trwania od 2-4 minut zawierających łącznie 170 wtrąceń. Uzyskane wartości czułości i przewidywalności były rzędu 80%.

W latach 2007-2009 Szczurowska (później Świetlicka), Kuniszyk-Józkowiak analizowały zagadnienia rozpoznawania blokad [76], przedłużeń głosek [72] oraz powtórzeń sylab [73] przy użyciu sztucznych sieci neuronowych. Wybrany został ten sam sposób parametryzacji sygnału: analiza FFT oknem 23ms, następnie filtracja filtrami 1/3 oktawowymi, zastosowanie filtru korekcyjnego A, normalizacja wektorów. Takie trójwymiarowe spektrogramy powstałe na bazie filtrów tercjowych były redukowane do dwóch wymiarów przy użyciu samoorganizujących się map (SOM, nazywane również sieciami Kohonena), przeobrażając wielowymiarowy wektor w pojedynczą wartość – indeks wygrywającego neuronu. Autorki przeanalizowały zachowanie tych sieci (sprawdzana była jakość uczenia, walidacji i testowania oraz błąd uczenia, walidacji i testowania) dla różnych parametrów rozmiaru sieci, uczenia i sąsiedztwa. W badaniach użyto 55 próbek płynnych i 55 nie płynnych w przypadku

blokad, 39 płynnych i 39 niepłynnych w przypadku powtórzeń sylab, 59 płynnych i 59 niepłynnych w przypadku przedłużeń głosek. Następnie wynikowe wektory sieci SOM klasyfikowane były do grup płynne/niepłynne za pomocą trzech klasyfikatorów – sieci SOM (self-organized maps), sieci MLP (multi-layer perceptron) oraz sieci RBF (radial basis function). Sieciami SOM udało uzyskać się rozpoznawania na poziomie: blokady 88%, powtórzenia sylab 87%, przedłużenia 74%. Sieciami MLP: blokady 96%, powtórzenia 95%, przedłużenia 99%. Sieciami RBF: blokady 92%, powtórzenia 82%, przedłużenia 69%.

W latach 2007-2011 Wiśniewski, Kuniszyk-Józkowiak [88] [89] [90] [87] do rozpoznawania przedłużeń i powtórzeń głosek użyli ukrytych modeli Markova. Parametryzacja sygnału wejściowego przebiegała według następującego algorytmu: podzielenie sygnału na 512 próbkowe ramki, obliczenie FFT dla każdej ramki, przeskalowanie widma do skali melowej, przefiltrowanie 20 filtrami trójkątnymi, wyliczenie melowych współczynników cepstralnych (MFCC). Następnie, w celu wygenerowania książki kodowej, wybierany był plik zawierający możliwie wszystkie fonemy, był on parametryzowany według w/w algorytmu, a następnie algorytmem k-means tworzona była książka kodowa. W pierwszym eksperymencie [88] posłużono się dwudziestoma czterema nagraniami, gdzie każde nagranie zawierało jedną niepłynność – łącznie 14 przedłużeń oraz 10 powtórzeń. Uzyskano wynik rozpoznawania powtórzeń głosek wynoszący 80%: oraz przedłużeń wynoszący 62%. W kolejnej pracy [89] skupiono się na przedłużeniach testując łącznie 192 modeli różniących się rozmiarem książki kodowej oraz liczbą stanów. Zwiększono również liczbę testowanych przedłużeń do 22. Najlepsze wyniki, około 80%, uzyskano dla wszystkich modeli z książką kodową o rozmiarze 512 (liczba stanów modelu okazała się nie mieć wpływu na wyniki). W 2010 autorzy poszli krok dalej tworząc algorytm do detekcji przedłużeń w mowie ciągłej z jednoczesną detekcją rodzaju przedłużanego fonemu [90]. Tak jak w poprzednich badaniach korzystano z książki kodowej o rozmiarze 512 wyznaczoną przy użyciu parametrów MFCC, ponadto obliczano również energię sygnału. Na podstawie nagrań dwóch osób o łącznej długości 4min 30sec utworzono 37 niezależnych modeli dla poszczególnych fonemów. System rozpoznał 80% przedłużeń. W 2011 Wiśniewski i Kuniszyk-Józkowiak zdecydowali się skorzystać z popularnego narzędzia HTK (Hidden Markov Model Toolkit) do rozpoznawania powtórzeń głosek. Jako materiału treningowego użyto 425 wypowiedzi (każde o długości od 100ms do kilku sekund). Dla każdej z nich utworzono zapis fonetyczny korzystając z 39 symboli (37 dla fonemów i 2 dla ciszy) i tak jak w poprzedniej pracy, dla każdego symbolu utworzono odrębny model. Do

testowania detekcji powtórzeń użyto 79 wypowiedzi (20 z nie płynnościami głosek k, d, t, p; reszta wypowiedzi z mową płynną). Uzyskano 89% skuteczność rozpoznawania powtórzeń. Trafność detekcji rodzaju powtarzanej głoski wyniósł 50%.

W 2008 Ravikumar, Rajagopal i Nagaraj [54] zaproponowali 4-etapowy algorytm detekcji powtórzeń sylab składający się z segmentacji, wyodrębnienia cech, dopasowania wyników i decyzji. W badaniach użyto 150 angielskich słów wypowiedzianych przez dziesięciu mówców, w których sylaby wydzielono manualnie. Do parametryzacji danych użyto 12 współczynników MFCC. Jako klasyfikatora (etap decyzyjny) użyto perceptronu, uczonego nagraniami ośmiu mówców, pozostałe nagrania (dwóch mówców) posłużyły do testowania uzyskując wynik 83%. W 2009 [53] ten sam zespół zaproponował ulepszenie w/w algorytmu poprzez zastosowanie klasyfikatora SVM (Support Vector Machine) – algorytmu wyznaczającego hiperpłaszczyznę oddzielającą zbiór wektorów reprezentujących wypowiedzi płynne od nie płynnych. Uśredniony wynik rozpoznawania wyniósł 94%. W 2011 Ravikumar, Ganesan [52] porównali wpływ liczby współczynników MFCC przy rozpoznawaniu nie płynności – do tego celu użyto tego samego zestawu danych wejściowych. Uprościli algorytm klasyfikujący, w którym wielowymiarowe wektory zredukowano do jednej wartości. Próg dzielący oba zbiory został wyznaczony na podstawie ośmiu nagrań (z dziesięciu). Przetestowano konfiguracje: 12, 13, 26 oraz 39 współczynników MFCC uzyskując najlepsze rozpoznawanie na poziomie 85% dla 39 wartości MFCC.

W latach 2010-2013 Kobus, Kuniszyk-Jóźkowiak skupili się na detekcji zaburzeń mowy przy użyciu parametrów LP (Linear Prediction). W 2010 [38] analizowali blokady p, b, t, d, k, g – blokadę definiujemy jako dłuższą ciszę zakończoną ploszą danej głoski. Z nagrań siedemnastu osób manualnie wycięto 4-sekundowe fragmenty z nie płynnościami oraz płynne fragmenty. Następnie każde fragment podzielono na okna i dla każdego okna wyliczono 15 współczynników LP. Tak uzyskane wektory były zredukowane sieciami Kohonena o rozmiarach 5x5, 6x6 i 7x7 do pojedynczych wartości – indeksu wygrywającego neuronu sieci Kohonena, a następnie były zaliczane do zbioru płynny/nie płynny przez klasyfikatory: sieci RBF (Radial Basis Function), sieci liniowe, sieci MLP (Multi-Layer Perceptron). Klasyfikatory były tworzone i testowane narzędziem Intelligent Problem Solver z pakietu STATISTICA. Najlepsze wyniki osiągnięto przy użyciu sieci MLP oraz przy rozmiarze sieci Kohonena wynoszącym 5x5. Detekcja nie płynności była na poziomie 76%. W 2013 [39] autorzy zastosowali podobną metodologię do detekcji przedłużeń – wprowadzono tylko jedną różnicę. Po redukcji parametrów siecią Kohonena, a przed zastosowanie klasyfikatora, dane

były podzielone na 400ms odcinki, które manualnie zostały oznaczone jako płynne/niepłynne. W ten sposób otrzymano 202 wypowiedzi płynne i 140 niepłynne. Najlepsze rezultaty uzyskano przy użyciu klasyfikatorów sieci MLP i RBF oraz z zastosowaniem sieci Kohonena o rozmiarze  $7 \times 7$  – powyżej 75%.

W 2012 roku Ooi Chia Ai, Hariharan [49] przeprowadzili badania w celu porównania skuteczności rozpoznawania niepłynności pomiędzy parametryzowaniem MFCC (Mel-Frequency Cepstrum Coefficients) a LPCC (Linear Prediction Cepstrum Coefficients). Użyto 39 nagrań z bazy nagrań UCLASS, z których manualnie wydzielono przedłużenia oraz powtórzenia. Z danych tych uzyskano 25 współczynników MFCC oraz 21 współczynników LPCC. Użyto dwóch klasyfikatorów: kNN (k-Nearest Neighbour) oraz LDA (Linear Discriminant Analysis). Klasyfikator kNN działający na zasadzie uczenia z nadzorem, wymagał wydzielenia zbiorów (nagrań) testujących oraz treningowych, a następnie każde nagranie testujące było porównywane ze wszystkimi nagraniami treningowymi. W eksperymentach liczba zbiorów (środków  $k$ ) była dobierana różnie (od 1 do 10), a dla każdego  $k$  wykonywanych było 10 eksperymentów za każdym razem inaczej przydzielając nagrania do zbiorów testujących i treningowych. W przypadku LDA dane również musiały być podzielone na dwie grupy: treningowe i testujące. Najlepsze wyniki rozpoznawania (wartość uśredniona dla przedłużeń i powtórzeń łącznie) dla poszczególnych konfiguracji wyniosły: LPCC+kNN 94.5%, MFCC+kNN 92.5%, LPCC+LDA 90%, MFCC+LDA 89%.

Z analizy w/w wymienionej literatury wynika, iż:

- badacze skupiają się na detekcji przedłużeń, powtórzeń głosek oraz powtórzeń sylab (rzadko pojawiają się też blokady oraz wtrącenia),
- do tej pory nikt nie próbował używać algorytmu CWT do parametryzacji sygnału mowy w celu detekcji niepłynności,
- prawie wszyscy korzystają z odsłuchowo segmentowanych danych wejściowych – jedynie Suszyński, Kuniszyk-Józkowiak [68] oraz Wiśniewski, Kuniszyk-Józkowiak [88] opracowali algorytmy rozpoznawania przedłużeń w mowie ciągłej.



## 4. Parametryzacja sygnału mowy

Przy tworzeniu procedur rozpoznawania niepełności, autor pracy korzystał ze skalogramów CWT obliczonych dla skal barkowych. W trakcie badań, dla porównania, spektrogramy wyznaczane były również algorytmami DWT, STFT oraz LP.

### 4.1. *Dyskretna transformata Fouriera*

Sygnał dźwiękowy można analizować wieloma metodami. Jedną z najbardziej podstawowych i powszechnie stosowanych jest transformata Fouriera (FT – Fourier transform) [28] [47] [56], za pomocą której uzyskuje się spektrogram, czyli wykres zawierający informacje o częstotliwościach, z jakich składa się badany sygnał. W obliczeniach komputerowych stosuje się jej dyskretny odpowiednik DFT (Discrete Fourier Transform).

Ograniczeniem transformaty Fouriera jest założenie, że dane wejściowe reprezentują sygnał stacjonarny (nie zmieniający się w czasie). Z tego powodu, przy przetwarzaniu sygnału mowy (który jest bardzo zmienny), należy używać algorytmu STFT (Short-Time Fourier Transform). W metodzie tej sygnał dzieli się na małe okna i zakłada się, że sygnał jest w tych oknach niezmienny, stacjonarny (ponieważ mowa ludzka w małych odstępach zmienia się bardzo mało – to założenie jest bliskie prawdy) i dla każdego takiego okna oblicza się transformatę. Niestety STFT, jako że zakłada stacjonarność sygnału w oknie, nie daje informacji, kiedy w danym oknie wystąpiła dana częstotliwość – wiadomo tylko, że wystąpiła. Dodatkowo widmo, czyli wynik transformaty dla jednego okna obrazujący występujące w nim częstotliwości, jest tym dokładniejszy im okno jest większe, zatem dokładność widma i dokładność umiejscowienia wystąpienia danej częstotliwości w czasie (w oknie) są do siebie odwrotnie proporcjonalne. Decydując się na jakąś szerokość okna, musimy dokonać wyboru – czy zależy nam bardziej na dokładności czasowej (wąskie okno) czy częstotliwościowej (szerokie okno).

Dyskretny algorytm STFT definiuje się następująco:

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) W_N^{-kn}, k = 0 \dots N-1$$

$$W_N^{-kn} = e^{-i2\pi \frac{kn}{N}} = \cos(2\pi \frac{kn}{N}) - i \sin(2\pi \frac{kn}{N})$$

wz. 4.1.1

gdzie:

$k$  – indeks kolejnych wartości transformaty,

$n$  – numer próbki sygnału wejściowego w oknie,

$N$  – liczba próbek w oknie,

$x(n)$  – sygnał wejściowy,

$X(k)$  – zespolona,  $k$ -ta wartość transformaty Fouriera.

$k$ -ta wartość odpowiada częstotliwości:

$$F = kF_p / N$$

wz. 4.1.2

gdzie:

$F_p$  – częstotliwość próbkowania.

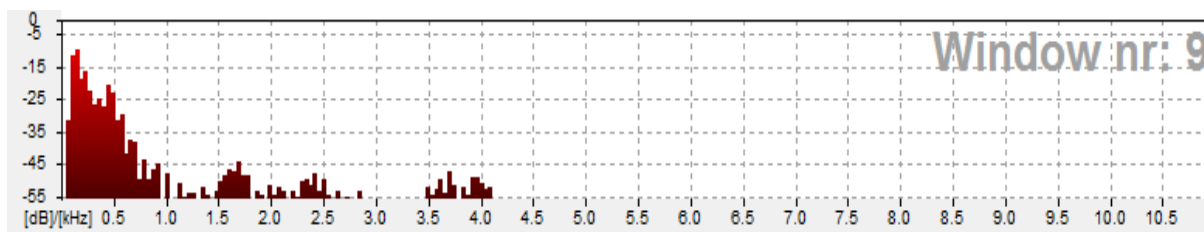
Aby otrzymać widmo należy wyznaczyć moduły ze wszystkich wartości  $X(k)$ . Skalę decybelową otrzymuje się logarytmizując w/w wartości według wzoru (wz. 4.1.3). Ponadto można wszystkie wartości  $X(k)$  podzielić przez maksymalną wartość widma, aby otrzymać znormalizowanie wartości decybelowe – największa wartość widma wynosi 0dB, a wszystkie pozostałe są mniejsze:

$$X' = 20 \log_{10} ( |X(k)| / |maxX| )$$

wz. 4.1.3

gdzie:

$maxX$  – największa wartość modułu widma transformaty Fouriera w całym analizowanym metodą STFT sygnale.



rys. 4.1:1 Przykładowe widmo sygnału uzyskane dyskretną analizą STFT w wybranym oknie czasowym. Zrzut ekranu z programu WaveBlaster.

Aby zobrazować rozkład częstotliwości dla całego sygnału, należy połączyć ze sobą widma kolejnych okien, uzyskując w ten sposób spektrogram. Każdy pionowy pasek (kolumna) spektrogramu przedstawia pojedyncze widmo, takie jak na rys. 4.1:1, na które patrzemy niejako ‘od góry’. Oś pozioma spektrogramu reprezentuje czas, natomiast pionowa częstotliwość. Wartość modułu transformaty jest reprezentowana kolorem na spektrogramie – od białego (małe natężenie), poprzez niebieskie, zielone i żółte aż do czerwonego (wysokie natężenie).



rys. 4.1:2 Spektrogramy uzyskane dyskretną analizą STFT dla szerokości okna (od góry): 23.2ms, 46.4ms, 92.9ms. Zrzut ekranu z programu WaveBlaster.

## 4.2. *Metoda liniowej predykcji*

Bardziej wygładzone widmo, a tym samym łatwiejsze do analizy, można otrzymać metodą liniowej predykcji. Metoda ta opiera się na fakcie, iż kolejne próbki sygnału mowy nie są losowe, tylko zmieniają się w sposób płynny [51] [93]. W związku z tym można aproksymować kolejne próbki poprzednimi:

$$v(n) = \sum_{k=1}^p \alpha_k s(n-k)$$

wz. 4.2.1

gdzie:

$s(n)$  – n-ta próbka sygnału wejściowego,

$v(n)$  – aproksymacja n-tej próbki,

$\alpha_k$  – kolejny k-ty współczynnik predykcji,

$p$  – rząd predykcji.

Celem wyznaczenia współczynników liniowej predykcji jest znalezienie jak najmniejszego błędu predykcji  $E$ :

$$E = \sum_n e^2(n) = \sum_n (s(n) - v(n))^2 = \sum_n \left( s(n) - \sum_{k=1}^p \alpha_k s(n-k) \right)^2$$

wz. 4.2.2

Tak jak w przypadku analizy Fouriera, analizowany sygnał powinien być stacjonarny – zatem tak samo jak w algorytmie STFT, dzielimy sygnał na okna i analizujemy każde okno osobno.

Wartości widma predykcyjnego  $L(f)$  uzyskujemy, korzystając ze wzoru:

$$L(f) = \frac{G}{1 - \sum_{k=1}^p \alpha_k e^{-i2\pi kf/F_s}}$$

wz. 4.2.3

gdzie

$\alpha_k$  – współczynniki predykcji,

$p$  – rząd predykcji,

$i$  – jednostka urojona,

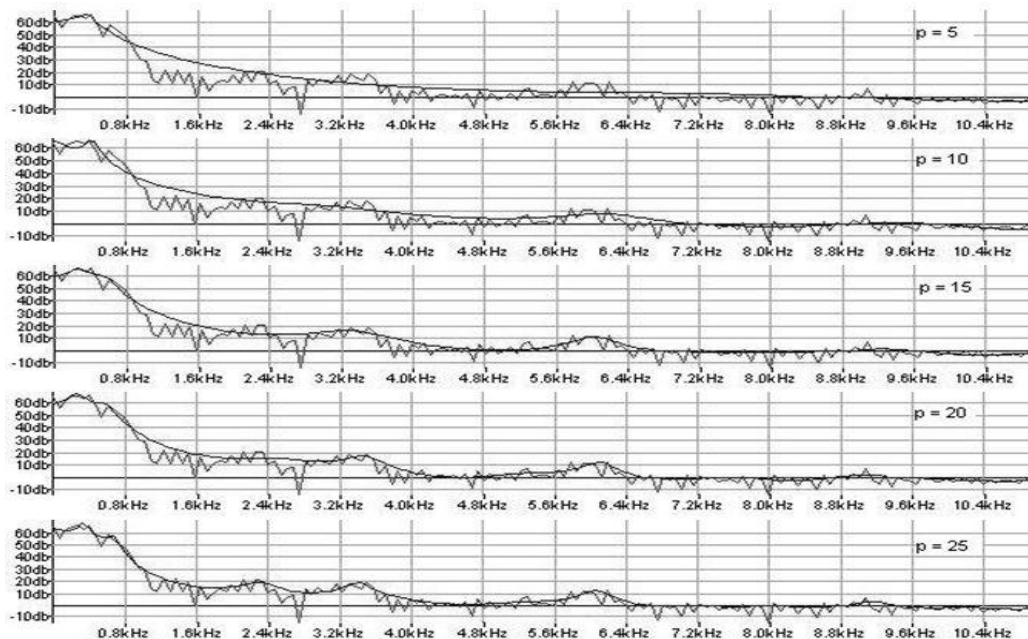
$F_s$  – częstotliwość próbkowania,

$f$  – interesująca nas częstotliwość,

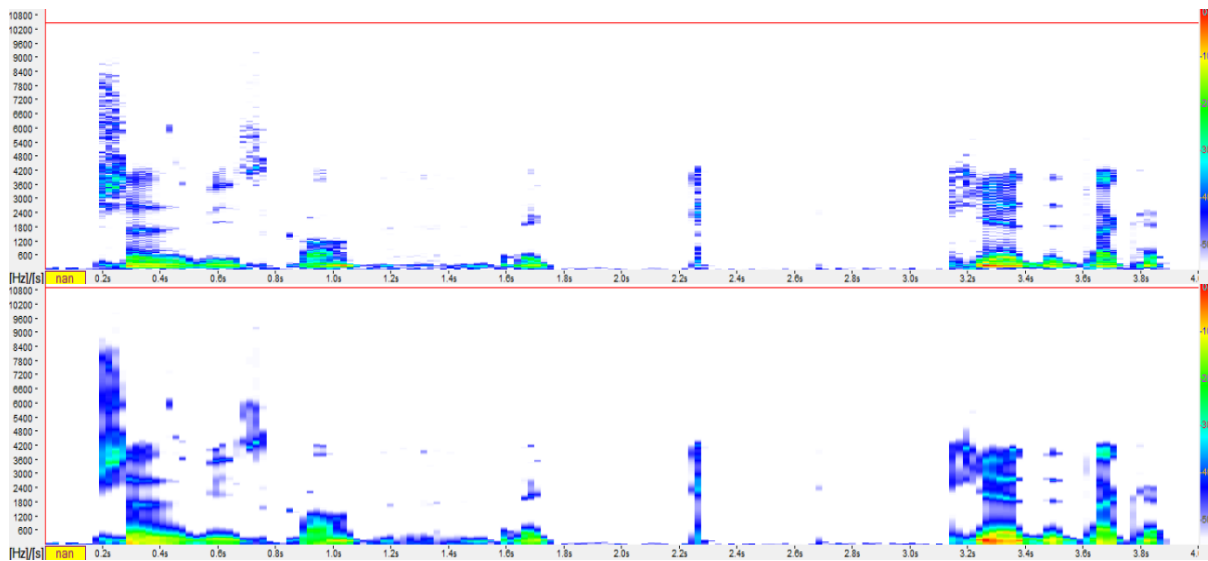
$G$  – współczynnik wzmocnienia równy pierwiastkowi błędowi predykcji ( $G = \sqrt{E}$ ).

Ważnym elementem jest rząd predykcji  $p$ , czyli ilość próbek wykorzystanych w aproksymacji. Jego zwiększenie powoduje lepszą aproksymację sygnału wejściowego współczynnikami LP (czyli zmniejszenie błędu predykcji) kosztem wzrostu liczby obliczeń oraz liczby współczynników LP (więcej danych do przechowywania i ewentualnej

późniejszej analizie). Ponadto zwiększenie rzędu predykcji  $p$  ma wpływ na dokładność widma predykcyjnego, co widać na poniższym rysunku:



rys. 4.2:1 Porównanie widm LP (gładka linia) dla różnego rzędu predykcji  $p$  z naniesionym widmem STFT (linia poszarpana). Zrzut ekranu z programu WaveBlaster.



rys. 4.2:2 Porównanie spektrogramów wypowiedzi „przewisko y k k kiedy tak”. Użyto okna 23ms - u góry STFT, u dołu LP dla rzędu predykcji  $p=25$  (spektrogram jest bardziej wygładzony). Zrzut ekranu z programu WaveBlaster.

### 4.3. Transformata falkowa

Alternatywą dla w/w analiz jest transformata falkowa (WT – Wavelet Transform) [3] [27] [37] [50] [84]. Wyróżnia się jej dwa rodzaje – ciągłą (ang. continuous wavelet transform –

CWT) oraz dyskretną (ang. discrete wavelet transform – DWT). Aby je opisać, najpierw należy przedstawić definicję falki.

Falka to funkcja  $\psi(n)$ , która spełnia następujące warunki:

- jej wartość średnia wynosi 0, czyli

$$\sum_t \psi(n) = 0$$

wz. 4.3.1

gdzie  $n$  – numer próbki.

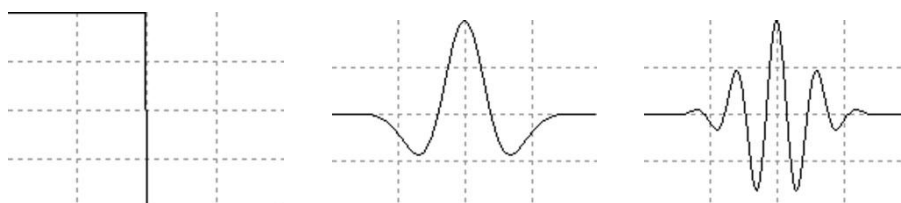
- ma wartości niezerowe tylko na skończonym przedziale  $\langle u, v \rangle$

$$\{\psi(n) \neq 0 : n \in \langle u, v \rangle\}$$

wz. 4.3.2

gdzie  $n$  – numer próbki.

Ponadto falka może zaczynać i kończyć się próbkami o wartości 0, czyli  $\psi(u)$  nie musi być pierwszą niezerową próbką, a  $\psi(v)$  nie musi być ostatnią niezerową próbką.



rys. 4.3:1 Przykładowe falki. Od lewej: Haara, Mexican Hat, Morleta.

Z każdej z falek bazowych tworzy się rodzinę falek względem parametrów  $a$  i  $b$ :

$$\psi_{a,b}(n) = \frac{1}{\sqrt{a}} \psi\left(\frac{n-b}{a}\right)$$

wz. 4.3.3

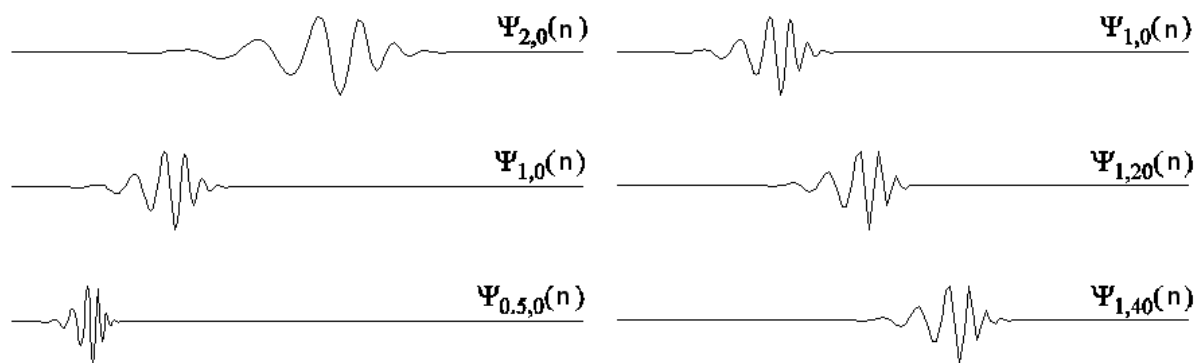
gdzie:

$a$  – skala,

$b$  – przesunięcie,

$n$  – numer próbki (czas),

$\psi(n)$  – falka bazowa.



rys. 4.3:2 Przykładowe funkcje z rodziny falki  $\psi_{a,b}(n)$

Współczynnik  $a$  odpowiada za skalowanie poziome falki (rozciąganie i zwężanie), natomiast współczynnik  $b$  odpowiada za przesunięcie falki. Sposób wykorzystania tak wygenerowanej rodziny falek opisany jest w kolejnych podrozdziałach.

### 4.3.1. Ciągła transformata falkowa

Ciągłą transformatę falkową dla sygnałów dyskretnych można przedstawić wzorem [3] [50] [84]:

$$CWT_{a,b} = \sum_n f(n) \cdot \psi_{a,b}(n)$$

wz. 4.3.1.1

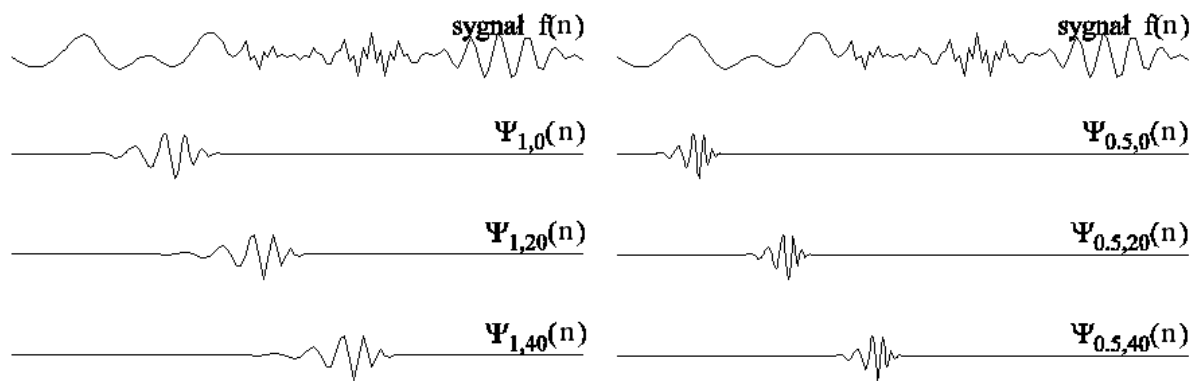
gdzie:

$n$  – numer próbki (czas),

$f(n)$  – sygnał wejściowy,

$\psi_{a,b}(n)$  – falka bazowa w skali  $a$  i przesunięciu  $b$ .

Zasadę działania CWT obrazuje rys. 4.3.1:1. Kolejne falki  $\psi_{a,b}(n)$  wywodzące się z tej samej falki bazowej  $\psi(n)$  (czyli należącej do tej samej rodziny) przemnaża się przez sygnał wejściowy i sumuje. Dla każdej pary  $a, b$  wyliczana jest wartość  $CWT_{a,b}$ . Na rysunku rys. 4.3.1:1 krok skali (parametr  $a$ ) maleje wykładniczo ( $2^x$ ), natomiast przesunięcie (parametr  $b$ ) rośnie liniowo (krok=20). Oczywiście liczby  $a$  i  $b$  można wybierać dowolnie i dlatego tę transformatę nazywamy ciągłą.



rys. 4.3.1:1 Schemat obliczania CWT

W pracy przyjęto, że przeciwne wartości  $CWT_{a,b}$  odzwierciedlają ten sam stopień podobieństwa (czyli wartość 2 i -2 oznaczają takie samo podobieństwo). W związku z tym do wszystkich obliczeń brane są wartości  $|CWT_{a,b}|$ . Ponadto wszędzie użyto relatywnej skali decybelowej, gdzie największej wartości skalogramu  $CWT_{MAX}$  przyporządkowano wartość 0dB, a pozostałe wartości decybelowe są ujemne:

$$CWT_{a,b}^0 = 20 \log_{10} \left( \frac{|CWT_{a,b}|}{|CWT_{MAX}|} \right)$$

wz. 4.3.1.2

gdzie:

$CWT_{MAX}$  – największa wartość CWT dla wszystkich skal  $a$  i przesunięć  $b$ .

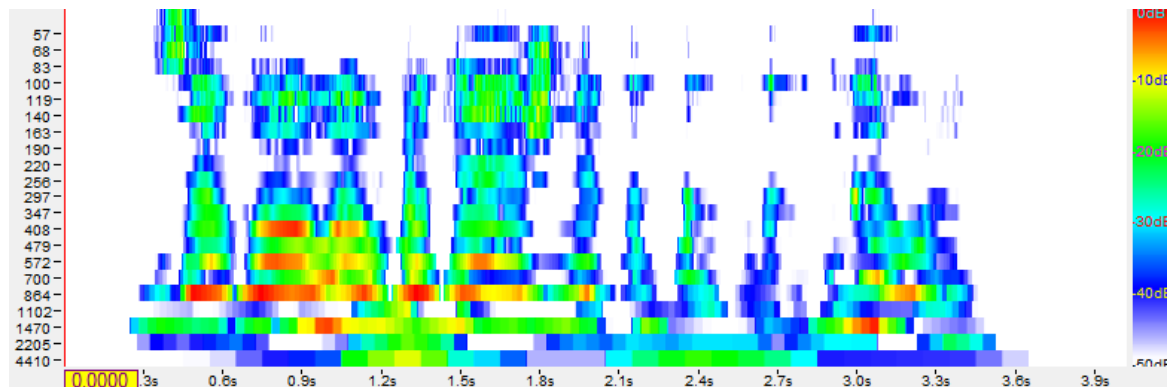
Należy zwrócić uwagę na sytuacje brzegowe. Ponieważ sygnał wejściowy jest dyskretny – falka musi składać się co najmniej z dwóch próbek. Jeżeli składałaby się z jednej, to zgodnie ze wz. 4.1.1 musiałaby wynosić 0, a zatem nie spełniałaby wz. 4.1.2. Falki nie możemy też rozszerzać w nieskończoność, jej długość nie powinna przekraczać długości sygnału. Po drugie sygnał wejściowy jest skończony, a zatem przesuwając falkę w prawo (tak jak na rys. 4.3.1:1) wyjdzie ona poza zakres sygnału wejściowego. Możemy temu zaradzić na kilka sposobów:

- uzupełnić sygnał wejściowy zerami –  $\dots, s_{n-3}, s_{n-2}, s_{n-1}, 0, 0, 0, \dots$
- uzupełnić sygnał wejściowy lustrzanym odbiciem jego końca –  $\dots, s_{n-3}, s_{n-2}, s_{n-1}, s_{n-1}, s_{n-2}, s_{n-3}, \dots$
- uzupełnić sygnał wejściowy cyklicznie, czyli próbkami z początku sygnału –  $\dots, s_{n-3}, s_{n-2}, s_{n-1}, s_0, s_1, s_2, \dots$



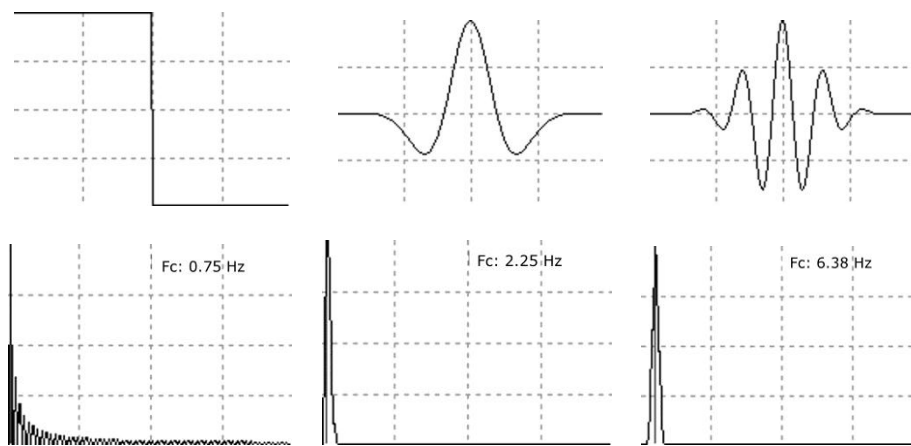
Oczywiście parametr  $b$  może być co najwyżej równy długości sygnału wejściowego pomniejszonego o jedną próbkę.

Oto przykładowy skalogram, czyli zbiór współczynników  $CWT_{a,b}^0$ :



rys. 4.3.1:2 Skalogram  $CWT^0$  wypowiedzi „zawołał potężny k k k król”. Parametr  $b$  (oś pozioma), który jest wprost proporcjonalny do czasu, został zamieniony na sekundy. Parametr  $a$  znajduje się na osi pionowej. Kolory obrazują wartości współczynników CWT wyrażone w dB. Zrzut ekranu z programu WaveBlaster.

Wynik analizy CWT jest charakterystyką czasowo-skalową, czyli przedstawia ona zmiany wartości CWT dla wybranych skal w czasie. Każda pozioma linia (odpowiadającej konkretnej skali  $a$ ) na rys. 4.3.1:2 przedstawia zmiany podobieństwa sygnału wejściowego do falki  $\psi_{a,b}(n)$ . Im wartość współczynnika  $CWT_{a,b}^0$  jest większa (oznaczona kolorem białym najmniejsza wartość, czerwonym największa), tym podobieństwo (tj. wartość bezwzględna iloczynu – wz. 4.3.1.1) jest większe. Aby uzyskać spektrogram należy wartości skal zamienić na odpowiadające im częstotliwości. Niestety odwzorowanie takie nie jest dokładne, ponieważ falka zamiast jednej częstotliwości, reprezentuje całą ich grupę (co widać na rysunku rys. 4.3.1:3). Dokonuje się więc przybliżania, reprezentując cały zbiór częstotliwości falki  $\psi(n)$  poprzez jedną centralną częstotliwość  $F_C$ . Za  $F_C$  przyjmuje się częstotliwość, dla której widmo falki przyjmuje wartość maksymalną



rys. 4.3.1.3 Falki Haara, Mexican Hat, Morleta oraz odpowiadające im widma uzyskane algorytmem transformaty Fouriera

Daną skalę  $a$  falki  $\psi(n)$  można zamienić na odpowiadającą jej pseudo-częstotliwość według wzoru:

$$F = \frac{F_c \cdot d_s}{d}$$

wz. 4.3.1.3

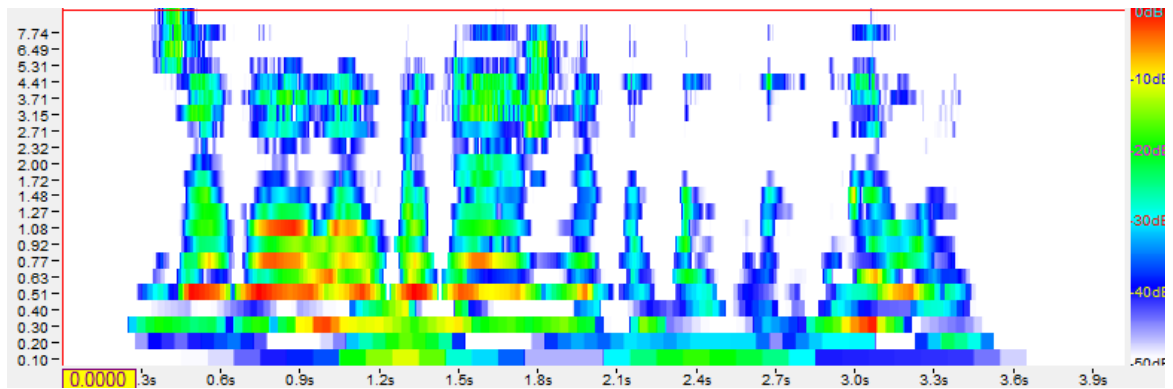
gdzie:

$F_c$  – częstotliwość środkowa falki  $\psi(n)$ ,

$d_s$  – liczba próbek zawarta w jednej sekundzie sygnału (wynika ona wprost z częstotliwość próbkowania sygnału wejściowego),

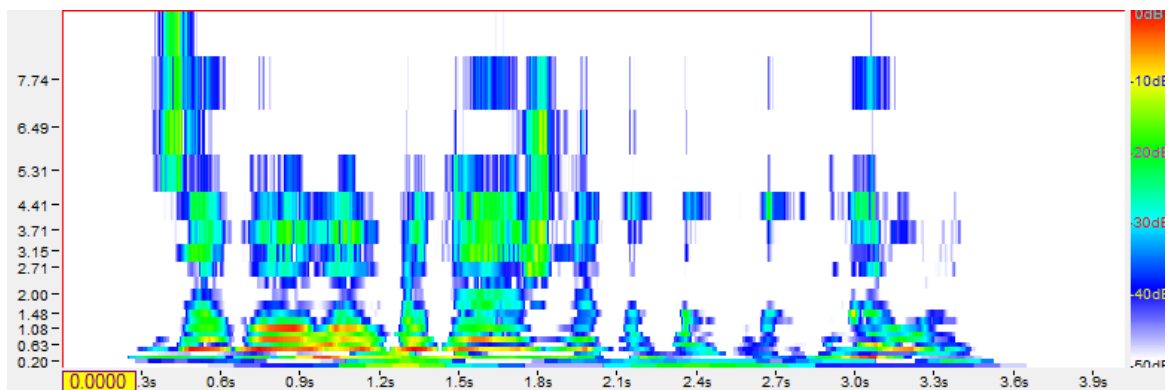
$d$  – szerokość falki w próbkach odpowiadającej danej skali  $a$ , czyli liczba próbek na jakiej reprezentowana jest falka w danej skali  $a$ .

Przewartościowana oś pionowa ze skal  $a$  na  $kHz$  jest zaprezentowana na poniższym rysunku:



rys. 4.3.1:4 Skalogram  $CWT^o$  wypowiedzi „zawolał potężny k k k król”.. Kolejne numery skal zostały zamienione na pseudo-częstotliwości w kHz (oś pionowana). Zrzut ekranu z programu WaveBlaster.

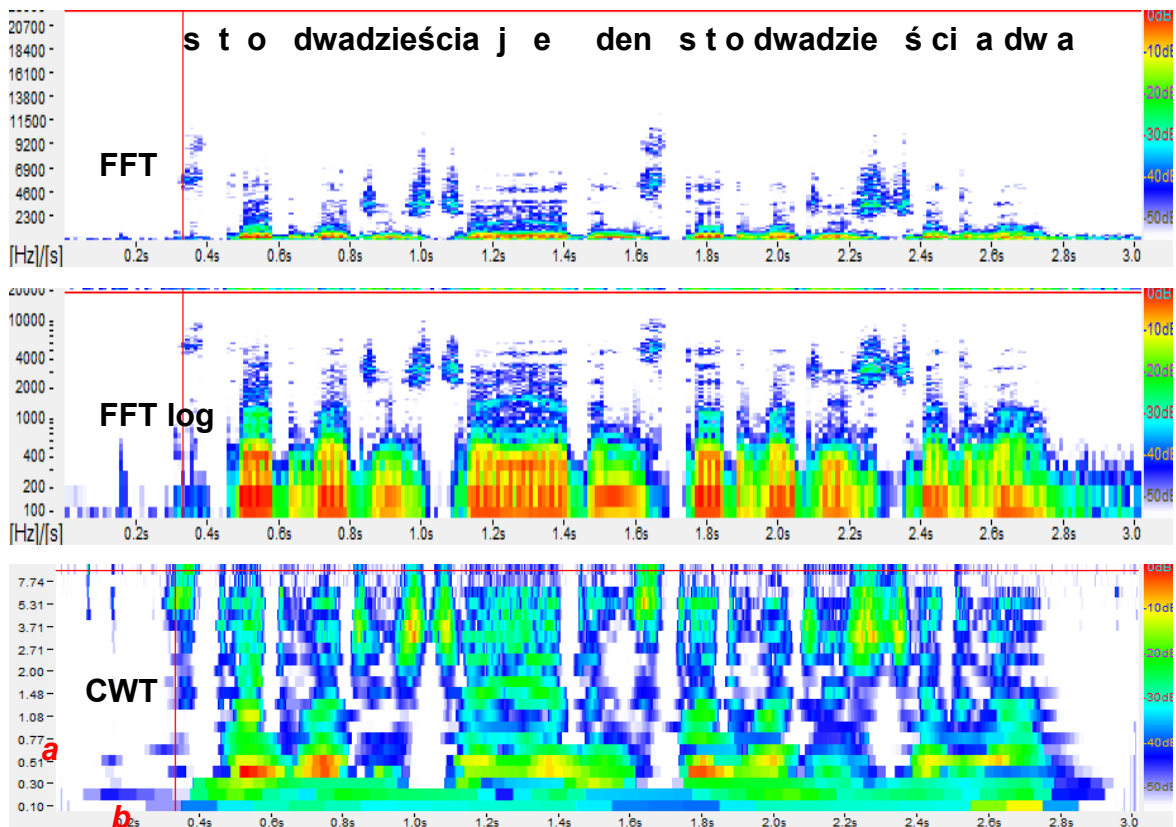
Skale  $a$  na osi pionowej zmieniały się liniowo (rys. 4.3.1:2), natomiast pseudo-częstotliwości nie (rys. 4.3.1:4), więc jest to skalogram. Jeżeli chcemy uzyskać spektrogram, należy przekształcić wykres tak, aby oś pozioma była liniowa względem częstotliwości.



rys. 4.3.1:5 Spektrogram powstały z przekształcenia rys. 4.3.1:4 tak, aby częstotliwości zmieniały się liniowo. Zrzut ekranu z programu WaveBlaster.

Jak widać transformata falkowa nie ustala na sztywno rozdzielczości czasowo-częstotliwościowej, jak to ma miejsce w transformacie Fouriera. Dla częstotliwości wysokich, ponieważ sygnał zmienia się tam szybko, można położyć nacisk na lokalizację w czasie (oś pozioma będzie wtedy gęsta). Dla niskich częstotliwości czas jest już mniej istotny, zatem rozdzielczość czasową można zmniejszyć. Oczywiście rozdzielczość czasowa i częstotliwościowa może wszędzie być wysoka – osiągamy to niestety dużym nakładem obliczeniowym.

Jak widzimy na rys. 4.3.1:6, widma CWT i FFT (Fast Fourier Transform) istotnie się różnią, chociaż wyraźnie widać, że są analizą tego samego sygnału. Duże znaczenie na kształt skalogramu CWT ma dobór falki macierzystej.



rys. 4.3.1:6 Porównanie widm 3-sekundowej wypowiedzi „Sto dwadzieścia jeden, sto dwadzieścia dwa”. Od góry: FFT, FFT z częstotliwością w skali logarytmicznej, CWT<sup>0</sup>. W przypadku CWT<sup>0</sup> użyto falki opisanej w rozdziale 4.2 dla skal barkowych. Zrzut ekranu z programu WaveBlaster.

### 4.3.2. Dyskretna transformata falkowa

Istnieje grupa falek, których wykresy tworzy się ze skończonego zbioru liczb – zwanych współczynnikami falkowymi  $h(k)$ . Z nich otrzymujemy współczynniki skalujące  $g(k)$  [1] [27] [37] [50] [84].

$$g(k) = (-1)^k \cdot h(N - 1 - k), \quad k = 0 \dots K - 1$$

wz. 4.3.2.1

gdzie:

$K$  – liczba wszystkich współczynników

Następnie korzystając z rekurencyjnych wzorów możemy uzyskać funkcję falkową  $\psi(n)$  i skalującą  $\varphi(n)$  z dowolną dokładnością, korzystając z rekurencyjnych wzorów:

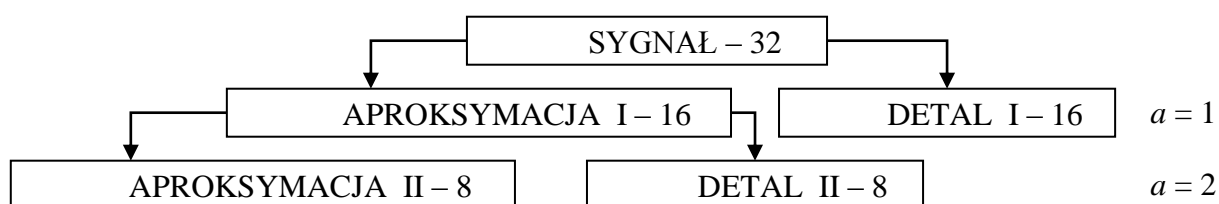
$$\psi(n) = \sqrt{2} \sum_{k=0}^{K-1} h(k) \varphi(2n - k) \quad \varphi(n) = \sqrt{2} \sum_{k=0}^{K-1} g(k) \varphi(2n - k)$$

wz. 4.3.2.2

tab. 4.3.2-1 Współczynniki falkowe przykładowych falek dyskretnych wraz wykresami falek, które reprezentują

Nazwa falki	$h(k)$	$\psi(n)$	$\varphi(n)$
Daubechies 2	0,48296291314453, 0,83651630373781, 0,22414386804201, -0,12940952255126		
Symmlet 4	0.03222310060408, -0.01260396726205, -0.09921954357696, 0.29785779560561, 0.80373875180680, 0.49761866763256, -0.02963552764603, -0.07576571478936		

Algorytm DWT ma zastosowanie tylko dla falek dyskretnych. Jest on dużo szybszy od algorytmu CWT, ale dużo mniej dokładny. Polega on na rozkładaniu sygnału na grupę wartości opisującą ogół (aproksymację) i grupę opisującą szczegóły (detal). Liczba wartości w każdej z grup jest o połowę mniejsza od ilości próbek sygnału wejściowego. Wartości należące do tych grup reprezentują nam jedną skalę  $a$  widma DWT. Operację powtarza się, przyjmując najczęściej za dane wejściowe wartości aproksymacji (można również detalu). W ten sposób otrzymamy kolejną skalę  $a+1$ , która będzie miała dwa razy mniej wartości. Dopóki liczba danych wejściowych jest większa od 1 możemy obliczać kolejne skale. Ponieważ w każdej skali mamy dwa razy mniej próbek zmiana skali o 1 dla DWT jest równoznaczna z podwojeniem skali dla CWT. Zatem w obu transformatach, CWT i DWT, skale  $a$  oznaczają co innego, ale są ze sobą związane prostą zależnością.



rys. 4.3.2:1 Schemat podziału na aproksymację i detal w algorytmie DWT

Schemat algorytmu jest następujący: dla uproszczenia założmy, że sygnał wejściowy składa się z 8 próbek  $I_0, \dots, I_7$  oraz, że rozkładamy tylko wartości aproksymacji. Mając współczynniki falkowe  $H_0, H_1, H_2, H_3$  oraz współczynniki skalujące  $G_0, G_1, G_2, G_3$  należy przemnożyć sygnał przez odpowiednią macierz:

$$\begin{bmatrix} L0 \\ H0 \\ L1 \\ H1 \\ L2 \\ H2 \\ L3 \\ H3 \end{bmatrix} = \begin{bmatrix} g0 & g1 & g2 & g3 & & & & \\ h0 & h1 & h2 & h3 & & & & \\ & & g0 & g1 & g2 & g3 & & \\ & & h0 & h1 & h2 & h3 & & \\ & & & & g0 & g1 & g2 & g3 \\ & & & & h0 & h1 & h2 & h3 \\ g2 & g3 & & & & & g0 & g1 \\ h2 & h3 & & & & & h0 & h1 \end{bmatrix} \cdot \begin{bmatrix} I0 \\ I1 \\ I2 \\ I3 \\ I4 \\ I5 \\ I6 \\ I7 \end{bmatrix}$$

wz. 4.3.2.3

gdzie:

$Lx$  – wartości dolno-przepustowe, czyli aproksymacja,

$Hx$  – wartości górno-przepustowe, czyli detal,

$Ix$  – wartości sygnału wejściowego,

$gx$  – współczynniki skalujące,

$hx$  – współczynniki falkowe.

Następnie należy przesortować tablicę wynikową (najpierw wartości  $Lx$ , a potem  $Hx$ ) i powtórzyć czynność dla wartości  $Lx$ , dopóki jest ich więcej niż jedna, a jako ostateczny wynik przyjąć ostatnią wartość aproksymacji i wszystkie wartości detalu.

tab. 4.3.2-2 Schemat wyznaczania ostatecznego wyniku DWT na podstawie wartości aproksymacji i detalu

<b>I0</b>	<b>I1</b>	<b>I2</b>	<b>I3</b>	<b>I4</b>	<b>I5</b>	<b>I6</b>	<b>I7</b>
<b>1L0</b>	1L1	1L2	1L3	1H0	1H1	1H2	1H3
<b>2L0</b>	2L1	2H0	2H1				
<b>3L0</b>	3H0						
<b>3L0</b>	3H0	2H0	2H1	1H0	1H1	1H2	1H3

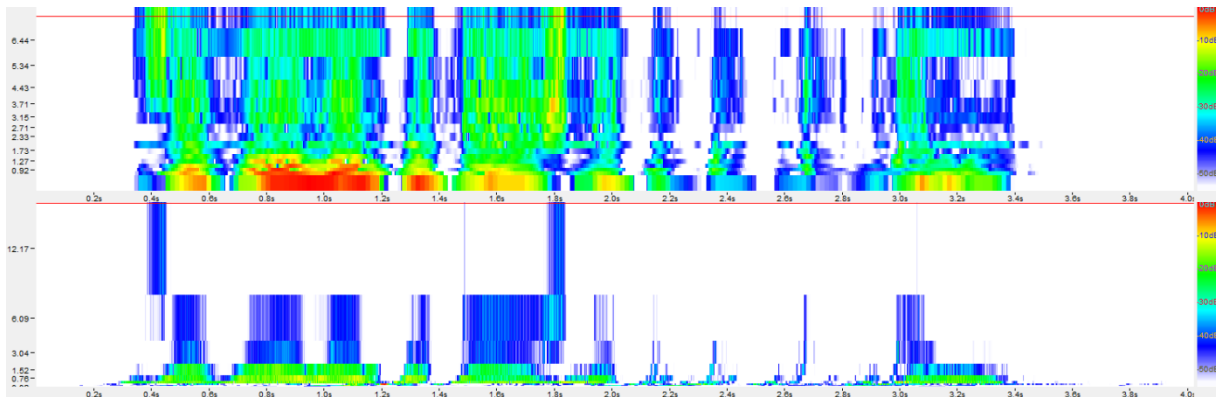
Rozmieszczenie wartości aproksymacji i detali na spektrogramie DWT oraz porównanie go z przykładowym spektrogramem CWT przedstawia poniższy schemat.

I0	I1	I2	I3	I4	I5	I6	I7
1H0	1H1	1H2	1H2				
2H0		2H1					
3H0							
3L0							

rys. 4.3.2:2 Schemat rozmieszczenia wartości DWT w oknie czasowym o szerokości 8 próbek

I0	I1	I2	I3	I4	I5	I6	I7
CWT 1,0	CWT 1,1	CWT 1,2	CWT 1,3	CWT 1,4	CWT 1,5	CWT 1,6	CWT 1,7
CWT 2,0	CWT 2,1	CWT 2,2	CWT 2,3	CWT 2,4	CWT 2,5	CWT 2,6	CWT 2,7
CWT 3,0	CWT 3,1	CWT 3,2	CWT 3,3	CWT 3,4	CWT 3,5	CWT 3,6	CWT 3,7
CWT 4,0	CWT 4,1	CWT 4,2	CWT 4,3	CWT 4,4	CWT 4,5	CWT 4,6	CWT 4,7

rys. 4.3.2:3 Schemat rozmieszczenia wartości CWT w oknie czasowym o szerokości 8 próbek



rys. 4.3.2:4 Spektrogram CWT (u góry) i DWT (u dołu) wypowiedzi „potężny k k k król”. DWT wyliczono na podstawie współczynników falkowych falki Daubechie18. CWT wyliczono na podstawie falki Daubechie18 dla okna 23ms (100% przesunięcia).

#### 4.4. Konstrukcja falki macierzystej

Ze względu na duże ograniczenia doboru falki i wyboru rozdzielczości czasowo-częstotliwościowej w DWT, autor pracy skupił się na analizie CWT, jako metodzie parametryzacji danych.

Na falkę macierzystą wybrano falkę powstałą na bazie rzeczywistej falki Morleta [29], ze względu na jej kształt (rys. 4.4:1), który jest podobny do przebiegu sygnału mowy:

$$\psi^*(t) = e^{-t^2/2} \cos(2\pi F_C t)$$

wz. 4.4.1

gdzie:

$F_C$  – częstotliwość środkowa falki,

$t$  – czas.

W literaturze można znaleźć również znormalizowaną postać tej falki [44] [55] – jest ona przemnożona przez stałe  $\frac{1}{4\sqrt{\pi}}$  lub  $\frac{1}{\sqrt{b\pi}}$  ( $b$  – współczynnik szerokości pasma). Można je jednak pominąć - ponieważ autor pracy stosuje w skalogramach CWT relatywną skalę decybelową

(wz. 4.2.3), z której wynika, że dowolna stała przemnażająca falkę zostanie skrócona w wyniku dzielenia  $\frac{|CWT_{a,b}|}{|CWT_{MAX}|}$ .

Falka ta ma jeszcze jedną bardzo istotną zaletę – może mieć różną częstotliwość środkową  $F_C$  – co wynika wprost z wz. 4.4.1, gdzie  $F_C$  jest parametrem funkcji cosinus. Jako, że badacze podają różne dolne zakresy częstotliwości słyszalnych przez człowieka (od 15 do 20 Hz), za  $F_C$  przyjęto wartość 20Hz. Na potrzeby tej rozprawy nazwiemy tę falkę Morlet20.



rys. 4.4.1 Falka Morleta20 (po lewej) oraz jej widmo Fouriera (po prawej)

Aby zastosować wzór falki Morleta20 do zadań dyskretnych, za przedział niezerowych wartości falki (wz. 4.3.2) przyjęto zbiór  $t \in \langle -4,4 \rangle$ . Aby utworzyć falkę matkę – należy wygenerować ją na zbiorze punktów odpowiadających jednej sekundzie. W tym celu należy ustalić częstotliwości próbkowania – w tej pracy zastosowano  $F_C=22050\text{Hz}$ , ponieważ z taką częstotliwością zostały nagrane wszystkie analizowane fragmenty mowy. Następnie należy spróbkować falkę w  $N$  równomiernie rozłożonych punktach na osi  $x$  (gdzie  $N = F_C$ ), zatem otrzymujemy zależność:

$$\psi(n) = \psi^* \left( -4 + n \cdot \frac{8}{N-1} \right)$$

wz. 4.4.2

gdzie:

$\psi^*(t)$  – ciągła reprezentacja falki Morlet20,

$\psi(n)$  – dyskretna reprezentacja falki Morlet20,

$n$  – kolejne indeksy próbek  $0 \dots (N-1)$ ,

$N$  – liczba próbek (dla falki matki  $N = F_C$ ).

Dla uproszczenia oznaczeń, autor pracy za numer skali  $a$  przyjął liczbę próbek, na których falka jest reprezentowana. Zatem falce matce Morlet20 zostanie przyporządkowana wartość  $a=22050$ . Falkę o podwojonej częstotliwości = 40Hz uzyskujemy dla skali  $a=11025$  poprzez zastosowanie wz. 4.4.2. dla  $N=11025$ , itd. – czyli



falkę dla dowolnej skali  $a$  uzyskujemy stosując wzór wz. 4.4.2. dla  $N=a$ . Jej częstotliwość  $F_a$  można wyznaczyć ze wzoru:

$$F_a = F_c d_s / a$$

wz. 4.4.3

gdzie:

$a$  – numer skali,

$F_c$  – częstotliwość środkowa falki *Morlet20*,

$d_s$  – liczba próbek zawarta w jednej sekundzie sygnału (wynika ona wprost z częstotliwość próbkowania sygnału wejściowego).

Każdą falkę, z tak wygenerowanej rodziny, musimy znormalizować względem średniej wartości oraz energii:

- Zgodnie ze wz. 4.3.1, wartość średnia falki musi wynosić 0. Aby ten warunek spełnić, musimy obliczyć sumę jej wartości:

$$\sigma = \sum_{n=0}^{N-1} \psi(n)$$

wz. 4.4.4

musimy tę sumę uśrednić:

$$\Delta = \frac{\sigma}{N}$$

wz. 4.4.5

a następnie musimy ją odjąć od wartości każdej próbki:

$$\overline{\psi(n)} = \psi(n) - \Delta$$

wz. 4.4.6

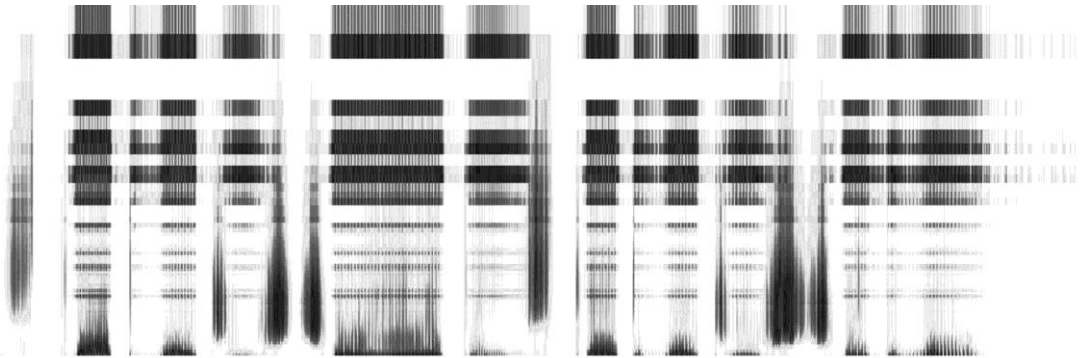
- Wszystkie falki powinny mieć taką samą energię

$$E = \sum_{n=0}^{N-1} \psi(n)^2$$

wz. 4.4.7

Jak już wspomniano wcześniej w tym rozdziale, ponieważ autor pracy stosuje w skalogramach CWT relatywną skalę decybelową, dowolna stała przemnażająca

falkę zostanie skrócona w wyniku dzielenia  $\frac{|CWT_{a,b}|}{|CWT_{MAX}|}$ . Wartość energii  $E$  (autor pracy przyjmuje  $E=I$ ) nie ma więc znaczenia, ważne aby dla wszystkich falek była taka sama, w przeciwnym wypadku falki o większej energii wygenerują zawyżone wartości współczynników  $CWT$  (rys. 4.4:2)



rys. 4.4:2 Skalogram CWT przykładowego 3-sekundowego sygnału mowy ze źle unormowanymi skalami. Wyraźnie widać zawyżone wartości (ciemniejsze pasy) dla skal o zbyt dużej energii.

We wzorze falki czynnikiem normalizacyjnym jest wartość  $\frac{1}{\sqrt{a}}$  (wz. 4.3.3), jednak w przypadku opisaney wcześniej generacji falek w reprezentacji dyskretnej, może być to normalizacja niedokładna – energie poszczególnych falek nie zawsze będą równe dokładnie 1. W tym celu dla każdej falki obliczana jest energia zgodnie ze wz. 4.4.7, a następnie wszystkie wartości falki przemnażane są według reguły:

$$\overline{\overline{\psi(n)}} = \frac{1}{\sqrt{E}} \cdot \overline{\psi(n)}$$

wz. 4.4.8

gdzie:

$n$  – kolejne indeksy próbek falki,

$\overline{\psi(n)}$  – uprzednio znormalizowana falka względem wartości średniej (wz. 4.4.6),

$E$  – energia falki znormalizowanej wg. wz. 4.4.6,

$\overline{\overline{\psi(n)}}$  – falka dwukrotnie znormalizowana – względem średniej wartości, a następnie względem energii.

#### 4.5. *Skala barkowa*

CWT, z racji tego, że jest ciągłą transformatą, może być wyliczona dla dowolnych skal  $a$ . Jednak wyliczanie ich wszystkich wydaje się nadmiarowe i na pewno wymagające

obliczeniowo – ze względu na samo CWT jak i na późniejsze jego przetwarzanie (skalogram składałby się z ogromnej liczby danych). Uznano, iż rozpoznawanie niepłynności na bazie sygnału sparametryzowanego przy zastosowaniu skali percepcyjnej, czyli bazującej na charakterystyce słuchu człowieka, powinno przynieść zadowalające rezultaty. Z tego powodu zdecydowano się na wybór skali barkowej, jednej z kilku popularnych skal percepcyjnych (takich jak skala melowa czy ERB [57]).

Wybrano reprezentację Hartmута [85]:

$$B = \frac{26.81}{1 + 1960/f} - 0.53$$

wz. 4.5.1

gdzie:

$f$  – częstotliwość w Hz.

Zestawienie skal wygląda następująco (przesunięcie  $b$  wynosi 50% długości falki):

tab. 4.5-1 Zestawienie skal dla  $a$  falki Morlet20 oraz odpowiadające im przesunięcia  $b$ , częstotliwości  $F_a$  i skale barkowe  $B$ . Częstotliwość próbkowania wynosiła 22050 Hz

$a$ [skala]	$b$ [ms]	$F_a$ [Hz]	$B$ [bark]
43	1,0	10317	22
55	1,2	7992	21
69	1,6	6407	20
84	1,9	5258	19
101	2,3	4386	18
119	2,7	3702	17
140	3,2	3152	16
163	3,7	2698	15
190	4,3	2319	14
221	5,0	1997	13
256	5,8	1720	12
298	6,8	1479	11
348	7,9	1268	10
408	9,3	1081	9
482	10,9	915	8
576	13,1	765	7
699	15,9	631	6
866	19,6	509	5
1107	25,1	399	4
1484	33,7	297	3
2159	49	204	2
3718	84,3	119	1

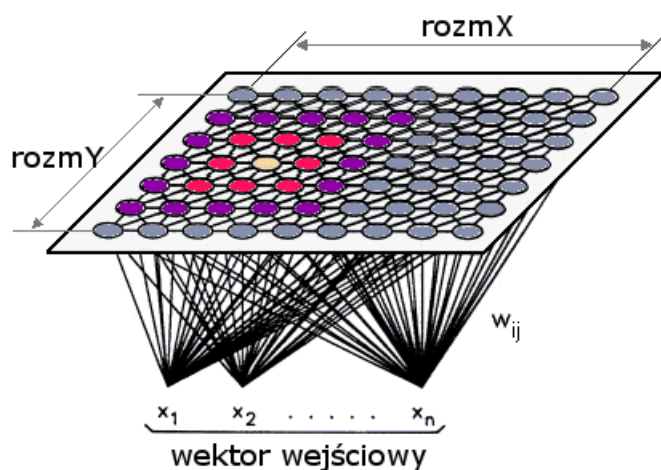
## 5. Grupowanie i klasyfikacja danych z zastosowaniem sztucznych sieci neuronowych

### sieci neuronowych

Surowy wynik CWT (na przykład na rys. 4.3.1:2) obliczony dla skal barkowych, jest punktem wyjścia dla wszystkich użytych metod detekcji niepełności mowy prezentowanych w tej pracy. Oznacza to, że wszystkie algorytmy post-procesingu oczekują skalogramu, jako danych wejściowych. W przypadku przedłużeń i powtórzeń głosek zastosowano sieci Kohonena do grupowania wektorów, a w przypadku powtórzeń głosek dodatkowo użyto perceptronu 3-warstwowego przy klasyfikacji fragmentów do zbiorów płynny/niepłynny.

#### 5.1. Sieci Kohonena

Sieć Kohonena (samoorganizująca mapa – ang. „self-organizing map” – SOM) [26] [30] [41] [74] [75] [76] [77] została zaproponowana przez Teuvo Kohonena. Podstawową ideą konstrukcji sieci Kohonena jest struktura połączonych elementów, zwanych neuronami, które konkurują o wektory wejściowe. Struktura sieci może być dowolna, ale najczęściej stosowane są mapy prostokątne.



rys. 5.1:1 Przykładowa mapa Kohonena z wygrzającym jednym neuronem i zaznaczonym jego sąsiedztwem

Rysunek rys. 5.1:1 prezentuje:

- $rozmX * rozmY = K$  – liczba neuronów w sieci,
- $n$  – wymiar wektora wejściowego,
- każdy wejściowy wektor  $\vec{X}$  posiada  $n$  składowych:  $\vec{X} = \{x_1, x_2, \dots, x_n\}$ ,

- każdy neuron  $y_i$  ma dokładnie  $n$  połączeń, każde powiązane z kolejnym elementem  $x_j$  wektora  $\vec{X}$ ,
- każda składowa  $x_j$  jest połączona z  $K$  neuronami (czyli ze wszystkimi), zatem mamy  $K \cdot n$  połączeń. Każde połączenie jest reprezentowane przez jego wagę  $w_{ji}$ ,  $i=1..n$ ,  $j=1..K$ .

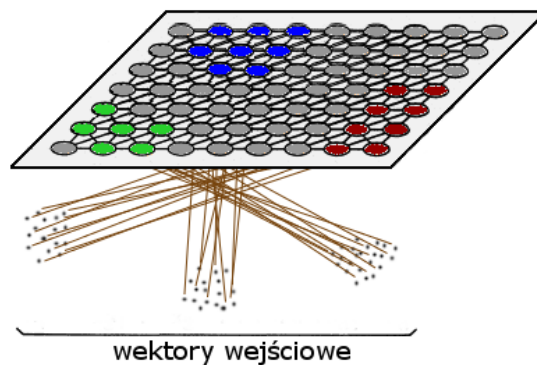
Każdy neuron o indeksie  $j$  jest definiowany przez wektor  $\vec{W}_j = w_{ji}$ ,  $i=1..n$ , którego składowe są zmieniane podczas treningu.

Podstawowy algorytm uczenia jest następujący:

1)	zainicjuj wszystkie wagi sieci wartościami losowymi z przedziału $\langle 0,1 \rangle$ oraz znormalizuj wszystkie wektory wejściowe również do przedziału $\langle 0,1 \rangle$
	<i>powtarzaj poniższe kroki ustaloną liczbę razy (każdy przebieg nazywamy epoką)</i>
2)	dla każdego wektora $\vec{X}$ ze zbioru wejściowego (można brać wektory według jakiejś kolejności lub losowo)
3)	znajdź wektor wygrywający $\vec{W}_j$ (przyporządkowany do $j$ -tego neuronu $n_j$ ), który jest najbliższym danemu wektorowi wejściowemu $\vec{X}$ (i.e. dystans pomiędzy $\vec{W}_j$ a $\vec{X}$ jest minimalny). Metryka może być dowolna, ale zazwyczaj stosuje się Euklidesową, gdzie dystans $d_j$ pomiędzy wektorem wejściowym $\vec{X}$ i $j$ -tym neuronem definiuje się wzorem: <div style="text-align: center;"> <math display="block">d_j = \sqrt{\sum_{i=1}^n (x_i - w_{ji})^2},</math> </div> <div style="text-align: right;">wz. 5.1.1</div>
4)	dla wszystkich neuronów $n_i$ sieci (wygrywający neuron oznaczony jest indeksem $j$ )
5)	zmień wagi wektora $\vec{W}_i$ (przyporządkowane neuronowi $n_i$ ) aby zbliżyły się do wartości wektora wejściowego $\vec{X}$ : <div style="text-align: center;"> <math display="block">\vec{W}_i = \vec{W}_i + s(i, j)\alpha(\vec{X} - \vec{W}_i)</math> </div> <div style="text-align: right;">wz. 5.1.2</div> <p>gdzie:</p> <p><math>\alpha</math> – jest współczynnikiem uczenia. Im większy, tym wagi neuronów szybciej</p>

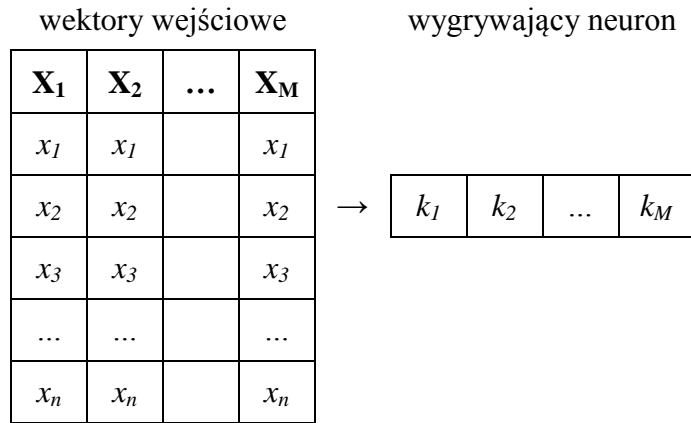
będą się zbliżały do wartości wektorów wejściowych,  
 $j$  – indeks neuronu zwycięskiego,  
 $i$  – indeks neuronu,  
 $s(i,j)$  – jest funkcją sąsiedztwa zwracającą wartość z przedziału  $\langle 0,1 \rangle$  według wybranej metryki na podstawie odległości w topologii sieci Kohonena między neuronem  $j$ -tym (zwycięskim) a  $i$ -tym. Dla indeksu  $i=j$  funkcja powinna zwracać wartość możliwie największą, a dla indeksów coraz bardziej oddalonych od neuronu zwycięskiego powinna zwracać wartości coraz mniejsze.

W wyniku takiego treningu, neurony na mapie Kohonena znajdujące się blisko siebie odpowiadają klasom wektorów wejściowych, które są do siebie podobne (patrz rys. 5.1:2). Dlatego takie regiony nazywamy mapami.



rys. 5.1:2 Wektory wejściowe z odpowiadającymi im mapami na sieci Kohonena. Podobne wektory mogą być przypisane do różnych neuronów w obrębie jednej mapy.

Sieci Kohonena, ze względu na bardzo dobrą właściwość grupowania, świetnie nadają się do redukcji wymiaru danych ([75] [76] [77]). Można zamienić wynik analizy CWT (czyli skalogram), który reprezentuje przestrzeń 3D na dane dwuwymiarowe w postaci sekwencji indeksów wygrywających neuronów. W tym celu należy podzielić skalogram na odcinki (okna) wzdłuż osi czasu, a następnie, używając ich jako wektorów wejściowych (jedno okno – to jeden wektor), wytrenować sieć Kohonena. Na końcu, za pomocą już wytrenowanej sieci, należy dla każdego wektora wejściowego (czyli okna skalogramu) wyznaczyć jedną wartość – numer wygrywającego neuronu (rys. 5.1:3).

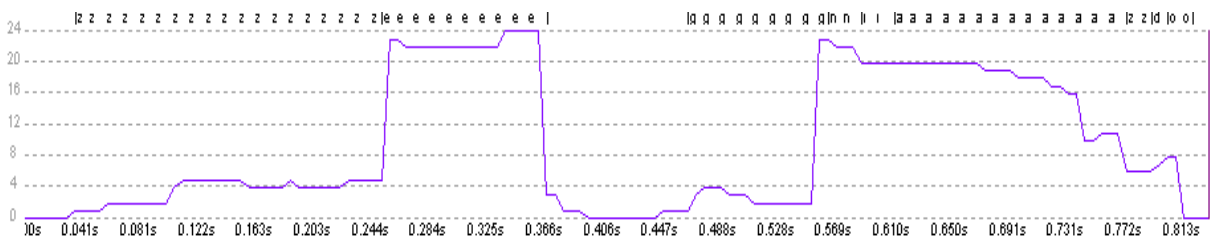


**rys. 5.1:3** Sposób redukcji skalogramu (3D) do sekwencji indeksów wygrywających neuronów sieci Kohonena (2D) dla kolejnych okien czasowych od 1 do  $M$ .

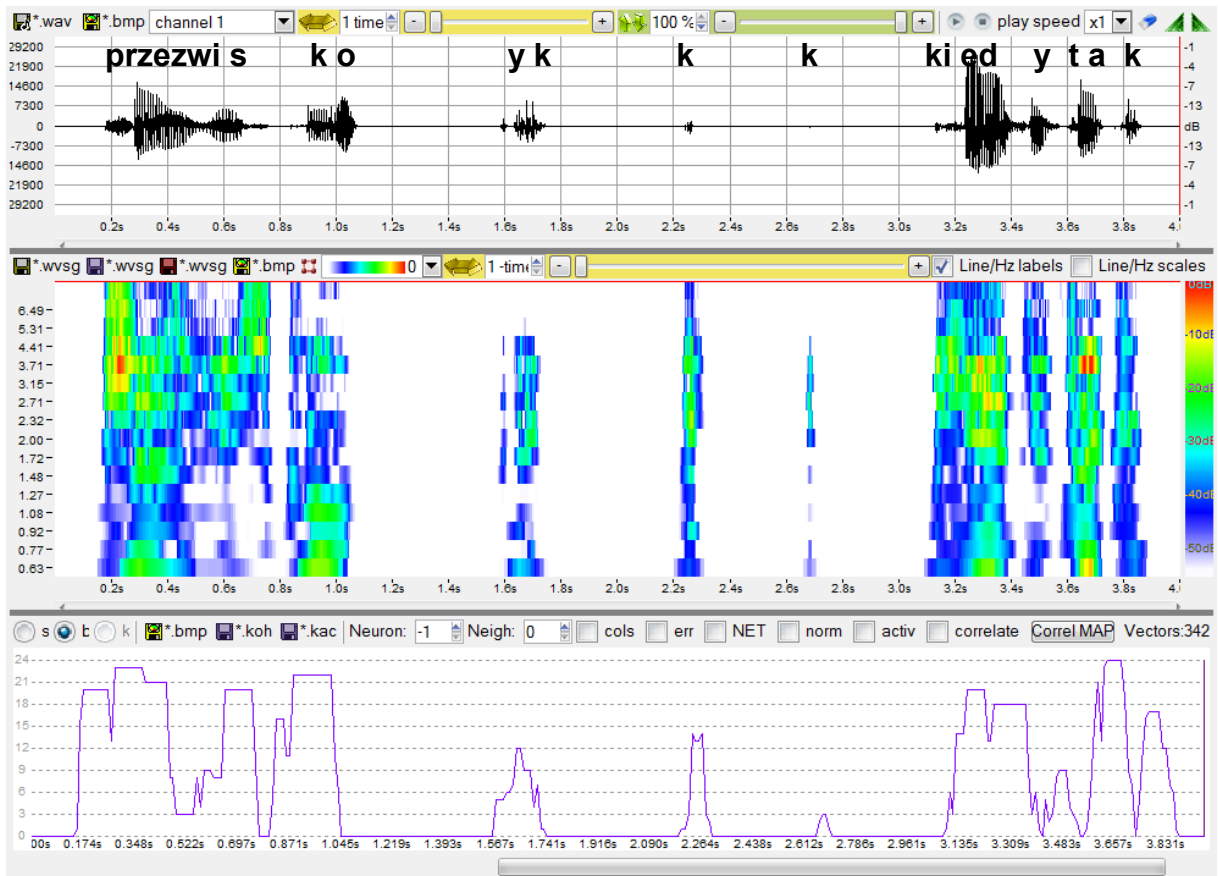
Tak zredukowany wynik jest łatwiejszy do analizy z powodu mniejszej ilości danych. Przykładowy wynik takiej redukcji prezentuje rys. 5.1:5 i rys. 5.1:6. Neurony sieci Kohonena ponumerowano (patrz rys. 5.1:4) od lewego-górnego rogu wierszami tak, aby móc je przedstawić w postaci sekwencji indeksów wygrywających neuronów.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

**rys. 5.1:4** Sposób numeracji neuronów w sieci Kohonena.



**rys. 5.1:5** Sekwencja indeksów wygrywających neuronów sieci Kohonena 5x5 dla wypowiedzi „że gniazdo”. Zrzut ekranu z programu WaveBlaster.



rys. 5.1:6 Zobrazowanie redukcji trójwymiarowego CWT<sup>0</sup> wypowiedzi „przezwi s k o y k k k k i e d y t a k” (na górze: oś Y – skale barkowe, oś X – czas) do sekwencji indeksów wygrywających neuronów sieci Kohonena (na dole: oś Y – wygrywający neuron sieci 5x5, oś X – czas). Zrzut ekranu z programu WaveBlaster.

Przed treningiem sieć jest inicjowana danymi losowymi, dlatego też ten sam ciąg wektorów wejściowych może dać inną sekwencję indeksów wygrywających neuronów. Dzieje się tak, ponieważ ze względu na tę losowość, tworzy się taka sama liczba skupisk (map), ale w innych rejonach sieci (rys. 5.1:8).

Aby tego uniknąć (tzn. aby za każdym razem algorytm dawał taki sam wynik bez względu na to jak sieć była zainicjowana), inicjowano sieć Kohonena na różne sposoby:

- losowymi wartościami o rozkładzie jednorodnym lub Gaussa,
- wybranymi wektorami wejściowymi posortowanymi w różny sposób,
- wektorami reprezentującymi środki skupisk uzyskane algorytmem k-means,
- posortowanymi neuronami (według modułu ich wag) uzyskanymi ze wstępnego treningu sieci Kohonena.

Jednak algorytm modyfikowania wygranych neuronów wraz z sąsiadami jest na tyle silny, że nie zauważono żadnej różnicy. Wyniki nadal nie były deterministyczne.



Ostatecznie autor rozprawy wprowadził następującą modyfikację algorytmu trenowania – ‘zerowanie pierwszego neuronu’. Po zainicjowaniu sieci, neuron o numerze 0 jest zerowany i oznaczany jako ‘tylko do odczytu’. Bierze on udział we wszystkich obliczeniach, ale jego wagi nie są zmieniane zgodnie ze wz. 5.1.2 (wartość wektora  $\vec{W}_0$  zawsze wynosi  $\{0,0,\dots,0\}$ ). Jego wagi są zawsze zerowe, dlatego też neuron ten zawsze przyciąga ciszę i inne ‘słabe’ sygnały do lewego górnego rogu sieci. W naturalny sposób, ze względu na modyfikację sąsiadów, wektory o średnich wartościach są grupowane w środkowych rejonach sieci, a wektory wejściowe o największych wartościach umieszczane są w prawym dolnym rogu (rys. 5.1:7).

1	1			2
1	1		2	
		2		
	2		3	3
2			3	3

**rys. 5.1:7** Zobrazowanie umiejscowienia map dla wektorów wejściowych słabych (1), średnich (2) i silnych (2) na sieci Kohonena

garnuszek

im

im

imniej



rys. 5.1:8 Różne sekwencje indeksów wygrywających neuronów sieci Kohonena 5x5 dla wypowiedzi „garnuszek im im im mniej”. Najniższy wykres jest uzyskany zmodyfikowanym algorytmem trenowania – ciska jest zawsze reprezentowana przez neuron nr 0. Zrzut ekranu z programu WaveBlaster.

## 5.2. Perceptron wielowarstwowy

Perceptron wielowarstwowy [74] [76] [78] [79] [81] [82] [83] jest jednym z podstawowych modeli sztucznych sieci neuronowych. Powstał on w oparciu o badania nad zachowaniem komórek nerwowych, czyli neuronów biologicznych. Model takiego sztucznego neuronu zakłada, że każdy neuron może mieć wiele wejść, na których przyjmuje sygnały wejściowe (pobudzenia) i jeżeli ich suma jest większa od ‘wartości progowej’ neuronu, to na wyjściu przechodzi on w stan aktywności o wartości zadanej wz. 5.2.1. Z kolei wyjście to może być połączone z wieloma innymi neuronami, dla których ta wartość aktywacji jest jednym z sygnałów wejściowych.

$$f(u_i) = \frac{1}{1 + e^{-\beta u_i}}$$

wz. 5.2.1

gdzie :

$f(u_i)$  – wartość funkcji aktywacji,

$\beta$  – stała dobierana przez użytkownika (wpływa na stromość funkcji aktywacji),

$u_i$  – suma sygnałów wejściowych.

$$u_i = \sum_{j=1}^N w_{ij}x_j + W_{i0}$$

wz. 5.2.2

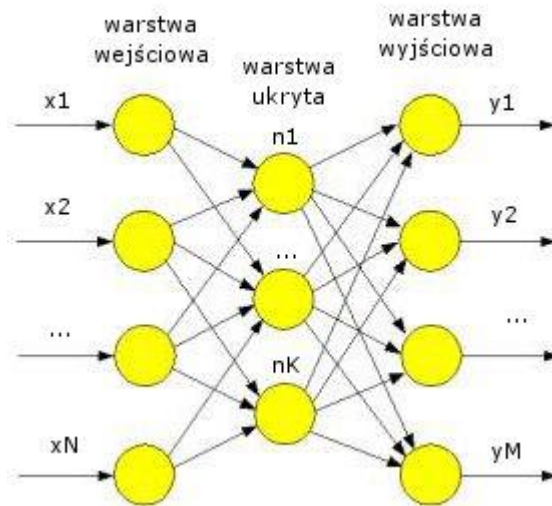
gdzie:

$w_{ij}$  – waga i-tego połączenia,

$x_j$  – wartość j-tego połączenia,

$W_{i0}$  – wartość progowa funkcji aktywacji.

Perceptron wielowarstwowy (ang. Multi layer perceptron – MLP) składa się z wielu warstw, gdzie warstwa oznacza grupę neuronów niepołączonych ze sobą, które jednocześnie są połączone z neuronami warstw poprzedniej i następnej (rys. 5.2:1). Istotną cechą takiej sieci jest jednokierunkowość, czyli przesyłanie sygnału tylko w jednym kierunku od warstwy pierwszej (wejściowej) do warstwy ostatniej (wyjściowej), co prezentują zwroty na połączeniach na rys. 5.2:1. Każda warstwa może mieć dowolną ilość neuronów. Prezentowany poniżej perceptron 3-warstwowy ma  $N$  neuronów wejściowych,  $K$  neuronów ukrytych i  $M$  neuronów wyjściowych – zatem oznacza się go symbolem MLP  $N$ - $K$ - $M$ . Liczba neuronów wejściowych jest związana z wymiarem wektorów wejściowych, natomiast liczba neuronów wyjściowych z wymiarem wektorów wyjściowych, czyli ilością klas (grup).



rys. 5.2:1 Schemat perceptronu 3-warstwowego przekształcającego wektor wejściowy  $X=\{x_1,x_2,\dots,x_N\}$  w wektor wyjściowy  $Y=\{y_1,y_2,\dots,y_M\}$  i posiadający  $K$  neuronów ukrytych.

Aby perceptron przekształcał dane zgodnie z naszymi oczekiwaniami należy odpowiednio „nauczyć” sieć, czyli ustawić wartości wag na jego połączeniach. W tej pracy skorzystano z dwóch algorytmów uczenia nadzorowanego (z nauczycielem):

- algorytm wstecznej propagacji (ang. back propagation),
- algorytm gradientów sprzężonych (ang. conjugate gradient).

Oznacza to, że przed procesem uczenia należy ręcznie przypisać każdy wektor wejściowy do odpowiadającego mu wektora wyjściowego. W ten sposób każdy wzorzec wejściowy ma przypisaną na wyjściu sieci wartość oczekiwaną. Na tej podstawie, w każdym kroku algorytmu, można wyliczyć funkcję błędu:

$$E^p = \frac{1}{2} \sum_{k=0}^M (y_k^p - d_k^p)^2$$

wz. 5.2.3

gdzie:

$M$  – liczba neuronów,

$y_k^p$  – wartość otrzymana na wyjściu dla  $i$ -tego neuronu i  $p$ -tego wektora wejściowego,

$d_k^p$  – wartość oczekiwana na wyjściu dla  $i$ -tego neuronu i  $p$ -tego wektora wejściowego.

W/w algorytmy należą do grupy gradientowych, które w sposób iteracyjny dążą do minimalizacji funkcji błędu.

Autor pracy wybrał ten rodzaj sieci na podstawie badań Szczurowskiej [72] [73] [74] [76], która wykazała ich wysoką przydatność w wykrywaniu wzorców mowy niepełnej.

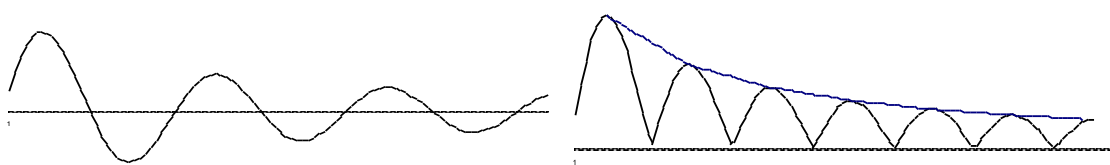
## 6. Algorytmy do automatycznego rozpoznawania niepełności w mowie ciągłej

Rozpoznawanie niepełności realizowano według następującego planu:

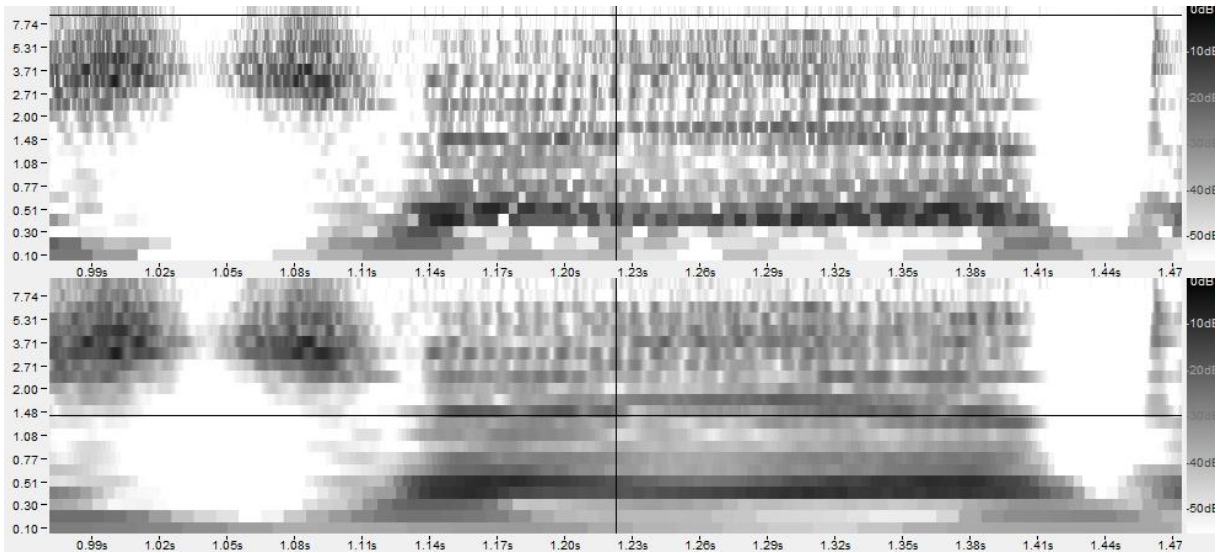
1. parametryzacja plików dźwiękowych algorytmem CWT,
2. uzyskanie sekwencji indeksów wygrywających neuronów na podstawie skalogramów CWT,
3. analiza tak uzyskanych sekwencji i dobranie odpowiednich algorytmów rozpoznawania dla każdej grupy niepełności,
4. zastosowanie miar oceny wyników rozpoznawania.

### 6.1. Parametryzacja sygnału przy użyciu CWT

Dla zadanej skali  $a$ , wartości  $CWT$  zmieniają się sinusoidalnie, co wynika wprost ze sposobu przemnażania falki przez sygnał dla kolejnych przesunięć  $b$  (patrz rys. 4.3.1:1). Widać to wyraźnie na rys. 6.1:1, gdzie wartości  $CWT$  rosą-zanikają-rosną-zanikają zamiast utrzymywać się na stałym poziomie. Dla uproszczenia późniejszego przetwarzania takiego skalogramu wygładzono go na zasadzie wyznaczania konturu oscylacji (co obrazuje rys. 6.1:1 oraz rys. 6.1:2):



rys. 6.1:1 Po lewej wartości  $CWT_{a,b}$  (oś pionowa) dla jednej skali  $a$  i zmieniającego się  $b$  (oś pozioma), po prawej wyznaczony kontur dla wartości  $|CWT_{a,b}|$



rys. 6.1:2 U góry skalogram  $CWT^0$  bez wygładzania, u dołu z wygładzaniem. Pionowa oś to skala  $a$  zamieniona na Hz (czyli oś częstotliwości), w poziomie wartości  $b$  zamienione na sekundy (czyli oś czasu). Zrzut ekranu z programu WaveBlaster.

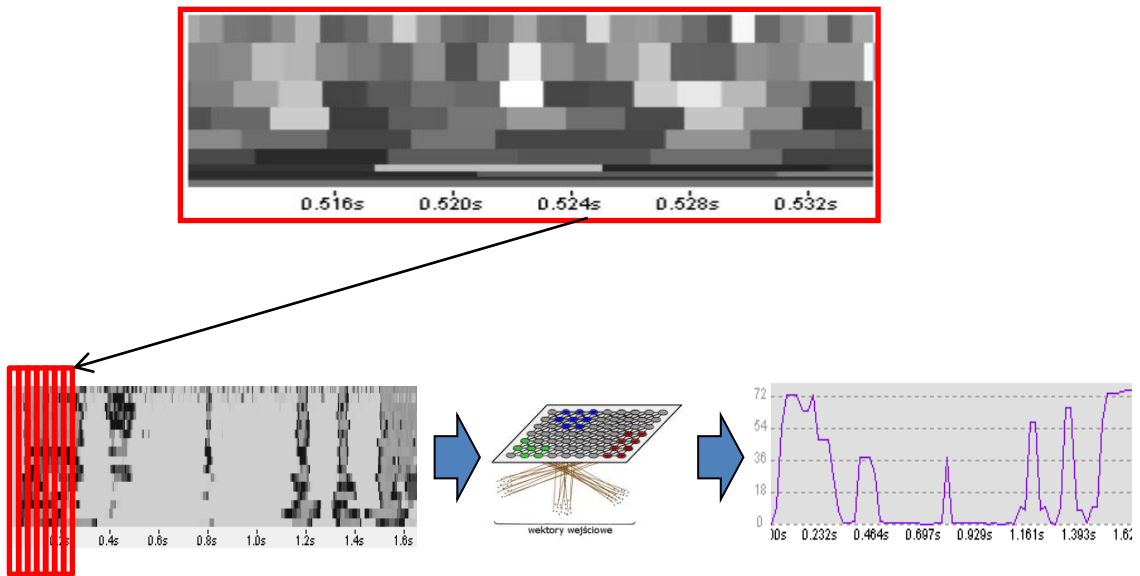
## 6.2. Tworzenie wektorów wejściowych na podstawie skalogramu

### *CWT*

W celu redukcji liczby sekwencji indeksów wygrywających neuronów sieci Kohonena, skalogram CWT podzielono na okna o szerokości  $23ms$  (przyjęto na podstawie prac Suszyńskiego [65] [68] [69] [70] [71]). W każdym  $i$ -tym oknie wyznaczano średnią arytmetyczną oddzielnie dla wartości każdej skali. Tak powstały wektor (wz. 6.2.1) był następnie przekazywany do sieci Kohonena (patrz rys. 6.2:1).

$$\vec{V} = \{mean(CWT_{55,i}), mean(CWT_{69,i}), \dots, mean(CWT_{2159,i}), mean(CWT_{3718,i})\}$$

wz. 6.2.1



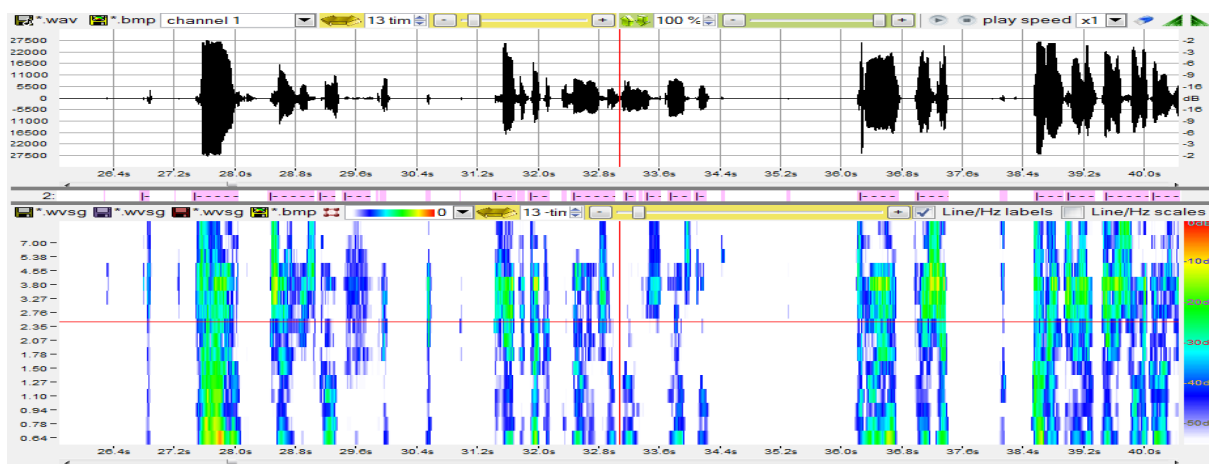
rys. 6.2:1 U góry jedno okno CWT<sup>0</sup> o szerokości 23ms. U dołu schemat obrazujący algorytm wyznaczający sekwencję indeksów wygrywających neuronów sieci Kohonena: 1) dzielenie CWT<sup>0</sup> na okna (tj. wektory V, gdzie każdy wektor odpowiada 23ms wypowiedzi); 2) przekazywanie okien (wektorów) do sieci Kohonena; 3) wyznaczanie sekwencji wygrywających neuronów.

### 6.3. Detekcja początku i końca fonacji

Po wyznaczeniu wektorów  $\vec{v}$  (wz. 6.2.1), ale przed przekazaniem ich do sieci Kohonena, należy wyznaczyć fragmenty fonacji. Jest to niezbędne, ponieważ wszystkie prezentowane metody detekcji niepłynności bazują na porównywaniu sąsiadujących fragmentów mowy (brane są pod uwagę czasy tych fonacji, odstępy między nimi).

Zastosowano prosty algorytm: wektor  $\vec{v}$  jest oznaczany jako cisza, jeżeli wszystkie jego wartości są mniejsze od parametru ‘odcięcia szumów’. Bazując na pracach Suszyńskiego [65] [68] [69] [70] [71] za punkt wyjścia dla tego parametru przyjęto wartość -55dB (gdzie maksymalna wartość wynosi 0 dB – patrz wz. 4.3.1.2). Pozostałe wektory są oznaczane jako fonacja. Pojęcie ‘fragmentu mowy’ definiujemy jako sekwencje wektorów fonacji.





rys. 6.3:1 Przykład oscylogramu (u góry), skalogramu (u dołu) i automatycznie wyznaczonych fragmentów mowy (po środku). Zrzut ekranu z programu WaveBlaster.

#### 6.4. Ocena wyników rozpoznawania

Wyniki rozpoznawania zostały obliczone za pomocą wzorów [2]:

$$czul = \frac{P}{A}, przew = \frac{P}{P + B}$$

wz. 6.4.1

gdzie:

*czul* – czułość (sensability),

*przew* – przewidywalność (predictability),

*P* – liczba poprawnie rozpoznanych niepłynności,

*B* – liczba płynnych fragmentów błędnie oznaczonych jako niepłynne,

*A* – liczba niepłynności.

Zatem im większa czułość tym więcej poprawnie rozpoznanych niepłynności, natomiast im większa przewidywalność tym mniej błędów. Parametry algorytmów rozpoznawania dobierane są tak, aby uzyskać jak największe wartości obu współczynników. Skupienie się tylko na jednym, tj. optymalizacja algorytmu tak, aby czułość lub przewidywalność była jak największa, zawsze powoduje nieproporcjonalny spadek wartości drugiego współczynnika.

#### 6.5. Wybór algorytmów oraz wyniki rozpoznawania

Rodzaje niepłynności analizowane w tej rozprawie istotnie różnią się od siebie – z tego powodu autor pracy postanowił każdy z nich analizować niezależnie, aby jak najlepiej dopasować algorytmy rozpoznawania do rodzaju wykrywanych cech. Proces ten można podzielić na cztery etapy:

1. Analiza materiału dźwiękowego – tj. czasów trwania nie płynności w badanych sygnałach, jak i czasów przerw między nimi, czasów ciszy przed lub po nie płynności na podstawie obszernego zbioru nie płynnych wypowiedzi osób jaskających się i porównaniu ich z płynnymi odpowiednikami. Na tym etapie utworzono różne statystyki czasowe, na bazie których wyznaczano warunki brzegowe dla analizowanych fragmentów mowy.
2. Analiza wyników parametryzacji fragmentów nie płynnych oraz ich płynnych odpowiedników. Na tym etapie szczegółowo przeanalizowano skalogramy CWT dla fragmentów płynnych i nie płynnych jak również ich sekwencje indeksów neuronów wygrywających sieci Kohonena. Na tej podstawie zaproponowano algorytmy rozpoznawania danej nie płynności.
3. Utworzenie procedur detekcji nie płynności, wyselekcjonowanie przestrzeni parametrów, które będą w nich zmieniane.
4. Przeprowadzenie rozpoznawania nie płynności w/w procedurami – wybranie wartości początkowych dla parametrów oraz modyfikację tych wartości w kolejnych etapach testowania na podstawie analiz otrzymywanych wyników. Ostateczne wyselekcjonowanie najlepszych wyników rozpoznawania.

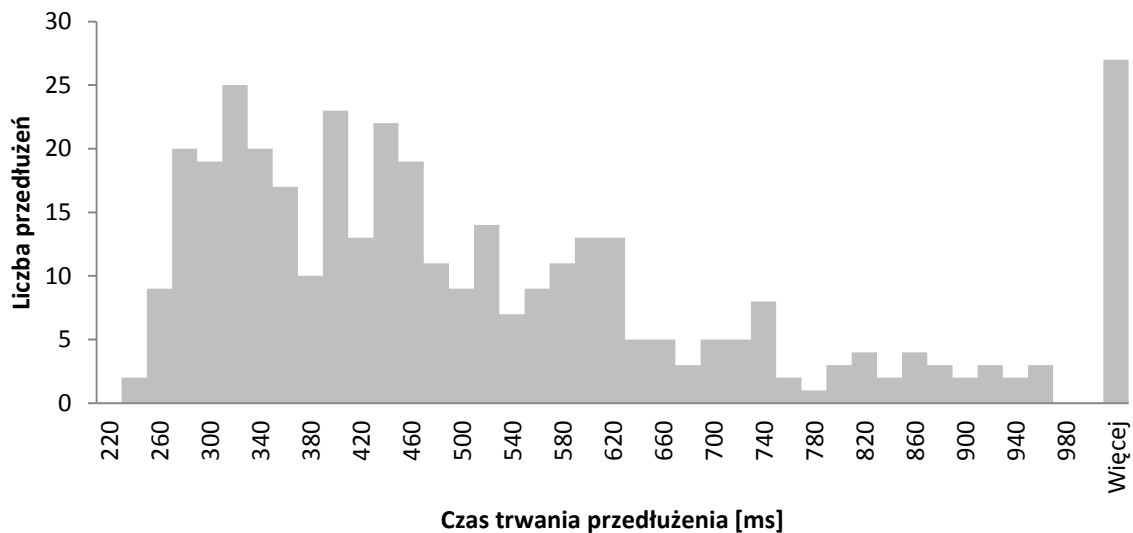
### **6.5.1. Metoda detekcji przedłużeń**

#### 6.5.1.1. Materiał dźwiękowy

Jako materiał badawczy posłużyły nagrania dziesięciu osób w tym sześciu jaskających się. W nagraniach, metodą odsłuchową, odnaleziono wszystkie przedłużenia, a następnie wycięto je z czterosekundowym otoczeniem mowy płynnej. W nagraniach płynnych również wycięto cztero-sekundowe fragmenty (w sposób losowy). Wszystkie fragmenty zostały połączone tworząc jedną wypowiedź trwającą 18 min. 32 s. Statystyki liczby przedłużeń przedstawia tab. 4.3.2-1. Odsłuchowo (posiłkując się wykresem oscylogramu oraz spektrogramu STFT) stworzono również histogram czasów trwania przedłużeń (rys. 6.5:1).

**tab. 6.5-1 Liczba przedłużeń w materiale badawczym.**

<i>a</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>	<i>i</i>	<i>j</i>	<i>m</i>	<i>n</i>	<i>o</i>	<i>s</i>	<i>sz</i>	<i>ś</i>	<i>u</i>	<i>w</i>	<i>y</i>	<i>z</i>	<i>ż</i>	<i>ź</i>	<b>Suma</b>
4	10	11	1	8	12	13	17	29	16	65	15	39	6	34	26	46	6	15	373



rys. 6.5:1 Histogram czasów trwania przedłużeń

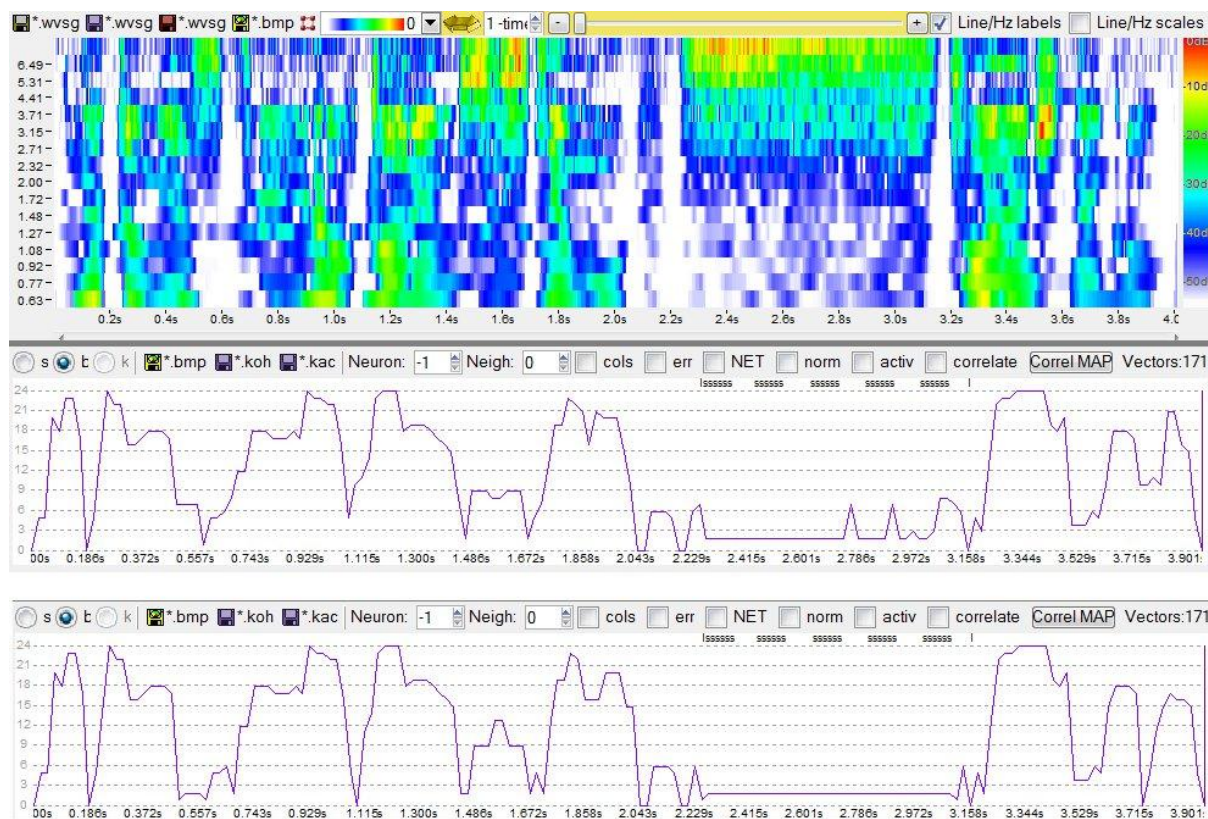
Najkrótsze badane przedłużenie trwało 226 ms, prawie wszystkie były dłuższe niż 250 ms.

#### 6.5.1.2. Analiza wyników parametryzacji

Neurony sieci Kohonena grupują podobne wektory wejściowe (patrz rozdział 5.1), dlatego oczekiwano, iż na odcinkach z przedłużeniami będzie wygrywał tylko jeden neuron. Efekt ten uzyskiwano tylko wtedy, jeżeli liczba neuronów sieci (czyli jej rozmiar) była zbliżona do liczby fonemów we fragmencie. Jeżeli neuronów było za mało w stosunku do liczby fonemów w sygnale wejściowym – odmienne fonemy były grupowane przez ten sam neuron (czyli uzyskiwaliśmy wrażenie przedłużenia w płynnych fragmentach), jeżeli neuronów było za dużo – kilka neuronów grupowało ten sam fonem (czyli uzyskiwaliśmy efekt płynności we fragmentach z zaburzeniami). Fragmenty z mową płynną posiadają dużo większe zróżnicowanie fonemów od fragmentów tej samej długości zawierających przedłużenie, ponieważ przedłużenie wypełnia jego znaczną część. Ponieważ rozmiar sieci musi być stały dla całej wypowiedzi (tj. dla wszystkich fragmentów) dobranie rozmiaru sieci, która odwzorowałaby tylko jeden wygrywający neuron tylko dla odcinka z przedłużeniem okazało się niemożliwe.

Z tego powodu, po wytrenowaniu sieci Kohonena wektorami CWT<sup>0</sup>, ale przed wygenerowaniem sekwencji indeksów neuronów wygrywających, zastosowano dodatkową, autorską modyfikację nazwaną ‘redukcją sieci Kohonena’. Ideą ‘redukcji’ jest wygładzenie

sekwencji indeksów wygrywających neuronów sieci Kohonena na odcinkach, na których znajduje się ten sam fonem (rys. 6.5:2).



rys. 6.5:2 Obrisy wygrywającego neuronu sieci Kohonena o rozmiarze 5x5 dla fragmentu „natomiast autorem systemu sssssslonecznego”. U góry obrisy bez ‘redukcji’, u dołu z ‘redukcją’. Zrzut ekranu z programu WaveBlaster.

Procedura redukcji jest następująca:

- Znajdź dwa najbliższe neurony  $k_A$ ,  $k_B$  (odległość mierzona metryką Euklidesową pomiędzy wagami neuronów).
- Jeżeli odległość jest mniejsza niż zadany dystans  $\varepsilon$  wypełnij wagi ‘słabszego’ neuronu zerami (tj. neuronu z niższą ilością przypisanych wektorów wejściowych). Dzięki temu wektory wejściowe ‘słabszego’ neuronu będą najprawdopodobniej przypisywane ‘silniejszemu’ neuronowi. Proces ten obrazuje rys. 6.5:3 i wz. 6.5.1.

$$\begin{cases} |\vec{k}_A - \vec{k}_B| < \varepsilon, \vec{k}_A \vee \vec{k}_B = \vec{0} \\ \text{w p.p., nic nie rób} \end{cases}$$

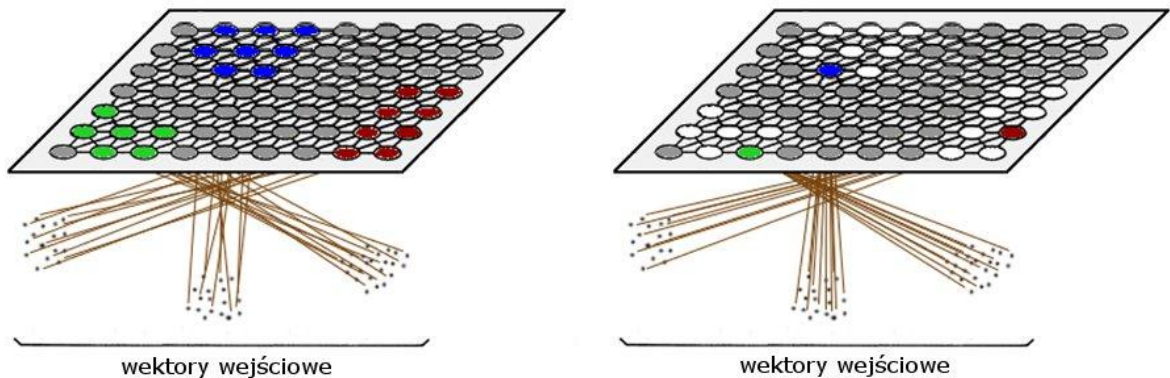
wz. 6.5.1

gdzie:

$\vec{k}_A, \vec{k}_B$  – wektory reprezentowane przez wagi neuronów  $k_A$ ,  $k_B$ ,

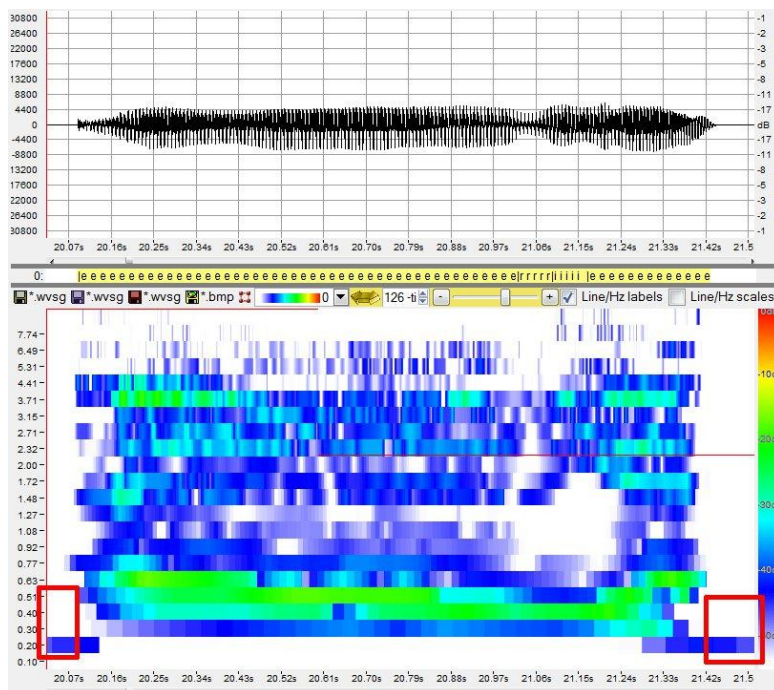
$\varepsilon$  – dystans, czyli wartość graniczna odległości pomiędzy wektorami  $\vec{k}_A$  i  $\vec{k}_B$ .

- Powtarzaj dwa poprzednie kroki dopóki istnieje para neuronów, których odległość jest mniejsza od w/w progu.



rys. 6.5:3 Sieć Kohonena przed redukcją (po lewej) i po redukcji (po prawej). Neurony z całej mapy (zielonej, niebieskiej lub czerwonej) powinny być grupowane przez najsilniejszy neuron z danej grupy.

Zauważono, że niskoczęstotliwościowe składowe skalogramu CWT (skale barkowe  $B=1..3$  odpowiadające zakresowi  $100-300\text{ Hz}$ ) bardzo często rozpoczynają się trochę przed fonacją i utrzymują się jeszcze po niej (rys. 6.5:4) utrudniając tym samym poprawną detekcję fonacji. Z tego powodu zdecydowano się na użycie tylko 18 skal barkowych ( $B=4,5,6,7,\dots,21$ ).



rys. 6.5:4 Oscylogram i skalogram CWT dla skal barkowych wypowiedzi „erie”. Czerwonymi prostokątami zaznaczono niechciane wartości CWT ‘wystające’ poza fonację. Zrzut ekranu z programu WaveBlaster.

### 6.5.1.3. Procedura detekcji przedłużeń

Przyjęto następującą procedurę detekcji przedłużeń (podkreślone elementy są parametrami algorytmu):

1)	wczytaj plik dźwiękowy
2)	oblicz współczynniki $CWT^0$ (wz. 4.3.1.2) dla skal barkowych (wz. 4.5.1)
3)	wykryj fragmenty mowy dla odcięcia szumów na poziomie -55dB (rozdział 6.3) dla szerokości okna 23,2ms (rozdział 6.2) i 50% przesunięcia okna (50% dla zwiększenia rozdzielczości wykrywania fonacji)  <i>następnie każdy fragment podziel na okna o szerokości 23,2ms i 100% przesunięcia okna i ponownie oblicz wektory (wz. 6.2.1)</i>
4)	dla fragmentów mowy dłuższych niż 200ms:
5)	wytnij fragment $CWT^0$ z zadaniem <u>otoczeniem</u> ( <i>otocz_wycinania</i> )
6)	zredukuj współczynniki $CWT^0$ do jednego indeksu neuronu zwycięskiego w każdym oknie dla sieci Kohonena o zadanej <u>rozmiarze</u> ( <i>rozm_Koh</i> ) i zadanej <u>sąsiedztwie</u> ( <i>sas_Koh</i> ). Sieć uczona była przez 100 epok z liniowo malejącym współczynnikiem uczenia 0.20-0.10
7)	dla tak wytrenowanej sieci Kohonena zastosuj ‘redukcję’ neuronów z zadaniem <u>dystansem</u> ( <i>dyst_redukcji</i> ) i ponownie wygeneruj sekwencję indeksów neuronów zwycięskich
8)	jeżeli w w/w sekwencji istnieje odcinek dłuższy od zadanej <u>długości</u> ( <i>dl_sekwencji</i> ), w którym wygrywa tylko jeden neuron, oznacz go jako przedłużenie
9)	na podstawie odsłuchowych i automatycznych zaznaczeń wygeneruj współczynniki wykrywalności <i>czul</i> i <i>przew</i> (wz. 6.4.1)

W kroku 4) wartość 200ms została przyjęta na podstawie histogramu przedstawionego na rys. 6.5:1.

Jak wyjaśniono w rozdziale 6.5.1.2, liczba fonemów we fragmencie, a tym samym długość fragmentu, jak również parametry sieci Kohonena mają duże znaczenie przy

generowaniu sekwencji indeksów neuronów zwycięskich. Z tego powodu w kroku 5) otoczenie wycinanego fragmentu mowy, czyli całkowita długość wycinanego odcinka (*otocz\_wycinania*) jest parametrem algorytmu. W kroku 6) rozmiar (*rozm\_Koh*) i sąsiedztwo (*sas\_Koh*) uczenia są parametrami algorytmu.

W kroku 7) parametrem jest dystans (*dyst\_redukcji*) algorytmu ‘redukcji sieci Kohonena’ (wz. 6.5.1).

Mimo, iż każde przedłużenie w badanym materiale jest dłuższe niż 226 ms to odpowiadająca mu sekwencja indeksów wygrywających utrzymująca stałą wartość opisaną w kroku 8) nie musi spełniać tego warunku – z tego powodu zdecydowano się na wprowadzenie parametru *dl\_sekwencji* oznaczającą minimalną długość takiej sekwencji. Należy pamiętać, że ustalenie parametru *dl\_sekwencji* na zbyt małą wartość spowoduje nadmierne zaliczanie fragmentów płynnych do przedłużeń.

#### 6.5.1.4. Wyniki rozpoznawania przedłużeń

Wszystkie serie danych sprawdzano dla parametru *dyst\_redukcji* = 0.30, 0.35, 0.40, 0.45, 0.50, 0.55.

tab. 6.5-2 Wyniki (w [%]) automatycznego rozpoznawania przedłużeń. C – czuł, P – przew (wzór 6.4.1), P1 – otocz\_wycinania, P2 – rozm\_Koh, P3 – sas\_Koh, P4 – dyst\_redukcji, P5 – dl\_sekwencji

	P4=0.30		P4=0.35		P4=0.40		P4=0.45		P4=0.50		P4=0.55	
	C	P	C	P	C	P	C	P	C	P	C	P
seria 1												
P1=0 P2=3x3 P3=2.5-0.5 P5=250	63	94	68	94	71	95	74	93	78	90	81	87
P1=0 P2=3x3 P3=2.5-1.0 P5=250	74	80	76	76	80	76	82	73	85	69	86	67
P1=0 P2=4x4 P3=2.5-0.5 P5=250	49	94	52	94	60	94	63	94	68	93	75	92
P1=0 P2=4x4 P3=2.5-1.0 P5=250	62	92	65	89	71	88	76	84	78	81	84	81
P1=0 P2=3x3 P3=2.5-0.5 P5=200	75	76	81	77	82	77	84	76	87	74	90	70
P1=0 P2=3x3 P3=2.5-1.0 P5=200	77	80	78	76	81	72	84	70	87	67	90	51
P1=0 P2=4x4 P3=2.5-0.5 P5=200	61	87	66	87	72	86	76	86	81	82	84	80
P1=0 P2=4x4 P3=2.5-1.0 P5=200	76	78	80	76	83	73	84	70	86	67	89	63
seria 2												
P1=0 P2=5x5 P3=2.5-0.5 P5=250	37	95	43	95	52	98	57	96	62	96	68	95
seria 3												
P1=1500 P2=3x3 P3=2.5-0.5 P5=250	78	83	81	84	82	83	85	82	86	80	87	78
P1=1500 P2=3x3 P3=2.5-1.0 P5=250	84	72	80	68	81	64	90	64	91	61	93	59
P1=1500 P2=4x4 P3=2.5-0.5 P5=250	62	89	63	90	72	90	80	89	83	88	84	86
P1=1500 P2=4x4 P3=2.5-1.0 P5=250	77	86	80	84	85	82	86	78	90	77	91	74
P1=1500 P2=3x3 P3=2.5-0.5 P5=200	88	66	91	68	92	66	92	64	93	62	94	60
P1=1500 P2=3x3 P3=2.5-1.0 P5=200	90	55	92	52	93	50	94	47	96	45	96	43
P1=1500 P2=4x4 P3=2.5-0.5 P5=200	77	81	81	80	87	80	89	77	92	73	94	73

P1=1500 P2=4x4 P3=2.5-1.0 P5=200	89	72	90	66	93	65	95	63	96	59	97	56
seria 4												
P1=1500 P2=5x5 P3=2.5-0.5 P5=250	49	92	63	93	65	93	66	91	74	91	77	88
seria 5												
P1=1000 P2=4x4 P3=2.5-0.5 P5=250	51	93	59	92	62	92	69	91	73	90	80	88
P1=1500 P2=4x4 P3=2.5-0.5 P5=250	62	89	63	90	72	90	80	89	83	88	84	86
P1=2000 P2=4x4 P3=2.5-0.5 P5=250	67	91	71	91	75	89	78	87	80	86	85	85
P1=2500 P2=4x4 P3=2.5-0.5 P5=250	73	91	77	88	78	88	81	87	82	85	<b>92</b>	<b>82</b>
P1=3000 P2=4x4 P3=2.5-0.5 P5=250	76	89	78	87	81	88	83	86	84	84	87	84
P1=3500 P2=4x4 P3=2.5-0.5 P5=250	73	88	77	88	77	88	82	86	82	82	87	79
P1=2500 P2=5x5 P3=2.5-0.5 P5=250	60	92	65	91	68	90	75	89	79	88	82	86
P1=3000 P2=5x5 P3=2.5-0.5 P5=250	65	93	70	92	74	91	75	89	79	89	83	86
P1=3500 P2=5x5 P3=2.5-0.5 P5=250	69	93	72	91	72	90	77	89	82	87	82	83

W pierwszej serii testów sprawdzono konfiguracje:

- *otocz\_wycinania* = 0 (czyli brak otoczenia),
- *rozm\_Koh* = 3x3 oraz 4x4,
- *sas\_Koh* = 2.5-0.5 oraz 2.5-1.0,
- *dl\_sekwencji* = 200ms oraz 250ms.

W większości przypadków dla wartości *dl\_sekwencji* =250 ms otrzymywano lepsze rezultaty. Czułość była mniejsza (czyli znaleziono mniej przedłużeń – co jest oczywiste z racji większego progu długości), ale przewidywalność była większa (czyli algorytm robił mniej błędów). We wszystkich przypadkach w konfiguracji *sas\_Koh*=2.5-0.5 uzyskiwano lepszą rozpoznawalność niż w *sas\_Koh* =2.5-1.0. Rozmiar sieci *rozm\_Koh*=3x3/4x4 nie spowodował wyraźnych różnic.

Na wszelki wypadek w serii drugiej sprawdzono jeszcze sieć o rozmiarze 5x5 – skorzystano z wniosków z serii 1 i ustawiono *sas\_Koh*=2.5-0.5, *dl\_sekwencji*=250ms.

Uznano, że wprowadzenie stałej szerokości wektor *otocz\_wycinania* = 1500ms może zupełnie zmienić zachowanie sieci, więc postanowiono powtórzyć obie serie danych (serię 1 i 2) z tym parametrem – tworząc serie 3 i 4. Wyniki okazały się być takie same lub lepsze. Widać również, że sieć 4x4 radzi sobie lepiej od sieci 3x3.

Skoro stała szerokość wycinanego fragmentu dała bardziej obiecujące wyniki, postanowiono dokładniej zbadać parametr *otocz\_wycinania* – przyjęto wartości 1000ms, 1500ms, 2000ms, 2500ms, 3000ms, 3500ms. Kolejnym wnioskiem było ustawienie konfiguracji na *rozm\_Koh* =4x4, *sas\_Koh* =2.5-0.5, P5=250ms. Dodatkowo zbadano jeszcze



większą sieć (5x5), aby liczba neuronów sieci lepiej odpowiadała ilości fonemów w fragmencie, ponieważ dla dłuższych fragmentów zwiększa się liczba zawartych w nim fonemów (patrz rozdział 6.5.1.2).

Dla wszystkich serii zwiększanie parametru *dyst\_redukcji* dawało lepsze rezultaty, tj. wzrost czułości był bardziej znaczący niż towarzyszący mu spadek przewidywalności. Większe wartości nie były sprawdzane, ponieważ w większości przypadków spadek przewidywalności stawał się zbyt duży.

Najlepszy wynik uzyskano na poziomie *czul=92%*, *przew=82%*.

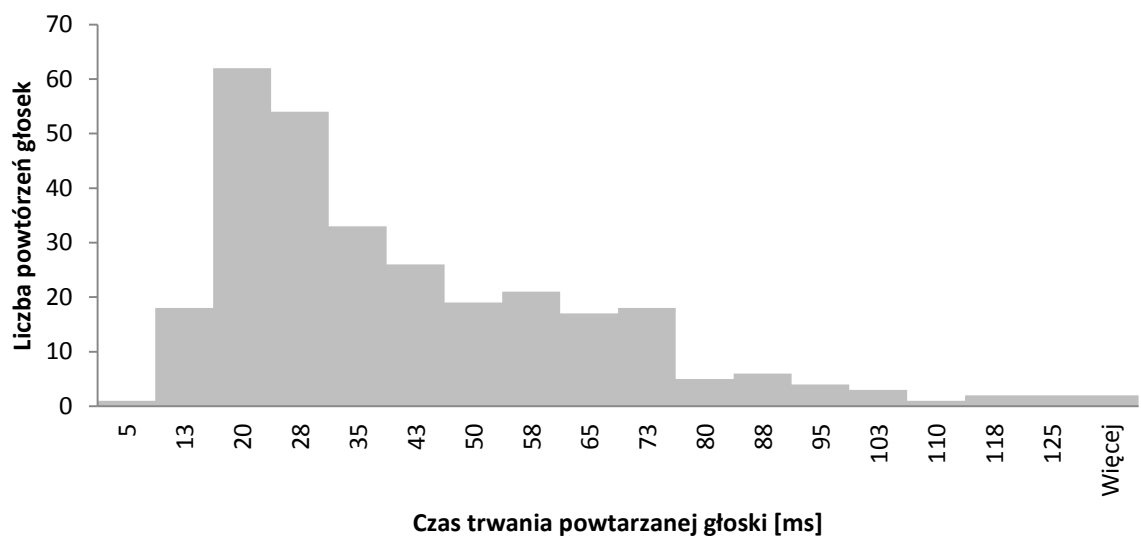
### **6.5.2. Metoda detekcji powtórzeń głosek**

#### 6.5.2.1. Materiał dźwiękowy

Jako materiał badawczy posłużyły nagrania dziewięciu osób jaskających się (jedna kobieta, pięciu mężczyzn oraz trzech chłopców). W nagraniach, metodą odsłuchową, odnaleziono wszystkie powtórzenia, a następnie wycięto je z czterosekundowym otoczeniem mowy płynnej. Wszystkie fragmenty zostały połączone tworząc jedną wypowiedź trwającą 9 min. 43 s. Statystyki liczby powtórzeń przedstawia tab. 6.5-3. Odsłuchowo (posiłkując się wykresem oscylogramu oraz spektrogramu STFT) stworzono również histogramy czasów trwania powtarzanych głosek (rys. 6.5:5) oraz przerw pomiędzy powtórzeniami (rys. 6.5:6):

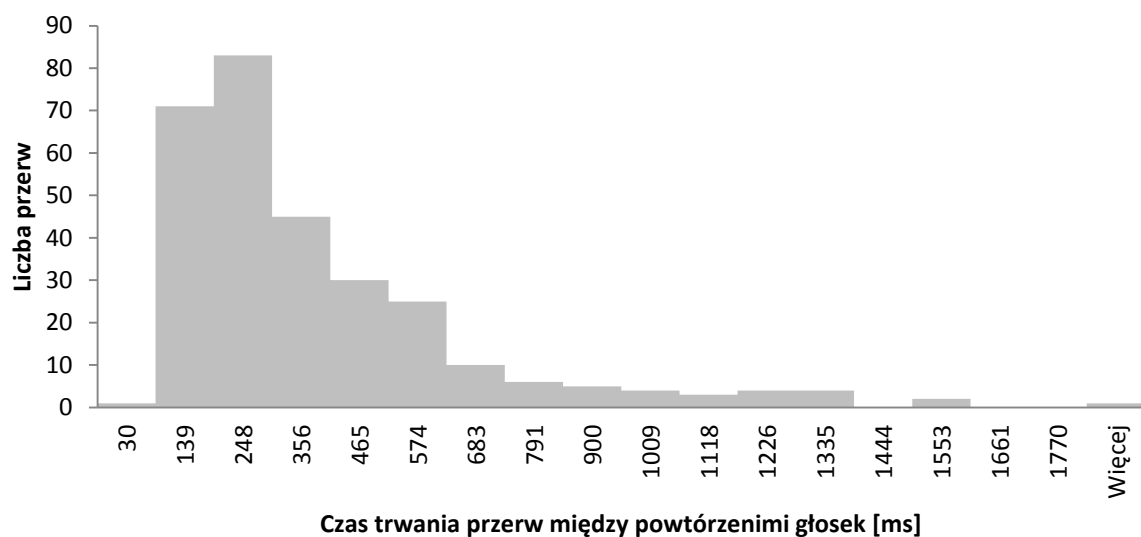
**tab. 6.5-3** Liczba wystąpień par głosek w materiale badawczym, z których druga jest powtórzeniem pierwszej.

<i>b</i>	<i>d</i>	<i>g</i>	<i>k</i>	<i>n</i>	<i>o</i>	<i>p</i>	<i>t</i>	<b><i>Suma</i></b>
23	6	10	82	2	1	66	104	294



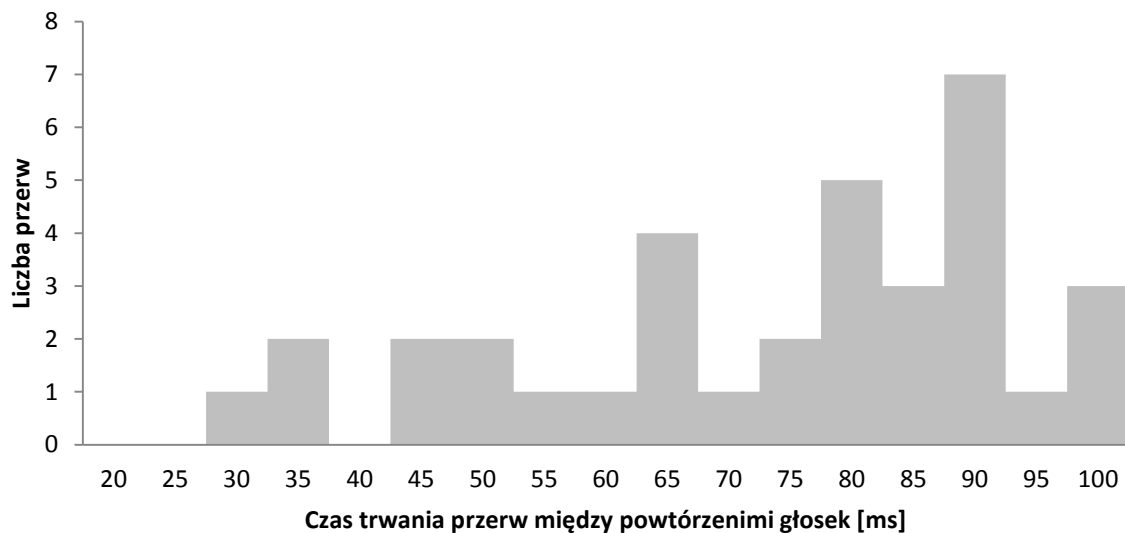
rys. 6.5:5 Histogram czasu trwania powtarzanej głoski

Najdłuższa zaburzona głoska trwała 131 ms.



rys. 6.5:6 Histogram czasu między powtórzeniami głosek

Jak widać, przerwy między powtórzeniami mogą być bardzo długie – zatem nie da się ustalić ich górnej granicy. Aby ustalić ich dolną granicę, przyjrzyjmy się początkowym wartościom histogramu z rys. 6.5:6.



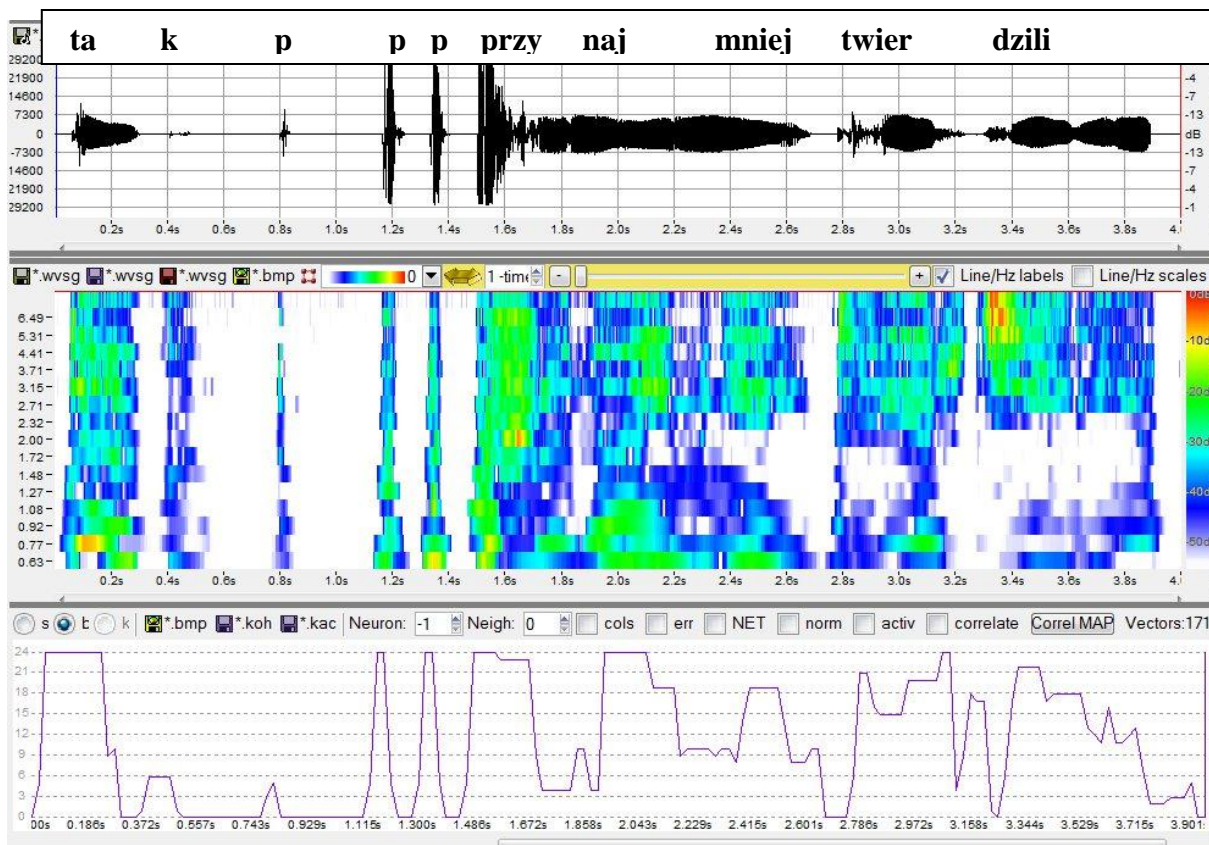
rys. 6.5:7 Histogram czasów między powtórzeniami głosek – początkowe zbiory

Wszystkie przerwy trwają co najmniej 30 ms.

#### 6.5.2.2. Analiza wyników parametryzacji

Na podstawie histogramu czasów trwania powtarzanej głoski (rys. 6.5:5), za maksymalną akceptowalną długość głoski przyjęto wartość 200 ms.

Sekwencja indeksów wygrywających neuronów sieci Kohonena dla powtórzeń głosek jest dość charakterystyczny: krótkie wzniesienie otoczone ciszą (przykład na rys. 6.5:8).



rys. 6.5:8 Sekwencja indeksów wygrywających neuronów sieci Kohonena dla wypowiedzi „tak p p p przynajmniej twierdzili”. Zrzut ekranu z programu WaveBlaster.

Nie zauważono wyraźnych różnic między sekwencjami indeksów wygrywających neuronów niepełnych powtórzeń głosek, a pojedynczymi, płynnie wypowiedzianymi spójnikami (takimi jak ‘i’, ‘a’), dlatego zdecydowano się skorzystać z doświadczeń Szczurowskiej [74] [76] i użyto perceptronu 3-warstwowego.

W tym celu skorzystano z narzędzia Intelligent Problem Solver pakietu STATISTICA (moduł Neural Networks) [64]. Dla zadanych danych wejściowych sprawdza on różne rozmiary sieci oraz różne kombinacje algorytmów uczenia. Jak uzasadniono w poprzednim akapicie – ustawiono opcje przeszukiwania tylko perceptronów 3-warstwowych. Narzędzie to nie używa wszystkich wektorów do uczenia sieci. Uczenie takie doprowadziłoby do dopasowania sieci do danych wejściowych, ztracając tym samym umiejętność generalizacji cech. Sieć nauczyłaby się w 100% wszystkich danych wejściowych (ich szczegółów) i nie poradziłaby sobie z rozpoznawaniem innych danych (czyli z generalizowaniem cech). Z tego powodu dane wejściowe są losowo dzielone na trzy grupy:

- zbiór uczący (50% wektorów),
- zbiór weryfikujący (25% wektorów),
- zbiór testujący (25% wektorów).

Dla każdego zbioru wylicza się niezależnie wartość błędu (wz. 5.2.3). Algorytmy uczenia (tj. modyfikacji wag) używają tylko wektorów ze zbioru uczącego, dzięki temu nie uczymy wszystkimi danymi wejściowymi. Wektory weryfikujące nie biorą udziału w modyfikowaniu wag sieci (czyli bezpośrednio nie trenują sieci), natomiast na jego podstawie, po zakończeniu każdej epoki uczenia, wyznaczany jest błąd uczenia. Proces ten sprawdza umiejętność generalizacji sieci, ponieważ wektory te nie biorą udziału w nauczaniu. Wartości błędu zbioru weryfikującego służą zatem do wewnętrznych decyzji algorytmu, w jaki sposób dobierać kolejne parametry uczenia sieci oraz kiedy przerwać uczenie – następuje przeuczenie sieci. Z taką sytuacją mamy do czynienia wtedy, gdy błąd wyznaczony na podstawie zbioru uczącego maleje, natomiast błąd na podstawie zbioru weryfikującego najpierw małał, a potem zaczyna rosnać. Zbiór testowy służy do ostatecznej oceny uzyskanej sieci.

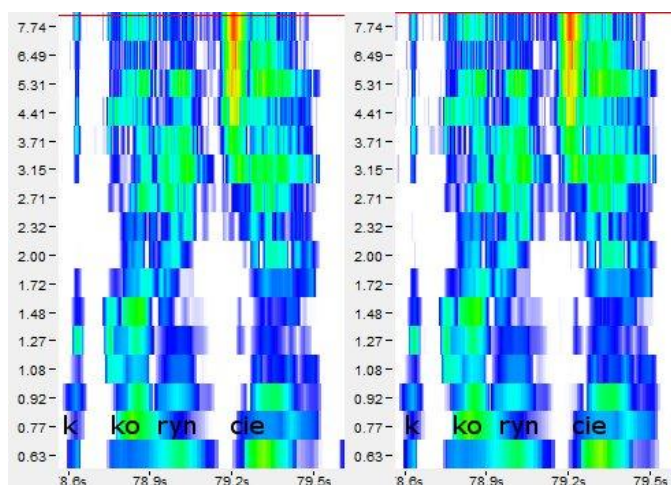
Aby zwiększyć wiarygodność otrzymywanych wyników, mimo iż STATISTICA używa tylko 50% wektorów do uczenia, postanowiono przekazać jej tylko część wektorów wejściowych. Sygnał wejściowy podzielono na 3 pliki o zbliżonym czasie trwania (liczbę niepełności w każdym pliku opisuje tab. 6.5-4), a następnie w pakiecie STATISTICA użyto tylko wektorów utworzonych z plików plik2 i plik3.

tab. 6.5-4 Liczba wystąpień powtórzeń głosek w materiale badawczym z podziałem na pliki

<b>plik</b>	<b>czas trwania</b>	<b>b</b>	<b>d</b>	<b>g</b>	<b>k</b>	<b>n</b>	<b>o</b>	<b>p</b>	<b>t</b>	<b>suma</b>
plik1	3min. 12s.	3		3	33			20	11	70
plik2	3min. 20s.	1	3		15			17	59	95
plik3	3min. 11s.	19	3	7	34	2	1	29	34	129
<b>Łącznie</b>	<b>9min. 43s.</b>	<b>23</b>	<b>6</b>	<b>10</b>	<b>82</b>	<b>2</b>	<b>1</b>	<b>66</b>	<b>104</b>	<b>294</b>

Zauważono, iż duży wpływ na wyniki ma algorytm detekcji początku i końca fonacji (opisany w rozdziale 6.3). To właśnie na podstawie fragmentów fonacji wycinane są odcinki CWT, z których tworzone są sekwencje wygrywających neuronów sieci Kohonena, używanych przez perceptron. Powtórzenia głosek są bardzo krótkie, często o małym natężeniu i czasami następują tak szybko po sobie, że można by uznać je za jedną, nieprzerwaną fonację – dlatego detekcja fonacji powinna być bardzo czuła. Jednocześnie należy unikać dzielenia płynnych wyrazów na kilka fragmentów fonacyjnych, więc nie może być zbyt czuła. Należy zatem wyciąć jak największą liczbę niepełności przy jak najmniejszym stopniu fragmentowania płynnych wyrazów. W związku z tym dobór parametrów detekcji fonacji znacząco wpływa na ilość poprawnie wyciętych powtórzeń głosek. Z tego powodu, przed właściwymi badaniami detekcji tego rodzaju niepełności,

przeprowadzono badania doboru parametrów algorytmu detekcji fonacji. Za istotne uznano czas, jaki dzieli wycinany fragment z następnym fragmentem oraz parametr odcięcia szumów (rys. 6.5:9).



rys. 6.5:9 Skalogramy CWT wypowiedzi „korynecie”: po lewej dla odcięcia szumów na poziomie -53dB, po prawej na poziomie -55dB. Ze względu na szumy dla wartości -55dB, pomiędzy powtórzeniem głoski ‘k’, pierwsza głoska ‘k’ potraktowana by była, jako część wyrazu ‘korynecie’ i nie zostałaby wycięta – tym samym powtórzenie głoski nie zostałoby rozpoznane.

Poprawność wyodrębniania fragmentów z przedłużeniami oraz powtórzeniami sylab, jako fonacji dużo dłuższych i o silniejszym natężeniu, nie musi być tak czułe – tym samym taka analiza była dla nich niepotrzebna.

Tak samo jak w przypadku skalogramów zawierających przedłużenia, tu również zauważono, że niskoczęstotliwościowe składowe skalogramu CWT bardzo często rozpoczynają się trochę przed fonacją i utrzymują się jeszcze po niej (rys. 6.5:4), utrudniając tym samym poprawną detekcję fonacji. Wyodrębnianie fonacji powtórzeń głosek musi być dużo bardziej precyzyjne w czasie, z tego powodu usunięto dwie dodatkowe skale niskoczęstotliwościowe zostawiając 16 skal barkowych ( $B=6,7, \dots, 21$ ).

### 6.5.2.3. Procedura detekcji powtórzeń głosek

Przyjęto następującą procedurę detekcji powtórzeń głosek (podkreślone elementy są parametrami algorytmu):

- 1) wczytaj plik dźwiękowy
- 2) oblicz współczynniki  $CWT^0$  (wz. 4.3.1.2) dla skal barkowych (wz. 4.5.1)
- 3) wykryj fragmenty mowy (fonacje) dla zadanego odcięcia szumów (poziom szumów) i zadanej minimalnej przerwy pomiędzy wyrazami (min przerwa) – dla szerokości

	okna 23,2ms i 50% przesunięcia okna <i>następnie każdy fragment podziel na okna o szerokości 23,2ms i 100% przesunięcia okna i ponownie oblicz wektory (wz. 6.2.1)</i>
4)	dla fonacji z plików plik2 i plik3 (czyli około 75% danych) krótszych niż 200ms:
5)	wytnij fragment $CWT^0$ z zadaniem <u>otoczeniem</u> ( <i>otocz wycinania</i> ) – każdy fragment z zaburzeniem składał się z 500 milisekundowego prefixu, następnie niepełności oraz postfixu odpowiedniej długości tak, aby łączna długość była równa zadanemu otoczeniu
6)	zredukuj $CWT^0$ do sekwencji indeksów wygrywających neuronów sieci Kohonena – na podstawie wyników wstępnych testów użyto tylko 16 skal barkowych: 6,7,..21 oraz tylko sieci 5x5 uczoną dla 100 epok przy wsp. uczenia 0.20-0.10 i wsp. sąsiedztwa 2.5-0.5.
7)	manualnie oznacz fragment jako płynny/niepłynny
8)	używając narzędzia „Intelligent Problem Solver” pakietu STATISTICA znajdź najlepszy perceptron 3-warstwowy dla danych z kroku 7) i wyeksportuj jego wagi do programu WaveBlaster
9)	dla 100% wyrazów krótszych niż 200ms:
10)	wytnij fragment $CWT^0$ i wygeneruj sekwencję indeksów wygrywających neuronów sieci Kohonena
11)	używając nauczonego perceptronu oznacz fragment jako płynny/niepłynny
12)	na podstawie odsłuchowych i automatycznych zaznaczeń wygeneruj statystyki wykrywalności

W kroku 3) zasadność użycia parametrów odcięcia szumów (*poziom szumów*) i minimalnej przerwy pomiędzy wyrazami (*min przerwa*) wynika z opisu w poprzednim rozdziale (patrz rys. 6.5:9).

W kroku 4) wartość 200ms została przyjęta na podstawie histogramu przedstawionego na rys. 6.5:5, natomiast pliki plik1, plik2, plik3 są opisane w tab. 6.5-3.

Jak wyjaśniono w rozdziale 6.5.1.2, liczba fonemów we fragmencie, a tym samym długość fragmentu ma duże znaczenie. Z tego powodu w kroku 5) otoczenie wycinanego

fragmentu mowy, czyli całkowita długość wycinanego odcinka (*otocz. wycinania*), jest parametrem algorytmu.

#### 6.5.2.4. Wyniki rozpoznawania powtórzeń głosek

W pierwszej serii badań sprawdzono trafność wyodrębniania niepełnych głosek – czyli krok 3) procedury detekcji powtórzeń głosek (rozdział 6.5.2.3). Nie badano zatem współczynników rozpoznawania niepełności, tylko ile zaburzonych głosek zostało poprawnie wyodrębnionych przez algorytm jako niezależne fonacje oraz ile przy tym zostało utworzonych fragmentów odpowiadających pełnym fonacjom.

tab. 6.5-5 Pierwsza seria testów. Liczba i procent wyodrębnionych powtórzeń, oraz fragmentów oznaczonych jako płynne dla plików plik1, plik2, plik3 dla parametrów G1 - poziom\_szumów, G2 - min\_przerwa

plik	liczba wyodrębnionych								
	powtórzeń głosek		płynnych fragm.	powtórzeń głosek		płynnych fragm.	powtórzeń głosek		płynnych fragm.
	<i>G1=55dB G2=50ms</i>			<i>G1=54dB G2=50ms</i>			<i>G1=53dB G2=50ms</i>		
plik1	50	71%	95	66	94%	123	65	93%	139
plik2	74	78%	92	83	87%	129	82	86%	139
plik3	98	76%	58	121	94%	132	116	90%	139
suma	<b>222</b>	<b>76%</b>	<b>245</b>	<b>270</b>	<b>92%</b>	<b>384</b>	<b>263</b>	<b>89%</b>	<b>417</b>
	<i>G1=55dB G2=40ms</i>			<i>G1=54dB G2=40ms</i>			<i>G1=53dB G2=40ms</i>		
plik1	56	80%	134	68	97%	144	67	96%	162
plik2	77	81%	126	84	88%	146	85	89%	154
plik3	104	81%	77	121	94%	146	116	90%	153
suma	<b>237</b>	<b>81%</b>	<b>337</b>	<b>273</b>	<b>93%</b>	<b>436</b>	<b>268</b>	<b>91%</b>	<b>469</b>
	<i>G1=55dB G2=30ms</i>			<i>G1=54dB G2=30ms</i>			<i>G1=53dB G2=30ms</i>		
plik1	56	80%	134	70	100%	165	69	99%	189
plik2	77	81%	126	91	96%	168	89	94%	178
plik3	104	81%	77	122	95%	161	117	91%	174
suma	<b>237</b>	<b>81%</b>	<b>337</b>	<b>283</b>	<b>96%</b>	<b>494</b>	<b>275</b>	<b>94%</b>	<b>541</b>
	<i>G1=55dB G2=0ms</i>			<i>G1=54dB G2=0ms</i>			<i>G1=53dB G2=0ms</i>		
plik1	59	84%	170	70	100%	199	70	100%	226
plik2	87	92%	168	93	98%	197	91	96%	209
plik3	98	76%	111	122	95%	188	118	91%	199
suma	<b>244</b>	<b>83%</b>	<b>449</b>	<b>285</b>	<b>97%</b>	<b>584</b>	<b>279</b>	<b>95%</b>	<b>634</b>

Wyniki badań zamieszczone w tab. 6.5-5 mają duże znaczenie w dalszym postępowaniu. Detekcja powtórzeń głosek operuje na wyodrębnionych fonacjach, więc jeżeli poprawnie wyodrębniono by tylko 60% wszystkich niepełnych głosek, można by było osiągać współczynniki rozpoznawania co najwyżej na poziomie 60%. Na podstawie tych wyników, w dalszej części badań, zdecydowano się wykorzystać konfigurację:



- *poziom\_szumów =55dB min\_przerwa =50ms* (najmniej wyodrębnionych nie płynności, ale również najmniej fragmentów płynnych),
- *poziom\_szumów =54dB min\_przerwa =50ms* (najlepszy kompromis pomiędzy liczbą wyodrębnionych nie płynności a liczbą płynnych fragmentów),
- *poziom\_szumów =54dB min\_przerwa =0ms* (najwięcej wyodrębnionych nie płynności, niestety bardzo dużo fragmentów płynnych).

W drugiej serii badań, za pomocą pakietu STATISTICA, szukano najlepszego 3-warstwowego perceptronu – czyli krok 8) procedury detekcji powtórzeń głosek (rozdział 6.5.2.3). Fragmenty były wycinane z *otocz\_wycinania* równym: 700ms, 1000ms, 1500ms, 2000ms, 2500ms, 3000ms. Każdy fragment z zaburzeniem zawierał 500 milisekundowy prefix, następnie nie płynność oraz postfix odpowiedniej długości tak, aby łączna długość była równa zadanemu otoczeniu. Poniższa tabelka przedstawia wyniki rozpoznawania sumaryczne oraz osobne dla każdego ze zbiorów uczący/weryfikujący/testowy (znaczenie zbiorów rozdział 6.5.2.2).

tab. 6.5-6 Współczynniki rozpoznawania fragmentów płynnych i fragmentów z powtórzeniami dla plików plik2 i plik3 przez sieci w pakiecie STATISTICA. G1 - *poziom szumów*, G2 - *min przerwa*, G3 - *otocz wycinania*. sieć MLP zawiera numer porządkowy oraz ilość neuronów w warstwach, algorytm uczenia: BP100 oznacza wsteczną propagację dla 100 epok, CG20b oznacza metodę gradientów dla 20 epok.

G3	sieć MLP		algorytm uczenia	G1 G2	łącznie[%]		uczący[%]		weryfikujący		testujący[%]	
					powt.	płynne	powt.	płynne	powt.	płynne	powt.	płynne
700 ms	1	31-130-1	BP29b	G1=50ms G2=50ms G1=55dB	98,2	97,7	99,4	99,0	97,6	96,4	96,5	96,4
1000 ms	2	44-91-1	BP100,CG20b		98,8	99,2	99,6	99,9	97,9	98,8	98,1	98,0
1500 ms	3	65-78-1	BP100,CG37b		99,5	98,0	100,0	99,8	99,4	96,5	99,8	96,5
2000 ms	4	87-87-1	BP100,CG28b		99,3	99,2	100,0	100,0	98,5	98,9	98,9	98,0
2500 ms	5	108-74-1	BP100,CG44b		99,7	99,3	100,0	100,0	99,2	99,4	99,7	97,9
3000 ms	6	130-130-1	BP100,CG15b		99,1	99,8	99,8	100,0	98,4	99,4	98,6	99,8
700 ms	7	31-130-1	BP33b	G1=50ms G2=50ms G1=54dB	97,1	97,7	97,9	99,0	95,5	96,2	97,0	96,5
1000 ms	8	44-91-1	BP100,CG20b		99,8	98,4	100,0	99,6	99,5	97,7	99,7	96,6
1500 ms	9	65-83-1	BP100,CG28b		99,4	99,5	100,0	100,0	98,5	98,9	99,0	99,2
2000 ms	10	87-74-1	BP100,CG55b		99,8	98,6	100,0	100,0	99,6	96,7	99,7	97,7
2500 ms	11	108-100-1	BP100,CG42b		99,8	98,8	100,0	100,0	99,4	97,5	99,9	97,5
3000 ms	12	130-98-1	BP100,CG49b		99,5	99,7	100,0	100,0	99,2	99,3	99,0	99,5
700 ms	13	31-130-1	BP14b	G1=30ms G2=30ms G1=54dB	97,3	98,0	98,1	99,1	96,1	96,5	96,8	97,4
1000 ms	14	44-130-1	BP30b		98,8	98,8	99,8	99,7	97,7	98,0	98,0	97,7
1500 ms	15	65-101-1	BP100,CG25b		99,3	99,4	100,0	100,0	98,6	98,2	98,5	99,6
2000 ms	16	87-99-1	BP100,CG42b		99,9	97,7	100,0	100,0	99,8	95,1	99,7	95,9
2500 ms	17	108-130-1	BP95b		99,9	98,2	100,0	100,0	99,8	96,5	99,8	96,4
3000 ms	18	130-98-1	BP100,CG56b		100,0	97,8	100,0	100,0	99,9	94,6	99,9	96,9

Jak widać wszystkie sieci idealnie różnicują wektory z grup płynne/niepłynne – również w zbiorze testującym. Dzieje się tak dzięki algorytmowi wyodrębniania fonacji – wszystkie wektory niepłynne podawane na perceptron miały fragment niepłynny w tym samym miejscu. Dzięki tak klarownym danym wejściowym, perceptron nie musiał dodatkowo wyuczać się rozpoznawania niepłynności w dowolnym miejscu wektora wejściowego. To dowodzi zasadności używania w/w algorytmu.

Po wyeksportowaniu wszystkich 18 modeli z pakietu STATISTICA do programu WaveBlaster, przeprowadzono trzecią serię badań tj. detekcję niepłynności na całym zbiorze wejściowym przy użyciu nauczonych perceptronów – czyli krok 11) procedury detekcji powtórzeń głosek (rozdział 6.5.2.3).

tab. 6.5-7 Wyniki rozpoznawanie powtórzeń głosek w mowie ciągłej dla 18 sieci MLP z tabeli 6.5-5. *czul, przew* (wz. 6.4.1), *G1 - poziom\_szumów, G2 - min\_przerwa, G3 - otocz\_wycinania*

G3	sieć	G1/G2	plik1		plik2		plik3		łącznie	
			<i>czul</i>	<i>przew</i>	<i>czul</i>	<i>przew</i>	<i>czul</i>	<i>przew</i>	<i>czul</i>	<i>przew</i>
700 ms	1	<i>G1=55dB</i> <i>G2=50ms</i>	71%	79%	75%	98%	75%	95%	74%	92%
1000 ms	2		64%	71%	76%	97%	76%	99%	73%	91%
1500 ms	3		72%	66%	77%	94%	76%	95%	75%	86%
2000 ms	4		67%	68%	77%	98%	75%	98%	74%	89%
2500 ms	5		74%	67%	77%	96%	76%	96%	76%	87%
3000 ms	6		60%	65%	77%	98%	76%	99%	73%	89%
700 ms	7	<i>G1=54dB</i> <i>G2=50ms</i>	52%	74%	85%	93%	92%	93%	80%	89%
1000 ms	8		62%	75%	87%	93%	93%	97%	84%	91%
<b>1500 ms</b>	<b>9</b>		<b>70%</b>	<b>83%</b>	<b>87%</b>	<b>98%</b>	<b>93%</b>	<b>98%</b>	<b>86%</b>	<b>95%</b>
2000 ms	10		72%	70%	87%	96%	93%	93%	86%	88%
2500 ms	11		75%	72%	87%	96%	93%	96%	87%	90%
3000 ms	12		60%	71%	87%	96%	93%	96%	83%	91%
700 ms	13	<i>G1=54dB</i> <i>G2=30ms</i>	58%	67%	94%	93%	93%	93%	85%	87%
1000 ms	14		61%	76%	94%	98%	92%	98%	85%	93%
1500 ms	15		67%	75%	94%	97%	93%	98%	87%	92%
2000 ms	16		44%	72%	91%	98%	90%	99%	79%	94%
2500 ms	17		60%	72%	93%	100%	91%	100%	84%	94%
3000 ms	18		51%	69%	88%	100%	93%	100%	81%	94%

Wyniki dla plików plik2 i plik3 są zgodne z oczekiwaniami. Perceptrony rozpoznają bardzo dobrze, ale jak napisano wcześniej w tym akapicie, wyniki rozpoznawania mogą być co najwyżej tak dobre jak trafność wyodrębniania fragmentów niepełnych. Zgodnie z tab. 6.5-5 konfiguracja:

- *poziom\_szumów=55dB min\_przerwa=50ms* wyodrębniła poprawnie 76% z wszystkich niepełności
- *poziom\_szumów =54dB min\_przerwa =50ms* wyodrębniła poprawnie 92% z wszystkich niepełności
- *poziom\_szumów =54dB min\_przerwa =0ms* wyodrębniła poprawnie 97% z wszystkich niepełności

Zatem wyniki dla plik2 i plik3 oscylują wokół w/w wartości.

Wyniki rozpoznawania dla pliku pierwszego powinny być równie dobre. Po dokładniejszej analizie okazało się, że plik ten zawiera kilka bardzo szybkich serii wielokrotnych powtórzeń (jak „p p p p p poszedł”). O ile algorytm detekcji fonacji poradził sobie z ich wyodrębnieniem, o tyle perceptron już nie poradził sobie z ich rozpoznaniem. Perceptron zawiódł dla wszystkich powtórzeń w poszczególnych seriach, dlatego każda seria

zmniejszyła liczbę detekcji o kilka elementów – przy tak małym pliku, 2-3 serie (każda po 4-5 powtórzeń) drastycznie obniżyły współczynnik rozpoznawania. Widać zatem, że o ile większość powtórzeń jest rozpoznawana doskonale, o tyle dla bardzo szybko powtarzanych głosek (tj. serii), które niemalże zlewają się w jedną nieprzerwaną fonację (praktycznie nie ma między nimi ciszy) należałoby opracować inny algorytm.

Najlepszy uzyskany wynik rozpoznawania dla wszystkich plików wynosi: *czul=86%*, *przew=95%*.

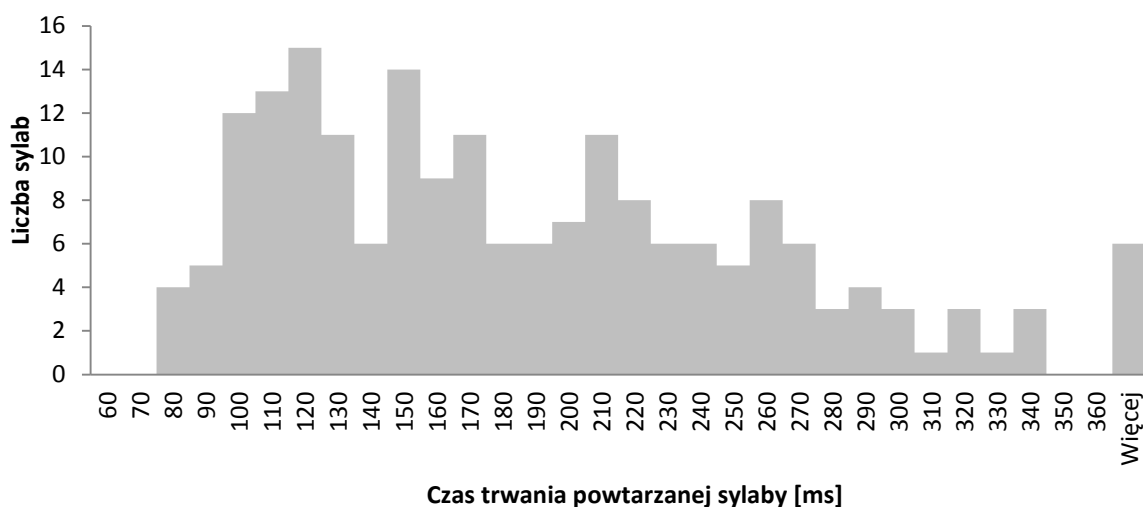
### 6.5.3. Metoda detekcji powtórzeń sylab

#### 6.5.3.1. Materiał dźwiękowy

Jako materiał badawczy posłużyły nagrania pięciu osób jaskających się (jedna kobieta, dwóch mężczyzn oraz dziewczynka i chłopiec). W nagraniach, metodą odsłuchową (posiłkując się wykresem oscylogramu oraz spektrogramu STFT), odnaleziono wszystkie powtórzenia a następnie wycięto je z otoczeniem mowy płynnej (łącznie mający długość 4 sekundy). Wszystkie fragmenty zostały połączone tworząc jedną wypowiedź trwającą 5 min. 26 s. Statystyki liczby powtórzeń przedstawia tab. 6.5-8. Stworzono również histogramy czasów trwania powtarzanych sylab (rys. 6.5:10) oraz przerw pomiędzy powtórzeniami (rys. 6.5:11):

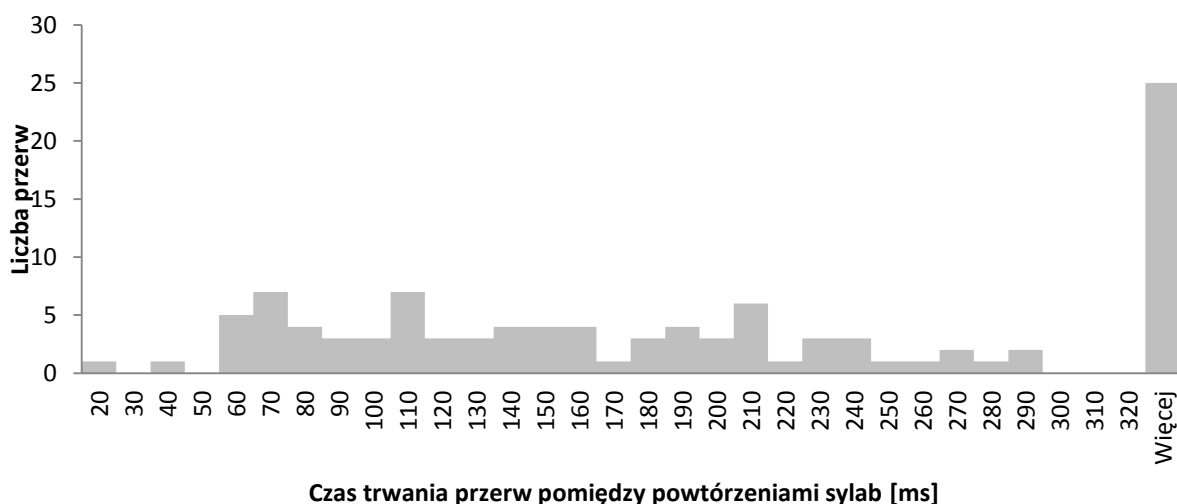
tab. 6.5-8 Liczba powtórzeń sylab w materiale badawczym.

liczba	typ sylaby
1	am, bo, co, dio, dol, dzie, e, go, ja, ni, o, od, pa, pos, pra, prz, prze, ra, sa, są, tam, te, tro, wo, zje, zo
2	ba, by, chło, do, im, wie, zie, zni, że
3	be, je, kie, ta
4	ko, mo, na, nie, wy
6	aa
8	ka, ma, po
	<b>Łącznie powtórzeń sylab: 106</b>
	<b>Łączna liczba fonacji: ~570</b>



rys. 6.5:10 Histogram czasu trwania powtarzanej sylaby

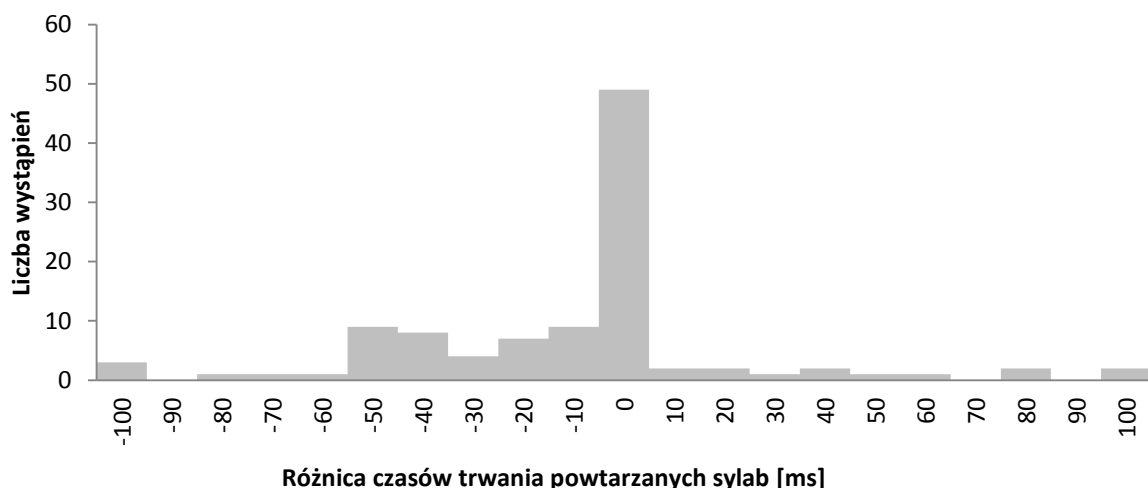
Ze statystyk wynika, że zaburzone sylaby trwają od 77 do 455ms.



rys. 6.5:11 Histogram czasów trwania przerw między powtórzeniami sylab

Najkrótsza przerwa trwa 18 ms.

Sprawdzanie powtórzeń sylab zawsze potrzebuje dwóch fragmentów fonacji – pierwszej i drugiej sylaby, dlatego stworzono histogram „różnic czasów trwania między pierwszą a drugą powtarzaną sylabą” (rys. 6.5:12). Jeżeli drugi fragment jest krótszy to różnica jest ujemna, jeżeli drugi fragment jest dłuższy to różnica jest dodatnia (czyli od długości drugiej sylaby odejmuje się długość pierwszej).



rys. 6.5:12 Histogram różnic czasów trwania między pierwszą a drugą powtarzaną sylabą

Ze statystyk wynika, że w badanym materiale druga sylaba może być co najwyżej krótsza o 143 ms, a dłuższa co najwyżej o 95 ms.

#### 6.5.3.2. Analiza wyników parametryzacji

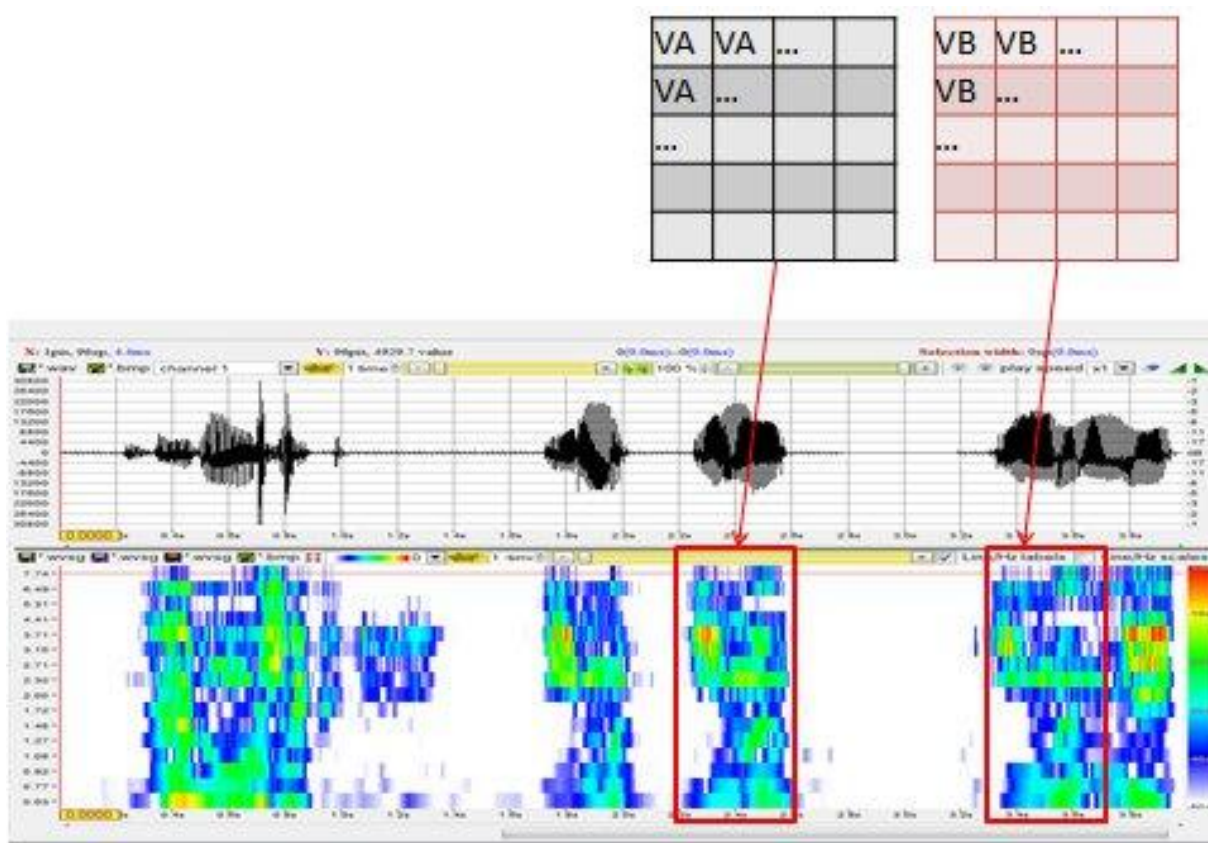
Na podstawie histogramu „czasów trwania powtarzanych sylab” (rys. 6.5:10), przyjęto iż długość sylaby może się wahać pomiędzy 70 a 500ms.

Wprowadzono za to inny parametr – minimalna różnica czasów powtarzanych sylab (rys. 6.5:12). Uznano, że jej najmniejsza wartość równa -143ms to przypadek odosobniony i przyjęto minimalną granicę równą -100ms. Mimo iż druga sylaba była co najwyżej dłuższa o 95 ms od pierwszej, uznano, że druga sylaba może być początkiem całego wyrazu, który może być bardzo długi – więc nie wprowadzono górnego ograniczenia dla tego parametru.

Powtarzane sylaby generują podobne wartości współczynników CWT, dlatego oczekiwano, że odpowiadające im sekwencje indeksów wygrywających neuronów sieci Kohonena będą również zbliżone. Dzięki temu, traktując w/w sekwencje indeksów jako wektory n-wymiarowe, będzie można wyliczać odległość między nimi (powinna być mała) lub wyliczać dla nich wartość korelacji (powinna być wysoka). Niestety, mimo podobieństw na poziomie współczynników CWT, nie udało się uzyskać satysfakcjonujących wyników na podstawie w/w wektorów. W tej sytuacji zdecydowano się na korelowanie wartości skalogramu CWT (bez udziału sieci Kohonena). W tym celu, po wykryciu wszystkich fonacji i pozostawieniu tylko sąsiadujących par sylab spełniających w/w kryteria – dostosowano długość drugiego fragmentu w każdej parze (zwiększono go poprzez doklejenie sygnału znajdującego się zaraz za nim lub zmniejszono poprzez obcięcie końca tej fonacji), aby był

równy długości pierwszego wyrazu. W ten sposób oba fragmenty trwają tyle samo – mają tyle samo okien i zawierają tyle samo skal w każdym oknie (patrz rys. 6.5:13).

Tak jak w przypadku detekcji powtórzeń głosek, zastosowano 16 skal barkowych ( $B=6,7,\dots,21$ ).



rys. 6.5:13 Prezentacja sposobu wyodrębniania sąsiadujących sylab w wypowiedzi „garnuszek im im im mniej”. Po wyodrębnieniu, pierwsza sylaba  $VA$  i druga sylaba  $VB$  są reprezentowane przez tą samą liczbę skal (wiersze) i okien (kolumny). Zrzut ekranu z programu WaveBlaster.

Następnie dla każdej pary tak skonstruowanych wektorów  $VA$  i  $VB$  wyliczono wartość korelacji [67]:

$$C = \frac{\sum_{j=0}^{L-1} \sum_{i=0}^{N-1} (VA_{ij} - \overline{VA})(VB_{ij} - \overline{VB})}{\sqrt{\sum_{j=0}^{L-1} \sum_{i=0}^{N-1} (VA_{ij} - \overline{VA})^2 \sum_{j=0}^{L-1} \sum_{i=0}^{N-1} (VB_{ij} - \overline{VB})^2}}$$

wz. 6.5.2

(6.5-2)

gdzie:

$L$  – liczba skal,

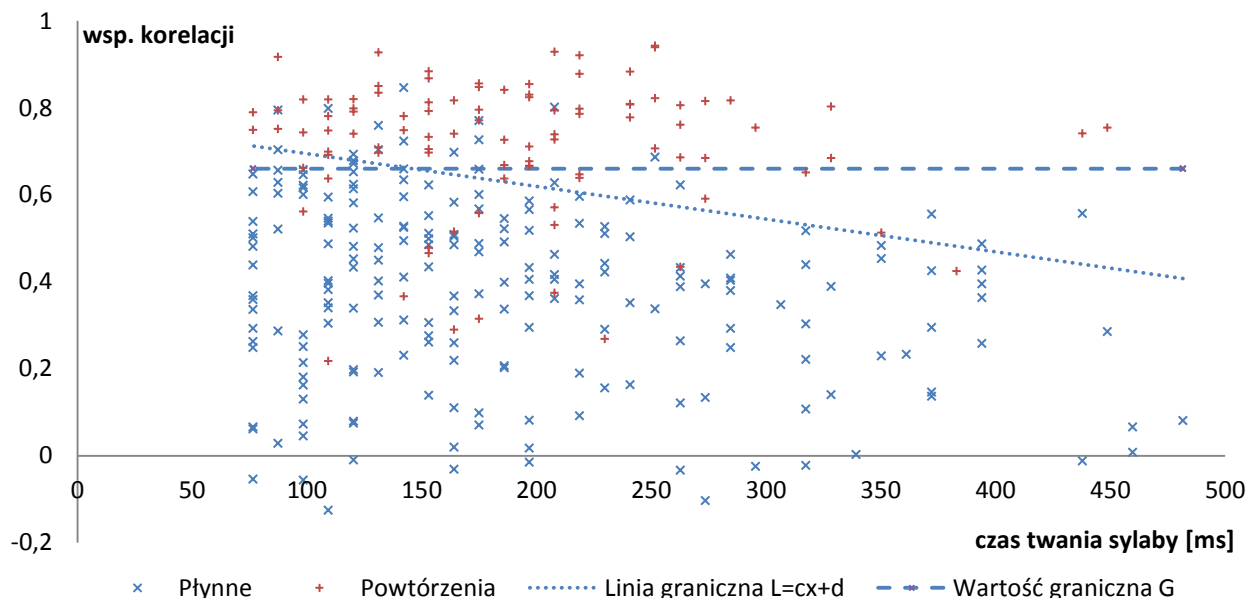
$N$  – liczba okien

$VA_{ij}, VB_{ij}$  – wartość  $CWT^0$  we fragmencie A / B dla j-tej skali w i-tym oknie,

$\overline{VA}, \overline{VB}$  – wartość średnia wszystkich wyrażen  $VA_{ij} / VB_{ij}$ .

Poniżej przedstawiono przykładowy wykres (rys. 6.5:14) współczynników korelacji, wyliczonych dla sylab wyodrębnionych w sposób opisany wyżej. Na rysunek naniesiono również wartość graniczną  $G=0.66$  najoptymalniej oddzielającą zbiór płynny od niepłynnego. Aby ją wyznaczyć, należy dla każdego punktu na wykresie wyznaczyć linię poziomą przechodzącą przed ten punkt – reprezentuje ona graniczną wartość współczynnika korelacji. Następnie należy zliczyć ilość poprawnie pogrupowanych punktów, tj. liczbę współczynników korelacji większych od tej granicy odpowiadających powtórzeniom niepłynnym oraz liczbę współczynników korelacji mniejszych od tej granicy odpowiadających powtórzeniom płynnym (czyli sąsiadujące sylaby są różne). Za wartość  $G$  przyjmujemy wartość graniczną współczynnika korelacji, dla którego liczba punktów poprawnie pogrupowanych jest największa.

Zauważono, że współczynniki korelacji dla powtórzeń sylab maleją wraz z długością sylab, dlatego postanowiono również wyznaczać przybliżoną optymalną linię graniczną  $L=cx+d$ , w tym przypadku równą:  $L= -0.00000340287x + 0.769797$ . Zbiór badanych prostych otrzymujemy generując proste przechodzące przez każdą parę punktów wykresu. Metoda wyznaczania najlepszej prostej jest taka sama jak dla wartości  $G$ .



rys. 6.5:14 Wykres wartości współczynników korelacji dla kolejnych sylab płynnych x oraz niepłynnie powtórzonych +. Na wykresie wyznaczona optymalna wartość graniczna oraz linia graniczna.



### 6.5.3.3. Procedura detekcji powtórzeń sylab

Przyjęto następującą procedurę detekcji powtórzeń sylab (podkreślone elementy są parametrami algorytmu):

1)	wczytaj plik dźwiękowy
2)	oblicz współczynniki $CWT^0$ (wz. 4.3.1.2) dla zadanych <u>skal barkowych</u> ( <i>skale barkowe</i> ) i falki Morleta o zadanej <u>częstotliwości środkowej</u> ( $F_c$ )
3)	wykryj fragmenty mowy (fonacje) dla szerokości okna 23,2ms i 50% przesunięcia okna  <i>następnie każdy fragment podziel na okna o szerokości 23,2ms i 100% przesunięcia okna i ponownie oblicz <math>CWT^0</math></i>
4)	dla każdej pary sąsiadujących fragmentów z których: <ul style="list-style-type: none"> <li>• pierwszy jest dłuższy niż 70ms i krótszy niż 500ms</li> <li>• drugi jest krótszy od pierwszego o co najwyżej 100ms</li> </ul>
5)	wyodrębnij odpowiadający fragment $CWT^0$ dla tych wyrazów dla zadanego <u>odcięcia szumów</u> ( <i>odc_szumów</i> )
6)	oblicz ich współczynnik korelacji (wz. 6.5.2)
7)	wyznacz optymalną wartość graniczną $G$ oraz linię graniczną $L=cx+d$ korelacji (rys. 6.5:14), oddzielającą płynne i niepłynne powtórzenia sylab
8)	dla tak wyznaczonej granicy korelacji, oznacz automatycznie pary sylab jako płynne/niepłynne
9)	na podstawie odsłuchowych i automatycznych zaznaczeń wygeneruj statystyki wykrywalności

W tej serii badań autor pracy skupił się nad parametryzacją samego CWT, dlatego wprowadzono zmienne skale (*skale\_barkowe*) oraz zmienną częstotliwość środkową ( $F_c$ ) w kroku 2).

Ograniczenia czasowe w kroku 4) wynikają z analizy histogramów opisanych w rozdziale 6.5.3.2.

Korelujemy całe fragmenty fonacji (sylaby), dlatego stwierdzono, że czułość wyodrębniania wyrazów (tj. dokładność wyszukiwania początku i końca fonacji) może mieć znaczenie. Z tego powodu wprowadzono parametr odcięcia szumów (*odc\_szumów*) w kroku 5).

#### 6.5.3.4. Wyniki rozpoznawania powtórzeń sylab

W pierwszej serii testów sprawdzono wpływ następujących parametrów na wyniki rozpoznawania:

- poziom odcięcia szumów (*odc\_szumów*) dla wartości: *-53dB*, *-54dB*, *-55dB*, *-56dB*, *-57dB*, *-58dB*
- liczbę skal barkowych (*skale\_barkowe*) dla opcji:
  - *bez\_0\_0* :  $B=1..22 \Rightarrow \sim 120\text{Hz} - \sim 10200\text{Hz}$ ,
  - *bez\_1\_0* :  $B=1..21 \Rightarrow \sim 120\text{Hz} - \sim 8000\text{Hz}$ ,
  - *bez\_1\_1* :  $B=2..21 \Rightarrow \sim 200\text{Hz} - \sim 8000\text{Hz}$ ,
  - *bez\_1\_2* :  $B=3..21 \Rightarrow \sim 300\text{Hz} - \sim 8000\text{Hz}$ ,
  - *bez\_1\_3* :  $B=4..21 \Rightarrow \sim 400\text{Hz} - \sim 8000\text{Hz}$ ,
  - *bez\_1\_4* :  $B=5..21 \Rightarrow \sim 510\text{Hz} - \sim 8000\text{Hz}$ ,
  - *bez\_1\_5* :  $B=6..21 \Rightarrow \sim 630\text{Hz} - \sim 8000\text{Hz}$ ,
  - *bez\_1\_6* :  $B=7..21 \Rightarrow \sim 760\text{Hz} - \sim 8000\text{Hz}$ ,

gdzie *bez\_X\_Y* oznacza pominięcie X skrajnych wysoko-częstotliwościowych skal i Y skrajnych nisko-częstotliwościowych skal (na przykład *bez\_1\_0* oznacza pominięcie skali barkowej 22, *bez\_1\_3* oznacza pominięcie skal barkowych 22, 3, 2, 1).

tab. 6.5-9 Wyniki automatycznego rozpoznawania powtórzeń sylab – seria pierwsza dla  $F_c=20\text{Hz}$ . czul, przew, P, B – patrz wz. 6.4.1; G, c, d – patrz rys. 6.5:14; S1 – skale\_barkowe, S2 – odc\_szumów.

S2	S1	G	P	B	czul	przew	c	d	P	B	czul	przew
-53dB	bez_0_0	0,7	74	28	70%	73%	-2,94635E-05	0,784183	88	43	83%	67%
	bez_1_0	0,71	72	23	68%	76%	-2,56415E-05	0,766188	88	43	83%	67%
	bez_1_1	0,69	75	25	71%	75%	-1,92166E-05	0,777467	77	22	73%	78%
	bez_1_2	0,67	74	27	70%	73%	-3,05874E-05	0,797238	76	19	72%	80%
	bez_1_3	0,64	80	28	75%	74%	-2,11768E-05	0,763394	74	14	70%	84%
	bez_1_4	0,66	74	18	70%	80%	-0,000024416	0,748644	77	15	73%	84%
	<b>bez_1_5</b>	<b>0,65</b>	<b>74</b>	<b>15</b>	<b>70%</b>	<b>83%</b>	<b>-3,35757E-05</b>	<b>0,733708</b>	<b>85</b>	<b>25</b>	<b>80%</b>	<b>77%</b>
	bez_1_6	0,66	72	15	68%	83%	-2,64775E-05	0,772354	80	27	75%	77%
-54dB	bez_0_0	0,7	78	31	74%	72%	-1,82942E-05	0,761543	86	38	81%	69%
	bez_1_0	0,7	76	31	72%	71%	-2,94212E-05	0,784531	88	43	83%	67%
	bez_1_1	0,69	77	27	73%	74%	-2,83063E-05	0,834954	73	16	69%	82%
	bez_1_2	0,68	75	27	71%	74%	-3,23934E-05	0,811601	78	21	74%	79%
	bez_1_3	0,66	78	22	74%	78%	-2,08754E-05	0,776850	74	12	70%	86%
	bez_1_4	0,66	76	20	72%	79%	-2,26489E-05	0,743577	80	18	75%	82%
	<b>bez_1_5</b>	<b>0,66</b>	<b>74</b>	<b>14</b>	<b>70%</b>	<b>84%</b>	<b>-2,78401E-05</b>	<b>0,764261</b>	<b>78</b>	<b>13</b>	<b>74%</b>	<b>86%</b>
	bez_1_6	0,66	74	19	70%	80%	-2,40156E-05	0,726125	74	15	70%	83%
-55dB	bez_0_0	0,71	76	29	72%	72%	-0,000025202	0,789729	87	40	82%	69%
	bez_1_0	0,71	76	29	72%	72%	-1,78139E-05	0,76548	86	38	81%	69%
	bez_1_1	0,7	76	26	72%	75%	-1,58039E-05	0,779944	77	21	73%	79%
	bez_1_2	0,69	75	26	71%	74%	-2,09902E-05	0,750872	85	32	80%	73%
	bez_1_3	0,67	78	23	74%	77%	-3,90384E-05	0,810985	83	25	78%	77%
	bez_1_4	0,67	76	19	72%	80%	-2,92527E-05	0,771779	80	17	75%	82%
	<b>bez_1_5</b>	<b>0,66</b>	<b>77</b>	<b>15</b>	<b>73%</b>	<b>84%</b>	<b>-1,54502E-05</b>	<b>0,720302</b>	<b>79</b>	<b>13</b>	<b>75%</b>	<b>86%</b>
	bez_1_6	0,67	76	18	72%	81%	-2,99003E-05	0,762243	79	16	75%	83%
-56dB	bez_0_0	0,72	76	28	72%	73%	-2,37818E-05	0,790665	87	40	82%	69%
	bez_1_0	0,72	76	27	72%	74%	-1,83487E-05	0,771577	86	39	81%	69%
	bez_1_1	0,72	75	24	71%	76%	-2,60678E-05	0,84059	73	15	69%	83%
	bez_1_2	0,68	79	33	75%	71%	-2,07705E-05	0,757224	85	32	80%	73%
	bez_1_3	0,68	78	22	74%	78%	-3,55817E-05	0,84365	75	13	71%	85%
	bez_1_4	0,68	76	18	72%	81%	-0,00003051	0,780063	81	17	76%	83%
	<b>bez_1_5</b>	<b>0,67</b>	<b>77</b>	<b>15</b>	<b>73%</b>	<b>84%</b>	<b>-3,68778E-05</b>	<b>0,811599</b>	<b>80</b>	<b>13</b>	<b>75%</b>	<b>86%</b>
	bez_1_6	0,68	76	20	72%	79%	-3,24334E-05	0,797230	77	16	73%	83%
-57dB	bez_0_0	0,73	76	28	72%	73%	-2,59823E-05	0,800018	87	42	82%	67%
	bez_1_0	0,73	75	27	71%	74%	-2,79983E-05	0,804565	88	44	83%	67%
	bez_1_1	0,73	75	24	71%	76%	-0,000026968	0,840283	78	21	74%	79%
	bez_1_2	0,68	81	34	76%	70%	-2,74772E-05	0,821556	78	18	74%	81%
	bez_1_3	0,69	78	22	74%	78%	-0,000022343	0,807316	74	12	70%	86%
	bez_1_4	0,68	77	19	73%	80%	-2,88979E-05	0,810073	78	13	74%	86%
	<b>bez_1_5</b>	<b>0,66</b>	<b>81</b>	<b>20</b>	<b>76%</b>	<b>80%</b>	<b>-3,25545E-05</b>	<b>0,806083</b>	<b>80</b>	<b>13</b>	<b>75%</b>	<b>86%</b>
	bez_1_6	0,67	79	19	75%	81%	-2,87840E-05	0,816478	78	13	74%	86%
-58dB	bez_0_0	0,73	77	32	73%	71%	-2,39623E-05	0,793158	88	43	83%	67%
	bez_1_0	0,75	72	24	68%	75%	-0,00002631	0,799879	89	45	84%	66%
	bez_1_1	0,72	77	30	73%	72%	-2,77439E-05	0,847547	78	22	74%	78%
	bez_1_2	0,7	79	31	75%	72%	-2,61473E-05	0,825186	78	19	74%	80%
	bez_1_3	0,69	79	24	75%	77%	-3,39391E-05	0,835328	80	20	75%	80%
	bez_1_4	0,66	82	26	77%	76%	-0,000032016	0,829123	79	14	75%	85%
	<b>bez_1_5</b>	<b>0,67</b>	<b>81</b>	<b>21</b>	<b>76%</b>	<b>79%</b>	<b>-6,57913E-06</b>	<b>0,699621</b>	<b>83</b>	<b>17</b>	<b>78%</b>	<b>83%</b>
	bez_1_6	0,67	79	23	75%	77%	-5,91344E-06	0,775229	80	17	75%	82%

W tab. 6.5-9 widać, że najlepsze rezultaty uzyskujemy zawsze dla parametru *skale\_barkowe=bez\_1\_5* ( $B=6..21 \Rightarrow \sim 630\text{Hz} - \sim 8000\text{Hz}$ ). Dla *odc\_szumów* równemu:  $-55\text{dB}$ ,  $-56\text{dB}$ ,  $-57\text{dB}$ ,  $-58\text{dB}$  wyniki są zbliżone do siebie (ewentualny wzrost czułości *czul* odbywa się proporcjonalnym spadkiem przewidywalności *przed*). Na tej podstawie przeprowadzono drugą serię badań dla konfiguracji:

- *skale\_barkowe=bez\_1\_5, odc\_szumów=-55dB*,
- *skale\_barkowe=bez\_1\_5, odc\_szumów=-58dB*,
- *skale\_barkowe=bez\_1\_5, odc\_szumów=-60dB*.

Dla każdej z w/w konfiguracji przetestowano 80 różnych częstotliwości środkowych falki Morleta (patrz wz. 4.2.1), tj.  $F_c = 1\text{Hz}, 1.5\text{Hz}, 2\text{Hz}, 2.5\text{Hz}, \dots, 39.5\text{Hz}, 40\text{Hz}$ . Z powodu dużej liczby wyników (3 konfiguracje x 80 częstotliwości = 240 wierszy) przedstawiono tylko najlepsze rezultaty:

tab. 6.5-10 Wyniki automatycznego rozpoznawania powtórzeń sylab – seria druga. *czul, przew, P, B* – wz. 6.4.1; *G, c, d* – patrz rys. 6.5:14; *S1* – *skale\_barkowe*, *S2* – *odc\_szumów*.

konfiguracja S2/S1	Fc	G	P	B	czul	przew	c	D	P	B	czul	przew
-55dB bez_1_5	21,5	0,64	81	22	76%	79%	-3,42651E-05	0,804401	81	12	76%	87%
	22	0,67	78	15	74%	84%	-2,13996E-05	0,747141	83	15	78%	85%
	27,5	0,64	77	12	73%	87%	-2,84418E-05	0,719641	83	16	78%	84%
	28	0,62	82	24	77%	77%	-3,94934E-05	0,817231	78	10	74%	89%
	34	0,65	77	14	73%	85%	-0,000022104	0,7301	77	10	73%	89%
-58dB bez_1_5	19	0,63	85	28	80%	75%	-1,58883E-05	0,713655	83	17	78%	83%
	21,5	0,63	88	34	83%	72%	-1,98112E-05	0,794358	77	9	73%	90%
	22	0,68	80	21	75%	79%	-2,31846E-05	0,788836	81	15	76%	84%
	27,5	0,68	76	10	72%	88%	-1,90554E-05	0,745788	80	9	75%	90%
	28	0,66	81	22	76%	79%	-3,87081E-05	0,834098	80	12	75%	87%
	34	0,67	78	14	74%	85%	-7,6241E-06	0,686152	82	16	77%	84%
-60dB bez_1_5	19	0,7	76	15	72%	84%	-1,46308E-05	0,726024	84	18	79%	82%
	21,5	0,68	83	25	78%	77%	-3,42921E-05	0,849908	83	17	78%	83%
	27,5	0,64	85	22	80%	79%	-1,14104E-05	0,728771	81	12	76%	87%
	28	0,66	83	28	78%	75%	-3,70733E-05	0,855078	78	11	74%	88%
	34,5	0,6	87	33	82%	73%	-4,88875E-05	0,888054	75	8	71%	90%
	<b>39,5</b>	<b>0,63</b>	<b>84</b>	<b>18</b>	<b>79%</b>	<b>82%</b>	<b>-2,01145E-06</b>	<b>0,635653</b>	<b>86</b>	<b>17</b>	<b>81%</b>	<b>83%</b>

Po drugiej serii testów (tab. 6.5-10) widać, że najlepsze rezultaty otrzymujemy dla  $F_c$  w okolicach  $21\text{Hz}$ ,  $28\text{Hz}$ ,  $34\text{Hz}$  i  $40\text{Hz}$ . Najlepszy wynik  $sens=81\%$ ,  $pred=83\%$  uzyskano dla wartości  $F_c=39,5\text{Hz}$ .

W wynikach obydwu serii widać również, że stosowanie linii granicznej  $L$ , zamiast wartości granicznej  $G$ , ma sens, gdyż zwiększa oba współczynniki *czułości* i *przewidywalności* o kilka procent.

## 7. Program do analizy i rozpoznawania niepełności mowy

Autor rozprawy stworzył program „WaveBlaster” – w ogromnej mierze ułatwiający proces badawczy.

Funkcjonalność prezentowanego programu obejmuje wiele algorytmów, paneli konfiguracyjnych i wizualizujących wyniki tych algorytmów, paneli umożliwiających wczytywanie oraz zapis wszelakich danych wejściowych, wyjściowych i konfiguracyjnych. Duża część tak rozbudowanej aplikacji była tworzona na potrzeby badań prowadzonych pod kątem tej rozprawy – mimo, że ostatecznie wykorzystano tylko niewielką jej część do napisania tej pracy (obrazuje to jak wiele analiz i opcji było sprawdzanych, zanim autor uzyskał satysfakcjonujące wyniki rozpoznawania niepełności w mowie ciągłej). Dzięki temu, że WaveBlaster zawiera różnorakie algorytmy z bogatą liczbą opcji do ich konfiguracji (DWT/CWT z dużą ilością falek, analiza Fouriera, analiza liniowej predykcji, generowanie modelu traktu głosowego, detekcja formantów, wyliczanie filtrów tercjowych, mapy Kohonena, algorytmy pre-empfazy, korelacji, k-Means, wyliczanie energii i obwiedni sygnału) – program ten stał się ogólnym narzędziem do analizy sygnałów (na przykład sygnału EMG [45]). Mimo, iż stworzenie narzędzia WaveBlaster pochłonęło bardzo dużo czasu, przyspieszyło to (lub wręcz umożliwiło) uzyskanie satysfakcjonujących wyników badań. Dzięki łatwej i szybkiej obsłudze, przeprowadzono nim niezliczoną ilość eksperymentów (analizy i wyniki przedstawione w tabelach tej rozprawy stanowią niewielką ich część).

Wszystkie komponenty programu, mimo iż dość liczne, są ze sobą powiązane. Dzięki temu szukając zależności, regularności w badanym sygnale, możemy oglądać dane z wielu perspektyw (powiększanie, pomniejszanie, przesuwanie, zmiana kolorów i skal). Dokładny system skal i miar jest bardzo przydatny w poszukiwaniu/oglądaniu choćby najmniejszych szczegółów. Wszystkie wykresy są sprzężone ze sobą (powiększanie, pomniejszanie, przesuwanie, zaznaczanie), w związku z tym, dla wybranego fragmentu możemy jednocześnie oglądać i porównywać wyniki wielu analiz (oscylogram, spektrogramy, widma, obrys sieci Kohonena) – co jest bardzo przydatną funkcjonalnością.

Prawie wszystkie badania przedstawione w niniejszej rozprawie zostały przeprowadzone z wykorzystaniem tego programu. Wyjątek stanowi wyszukiwanie najlepszego perceptronu przy detekcji powtórzeń głosek, gdzie dodatkowo użyto pakietu STATISTICA.

## 8. Podsumowanie

### 8.1. *Przegląd użytych metod*

Autor pracy opracował algorytmy rozpoznawania trzech rodzajów nie płynności, najczęściej badanych przez naukowców (wnioski z rozdziału 3 opisującego obecny stan badań), tj. przedłużeń, powtórzeń głosek, powtórzeń sylab. Dla każdego rodzaju opracowano odrębną procedurę rozpoznawania. We wszystkich procedurach:

- sygnał dźwiękowy najpierw parametryzowano ciągłą transformatą falkową przy wykorzystaniu falki Morlet20 (Morlet39,5 dla powtórzeń sylab) oraz skal barkowych,
- następnie dzielono go na okna i uśredniano wartości skal tworząc szesnasto lub osiemnasto elementowe wektory (w zależności od procedury),
- wektory o wartościach poniżej ‘odcięcia szumów’ oznaczano jako ciszę, wyznaczając na tej bazie fragmenty fonacji,
- dzięki dodatkowym kryteriom charakterystyk czasowych nie płynności (wyznaczonych na podstawie analizy materiału dźwiękowego), zredukowano fonacje nie spełniające czasowych warunków brzegowych.

W przypadku detekcji przedłużeń:

- zmniejszano wymiar wektorów za pomocą sieci Kohonena z nowatorskim, zmodyfikowanym algorytmem trenującym (‘redukcja neuronów sieci Kohonena’),
- wyszukiwano fragmentów, w których indeks wygrywającego neuronów utrzymywał się zadany parametrem czas trwania i oznaczano je, jako przedłużenia.

W przypadku detekcji powtórzeń głosek:

- zmniejszano wymiar wektorów za pomocą sieci Kohonena z nowatorskim, zmodyfikowanym algorytmem trenującym (‘zerowanie pierwszego neuronu’),

- klasyfikowano wektory do zbiorów powtórzenie głoski/płynność przy użyciu wielowarstwowego perceptronu.

W przypadku detekcji powtórzeń sylab:

- obliczano wartości korelacji sąsiadujących fonacji,
- na podstawie wartości granicznej korelacji (lub linii granicznej), klasyfikowano fonacje do zbiorów powtórzenie sylaby/płynność.

Mimo iż w/w procedury różnią się od siebie, dzięki temu, że bazującą na wspólnych algorytmach parametryzacji i podobnej metodologii, są do siebie zbliżone – stanowiąc niejako rodzinę procedur.

## **8.2. Podsumowanie wyników rozpoznawania**

Jako, że w mowie płynnej zbiór wektorów zawierających niepłynność jest dużo mniejszy od liczby wszystkich analizowanych wektorów, autor pracy uznał, iż pojedynczy parametr oznaczający liczbę wykrytych niepłynności byłby wysoce niewystarczający. Nie odzwierciedlały on liczby pomyłek w mowie płynnej, który przy takiej dysproporcji wektorów płynnych do niepłynnych jest równie ważny. Z tego powodu, do oceny rozpoznawania zaburzeń zastosowano [2] współczynniki czułości (wartość proporcjonalna do liczby wykrytych niepłynności) i przewidywalności (wartość odwrotnie proporcjonalna do liczby popełnionych błędów w mowie płynnej).

Do badań użyto nagrania dwudziestu osób jękających się oraz czterech osób zdrowych. Charakterystykę użytego materiału dźwiękowego przedstawia poniższa tabelka:

tab. 8.2-1 Analiza badanego materiału dźwiękowego dla poszczególnych rodzajów niepłynności

<b>rodzaj niepłynności</b>	<b>łączny czas trwania nagrań</b>	<b>liczba niepłynności</b>
przedłużenia	18 min. 32 s.	373
powtórzenia głosek	9 min. 43s.	294
powtórzenia sylab	5 min. 26 s.	106

Widać, że analizowany materiał był dość obszerny i zróżnicowany, zatem wydaje się, że opracowane metody powinny być uniwersalne, tj. powinny być tak samo skuteczne dla dowolnych wypowiedzi – oczywiście wypowiedzi w języku polskim. Trudno powiedzieć,



jakie wyniki moglibyśmy uzyskać dla innego języka – z racji pojawiania się w nim innych fonemów i innych ich sekwencji, odpowiedź nie jest oczywista.

Uzyskano następujące wyniki rozpoznawania:

tab. 8.2-2 Najlepsze wyniki rozpoznawania poszczególnych rodzajów niepełności

rodzaj niepełności	czułość	przewidywalność	konfiguracja parametrów
przedłużenia	92%	82%	<p>falka Morleta20  okno 23,2 ms, 100% przesunięcia  18 skal barkowych: 4-21  poziom odcięcia szumów: 55dB  otoczenie wycinania: 2,5s.  rozmiar sieci Kohonena: 4x4  sąsiedztwo uczenia sieci: 2,5-0,5  współczynniki uczenia: 0,2-0,1  długość uczenia: 100 epok  dystans redukcji sieci: 0,55  minimalna długość sekwencji 250ms  otoczenie wycinania 2000ms</p> <p><i>wykrywanie fonacji:</i>  falka Morleta20  okno 23,2 ms, 50% przesunięcia  18 skal barkowych: 4-21  poziom odcięcia szumów: 55dB</p>
powtórzenia głosek	86%	95%	<p>falka Morleta20  okno 23,2 ms, 100% przesunięcia  16 skal barkowych: 6-21  poziom odcięcia szumów 54dB  otoczenie wycinania 1,5s  rozmiar sieci Kohonena: 5x5  sąsiedztwo uczenia sieci: 2,5-0,5  współczynniki uczenia: 0,2-0,1  długość uczenia: 100 epok</p>

			<p>perceptron: MLP 65-83-1</p> <p>uczenie perceptronu: BP100,CG28b</p> <p>minimalna przerwa między fonacjami 50ms</p> <p><i>wykrywanie fonacji:</i></p> <p>falka Morleta20</p> <p>okno 23,2 ms, 50% przesunięcia</p> <p>16 skal barkowych: 6-21</p> <p>poziom odcięcia szumów 54dB</p>
powtórzenia sylab	81%	83%	<p>falka Morleta39,5</p> <p>okno 23,2 ms, 100% przesunięcia</p> <p>16 skal barkowych: 6-21</p> <p>poziom odcięcia szumów 60dB</p> <p>linia graniczna korelacji zbliżona do prostej</p> <p><math>y = 0,635650</math></p> <p><i>wykrywanie fonacji:</i></p> <p>falka Morleta20</p> <p>okno 23,2 ms, 50% przesunięcia</p> <p>16 skal barkowych: 6-21</p> <p>poziom odcięcia szumów 55dB</p>

Zaproponowane procedury rozpoznawania są w pełni automatyczne – fragmenty na żadnym etapie nie są manualnie (odsluchowo) segmentowane ani oznaczane. Podejście polegające na automatycznym i sekwencyjnym wycinaniu dużej liczby wektorów mowy płynnej powoduje, że są one bardzo zróżnicowane i często są bardzo podobne do wektorów mowy zaburzonej – co stanowi dużą trudność. Z tego powodu, w początkowych fazach tworzenia każdej z procedur, wyniki rozpoznawania były bardzo niskie, mimo iż wzorowano się na sprawdzonych przez innych badaczy rozwiązaniach. Z tego względu powstała potrzeba opracowania i weryfikacji nowych algorytmów, dzięki którym uzyskano wysoką skuteczność identyfikacji niepełności w mowie ciągłej.

### **8.3. Wnioski końcowe**

Realizując cel pracy, udało się stworzyć program automatycznie rozpoznający nie płynności w mowie ciągłej z jednoczesnym, precyzyjnym wyznaczeniem początku i końca każdego zaburzenia.

Parametryzowanie sygnału mowy algorytmem ciągłej transformaty falkowej jest dobrym rozwiązaniem, dzięki któremu możemy elastycznie wybierać pasma częstotliwości, w których chcemy wyznaczyć współczynniki spektrogramu. Parametryzacja taka wraz z zastosowaniem modelu percepcyjnego, polegającym na wyliczaniu spektrogramu dla skal barkowych, które odzwierciedlają charakterystykę słuchową człowieka, przyniosła zamierzone rezultaty – wyniki rozpoznawania nie płynności są wysokie.

Sieci Kohonena bardzo dobrze nadają się do redukcji danych wejściowych. Uzyskiwane sekwencje wygrywających neuronów bardzo upraszczają postać danych, czyniąc je łatwiejszymi do dalszej analizy. Przy dodatkowej modyfikacji algorytmu uczenia otrzymywano bardzo dobre (tj. łatwe do sklasyfikowania) odwzorowania przedłużeń i powtórzeń głosek. Powtórzenia sylab nie są już tak dobrze odwzorowywane (przynajmniej przy testowanych przez autora konfiguracjach parametrów sieci Kohonena) – to znaczy przebiegi wygrywających neuronów sieci Kohonena dla takich samych sylab są istotnie różne. Natomiast algorytm korelacyjny zastosowany bezpośrednio dla parametrów CWT (bez redukcji sieciami Kohonena) – generuje bardzo wysokie współczynniki korelacji.

Program WaveBlaster ma charakter badawczy. Zawiera bardzo dużo opcji oraz wylicza różnorodne (często nadmiarowe) współczynniki, ponieważ jest to niezbędne w procesie wyszukiwania zależności między danymi, tym samym optymalizując proces wyszukiwania nie płynności. W kolejnym kroku należałoby znacznie uprościć interfejs użytkownika, który na razie jest ukierunkowany na zastosowania naukowe – dzięki czemu, mógłby zacząć być stosowany przez logopedów. Należałoby również zoptymalizować program WaveBlaster, zarówno pod kątem szybkości działania jak również pod kątem potrzebnej pamięci RAM tak, aby mógł zacząć być efektywnie stosowany na mniej wymagających maszynach, dzięki czemu mógłby być powszechniej stosowany.

## 9. Bibliografia

- A.N. Akansu and R.A. Haddad, *Multiresolution signal decomposition.*: Academic  
1] Press, 2001.
- S. Barro and R. Marin, *Fuzzy Logic in Medicine.* New York: Physica-Verlag  
2] Heidenberg, 2002.
- J. T. Białasiewicz, *Falki i Aproksymacje.* Warszawa: Wydawnictwo Naukowo-  
3] Techniczne, 2000.
- J.P. Campbell Jr., "Speaker recognition: a tutorial," *Proceedings of the IEEE*, vol. 85,  
4] no. 9, pp. 1437 - 1462, 1997.
- K. Chanwoo, S. Kwang-deok, and S. Wonyong, "A robust formant Extraction  
5] algorithm combining spectral peak picking and root polishing," *EURASIP Journal on  
Applied Signal Processing*, pp. 33-33, 2006.
- I. Codello and W. Kuniszyk-Józkowiak, "„Wave Blaster” – a comprehensive tool for  
6] speech analysis and its application for vowel recognition using wavelet continuous  
transform with bark scales," *56 Otwarte Seminarium z Akustyki OSA*, pp. 63-68, 2009.
- I. Codello and W. Kuniszyk-Józkowiak, "Digital signals analysis with LPC method,"  
7] *Annales UMCS Informatica AI 5*, pp. 315-321, 2006.
- I. Codello and W. Kuniszyk-Józkowiak, "Formant paths tracking using Linear  
8] Prediction based methods," *Annales UMCS Informatica AI X(2)*, pp. 7-12, 2010.
- I. Codello and W. Kuniszyk-Józkowiak, "Wavelet analysis of speech signal," *Annales  
9] UMCS Informatica AI 6*, 2007.
- I. Codello, W. Kuniszyk-Józkowiak, T. Gryglewicz, and W. Suszyński, "Utterance  
10] intonation imaging using the cepstral analysis," *Annales UMCS Informatica AI 8(1)*, pp.  
157-163, 2008.
- I. Codello, W. Kuniszyk-Józkowiak, and A. Kobus, "Kohonen networks application  
11] in speech analysis algorithms," *Annales UMCS Informatica AI X(2)*, pp. 13-19, 2010.

I. Codello, W. Kuniszyk-Józkowiak, E. Smoła, and A. Kobus, "Automatic  
12] disordered sound repetition recognition in continuous speech using CWT and Kohonen  
network," *Annales UMCS Informatica*, 2012.

I. Codello, W. Kuniszyk-Józkowiak, E. Smoła, and A. Kobus, "Automatic  
13] disordered syllables repetition recognition in continuous speech using CWT and  
correlation," *Advances in Intelligent and Soft Computing*, 2013.

I. Codello, W. Kuniszyk-Józkowiak, E. Smoła, and A. Kobus, "Automatic  
14] prolongation recognition in disordered speech using CWT and Kohonen network," *Journal  
Of Medical Informatics & Technologies*, 2012.

I. Codello, W. Kuniszyk-Józkowiak, E. Smoła, and A. Kobus, "Disordered sound  
15] repetition recognition in continuous speech using CWT and Kohonen network," *Journal Of  
Medical Informatics & Technologies, Vol. 17*, pp. 123-130, 2011.

I. Codello, W. Kuniszyk-Józkowiak, E. Smoła, and A. Kobus, "Prolongation  
16] Recognition in Disordered Speech," *Proceedings of International Conference on Fuzzy  
Computation*, pp. 392-398, 2010.

I. Codello, W. Kuniszyk-Józkowiak, E. Smoła, and W. Suszyński, "Speaker  
17] Recognition using Continuous Wavelet Transform with Bark Scales," *Polish J. of Environ.  
Stud. Vol. 18*, pp. 78-82, 2009.

A. Czyżewski, A. Kaczmarek, and B. Kostek, "Intelligent processing of stuttered  
18] speech," *Journal of Intelligent Information Systems*, vol. 21, pp. 143-171, 2003.

A. Czyżewski, B. Kostek, and H. Skarżynski, *Technika komputerowa w audiologii,  
19] foniatrii i logopedii*. Warszawa: Exit, 2002.

M. Dzieńkowski, "Komputerowe słuchowo-wizualne diagnozowanie i terapia  
20] niepełności mowy," *praca doktorska, Instytut Biocybernetyki i Inżynierii Biomedycznej  
PAN*, 2007.

M. Dzieńkowski, W. Kuniszyk-Józkowiak, E. Smoła, and W. Suszyński, "Computer  
21] programme for speech impediment diagnosis and therapy," *Annales Informatica  
Universitatis Mariae Curie-Skłodowska*, pp. 21-29, 2003.

M. Dzieńkowski, W. Kuniszyk-Józkowiak, E. Smoła, and W. Suszyński, "Computer

22] speech echo-corrector," *Annales Informatica Universitatis Mariae Curie- Skłodowska*, pp. 315-322, 2004.

M. Dzieńkowski, W. Kuniszyk-Józkowiak, E. Smółka, and W. Suszyński, "Cyfrowa  
23] analiza plików dźwiękowych," *Lubelskie Akademickie Forum Informatyczne*, pp. 71- 78, 2002.

M. Dzieńkowski, W. Kuniszyk-Józkowiak, E. Smółka, and W. Suszyński,  
24] "Komputerowa zindywidualizowana terapia niepełnej mowy," *XIII Konferencja Naukowa Biocybernetyka i Inżynieria Biomedyczna*, vol. I, pp. 546-551, 2003.

M. Dzieńkowski, W. Kuniszyk-Józkowiak, E. Smółka, and W. Suszyński,  
25] "Komputerowe słuchowo-wizualne echo dla celów terapii niepełności mowy," *Obliczenia naukowe - wybrane problemy, PTI*, pp. 57-63, 2003.

S. Garfield, M. Elshaw, and S. Wermter, "Self-organizing networks for classification  
26] learning from normal and aphasic speech," *In The 23rd Conference of the Cognitive Science Society*, 2001.

P. Getreuer, *Filter Coefficients to popular wavelets.*, 2006. [Online].  
27] <http://read.pudn.com/downloads129/sourcecode/math/551883/Filter%20Coefficients%20to%20Popular%20Wavelets.PDF>

B. Gold and N. Morgan, *Speech and audio signal processing*. New York: John Wiley  
28] & Sons Inc., 2000.

P. Goupillaud, A. Grossmann, and J. Morlet, "Cycle-octave and related transforms in  
29] seismic signal analysis," *Geoexploration 23*, pp. 85-102, 1984-1985.

A. Horzyk and R. Tadeusiewicz, "Self-optimizing neural networks, Advances in  
30] neural networks," *Lecture notes in computer science*, pp. 150-155, 2004.

P. Howell and S. J. Sackin, "Automatic recognition of repetitions and prolongations  
31] in stuttered speech," *Stuttering: Proceedings of the First World Congress on Fluency Disorders*, pp. 372-374, 1995.

P. Howell, S. J. Sackin, and K. Glenn, "Development of two stage procedure for the  
32] automatic recognition of dysfluencies in the speech of children who stutter: II ANN recognition of repetitions and prolongations with supplied word segment markers," *J.*

*Speech Hear. Res.*, vol. 40, pp. 1085-1096, 1997a.

P. Howell, S. J. Sackin, and K. Glenn, "Development of two-stage procedure for the  
33] automatic recognition of dysfluencies in the speech of children who stutter: I Psychometric  
procedures appropriate for selection of training material for lexical dysfluency classifiers,"  
*J. Speech Hear. Res.*, vol. 40, pp. 1073-1084, 1997b.

X. Huang and A. Acero, *Spoken Language Processing: A Guide to Theory, Algorithm  
34] and System Development.*: Prentice-Hall Inc., 2001.

A. Izworski and W. Wszolek, "Wykorzystanie metod sztucznej inteligencji w  
35] diagnostyce i przetwarzaniu patologicznych sygnałów akustycznych," *Speech and  
Language Technology 3*, pp. 299-319, 1999.

A. Izworski, W. Wszolek, R. Tadeusiewicz, and T. Wszolek, "Understanding of  
36] deformed speech signals using vocal tract simulation," *Advances of Medicine and Health  
Care through Technology - the Challenge to Biomedical Engineering in Europe*, pp. 532-  
533, 2002.

A. Jansen and A. la Cour-Harbo, *Ripples in Mathematisc – The Discrete Wavelet  
37] Transform*. Berlin Heidelberg: Springer-Verlag, 2011.

A. Kobus, W. Kuniszyk-Józkowiak, E. Smółka, and I. Codello, "Speech Nonfluency  
38] Detection and Classification Based on Linear Prediction Coefficients and Neural  
Networks," *Journal of Medical Informatics & Technologies*, Vol. 15, pp. 135-144, 2010.

A. Kobus, W. Kuniszyk-Józkowiak, E. Smółka, I. Codello, and W. Suszyński, "The  
39] Prolongation-Type Speech Non-fluency Detection Based on the Linear Prediction  
Coefficients and the Neural Networks," *Proceedings of the 8th International Conference on  
Computer Recognition Systems CORES*, vol. 226, pp. 887-897, 2013.

A. Kobus, W. Kuniszyk-Józkowiak, E. Smółka, W. Suszyński, and I. Codello, "A  
40] new elliptical model of the vocal tract," *Journal Of Medical Informatics & Technologies*,  
Vol. 17, pp. 131-139, 2011.

T. Kohonen, "Self-Organizing Maps," pp. 2173-2179, 2001.  
41]

A. Komae and A. Sepehri, "Linear Prediction and Synthesis of Speech Signals,"

42] Department of Electrical and Computer Engineering, University of Maryland. [Online].  
<http://www.enee.umd.edu/~afshin/adsp2/proj2.pdf>

W. Kuniszyk-Józkowiak, "A comparison of speech envelopes of stutterers and  
43] nonstutterers," *J. Acoust. Soc. Am.*, pp. 1105-1110, 1996.

W. Kuniszyk-Józkowiak, *Przetwarzanie sygnałów biomedycznych*. Lublin:  
44] Uniwersytet Marii Curie-Skłodowskiej w Lublinie, 2011.

W. Kuniszyk-Józkowiak, J. Jaszczuk, T. Sacewicz, and I. Codello, "Time–frequency  
45] Analysis of the EMG Digital Signals," *Annales UMCS Informatica AI XII*, pp. 19-25, 2012.

W. Kuniszyk-Józkowiak and M. Sztubecki, "Analiza nie płynności mówienia osób  
46] jękających się. Stan aktualny i perspektywy," *Diagnoza i terapia osób jękających się,*  
*Warsztaty Logopedyczne*, pp. 16-25, 1997.

R.D. Lyons, *Wprowadzenie do cyfrowego przetwarzania sygnałów*. Warszawa,  
47] Wydawnictwa Komunikacji i Łączności, 2003.

E. Nöth et al., "Automatic stuttering recognition using Hidden Markov Models,"  
48] *Proc. Int. Conf. on Spoken Language Processing*, pp. 65-68, 2000.

Ooi Chia Ai, M. Hariharan, S. Yaacob, and LimSinChee, "Expert Systems with  
49] Applications," vol. 39, pp. 2157-2165, 2012.

R. Polikar. The wavelet tutorial. [Online].  
50] <http://users.rowan.edu/~polikar/WAVELETS/WTpart1.html>

L.R. Rabiner and R.W. Schafer, *Digital Processing of Speech Signals*. New Jersey:  
51] Prentice-Hall, Inc., 1978.

K. M. Ravikumar and S. Ganesan, "Comparison of Multidimensional MFCC Feature  
52] Vectors for Objective Assessment of Stuttered Disfluencies ," *Int. J. Advanced Networking*  
*and Applications* , vol. 2, no. 5, pp. 854-860, 2011.

K. M. Ravikumar, R. Rajagopal, and C. Nagaraj, "An Approach for Objective  
53] Assessment of Stuttered Speech Using MFCC Features," *DSP Journal*, vol. 9, no. 1, pp. 19-  
24, 2009.

K. M. Ravikumar, B. Reddy, R. Rajagopal, and C. Nagaraj, "Automatic Detection of



54] Syllable Repetition in Read Speech for Objective Assessment of Stuttered Disfluencies," *World Academy of Science, Engineering and Technology*, pp. 270-273, 2008.

I. Simonovski and M. Boltezar, "The norms and variances of the Gabor, Morlet and  
55] general harmonic wavelet functions," *Journal of Sound and Vibration*, vol. 264, no. 3, pp. 545-557, 2003.

S. W. Smith, *Digital signal processing*. San Diego, California, 1994.

56]

J. Smith and J. Abel, *Bark and ERB Bilinear Transforms.*: IEEE Transactions on  
57] Speech and Audio Processing, 1999.

E. Smółka, W. Kuniszyk-Józkowiak, M. Dzieńkowski, W. Suszyński, and M.  
58] Swietlicki, "Rozpoznawanie samogłosek w izolacji i mowie ciągłej z wykorzystaniem perceptronu wielowarstwowego," *Structures Waves - Human Health*, pp. 143-148, 2005.

E. Smółka, W. Kuniszyk-Józkowiak, and W. Suszyński, "Odwzorowanie płynnych i  
59] niepłynnych słów w sieci Kohonena," *XLIX Otwarte Seminarium z Akustyki*, 2002.

E. Smółka, W. Kuniszyk-Józkowiak, and W. Suszyński, "Reflection of fluent and  
60] nonfluent words in Kohonen network," *XLIX Open Seminar on Acoustics*, pp. 371-376, 2002.

E. Smółka, W. Kuniszyk-Józkowiak, W. Suszyński, and M. Dzieńkowski, "Speech  
61] syllabic structure extraction with application of Kohonen network," *Annales Informatica UMCS*, pp. 125-131, 2003.

E. Smółka, W. Kuniszyk-Józkowiak, W. Suszyński, M. Dzieńkowski, and I.  
62] Szczurowska, "Speech nonfluency recognition in two stages of Kohonen networks," *Structures - Waves - Human Health*, vol. XIII, no. 2, pp. 139-142, 2004.

E. Smółka, W. Kuniszyk-Józkowiak, E. Suszyński, and M. Wiśniewski, "Vowel  
63] recognition in continuous speech with application of MLP neural network," *Annales UMCS, Sectio AI Informatica vol. V*, pp. 139-144, 2006.

StatSoft, Inc. (1997). STATISTICA for Windows [Computer program manual].  
64] Tulsa, OK: StatSoft, Inc., 2300 East 14th Street, Tulsa, OK 74104, phone: (918) 749-1119, fax: (918) 749-2217, email: info@statsoftinc.com. [Online]. <http://www.statsoft.com>

W. Suszyński, *Automatyczne rozpoznawanie niepełności mowy*. Lublin-Gliwice: 66] Praca doktorska, 2005.

W. Suszyński, "Automatyczne rozpoznawanie niepełności mowy," *50 Otwarte 65] Seminarium z Akustyki*, pp. 386-389, 2003.

W. Suszyński and M. Dzieńkowski, "Detekcja niepełności mowy przy 67] wykorzystaniu funkcji korelacji," *51 Otwarte Seminarium z Akustyki*, pp. 386-389, 2004.

W. Suszyński, W. Kuniszyk-Józkowiak, E. Smółka, and M. Dzieńkowski, 68] "Automatic recognition of nasals prolongations in the speech of persons who stutter," *Structures - Waves - Human Health*, pp. 175-184, 2003.

W. Suszyński, W. Kuniszyk-Józkowiak, E. Smółka, and M. Dzieńkowski, 69] "Automatic recognition of non-fluent stops," *Annales UMCS Informatica*, pp. 183-189, 2004.

W. Suszyński, W. Kuniszyk-Józkowiak, E. Smółka, and M. Dzieńkowski, "Speech 70] disfluency detection with the correlative method," *Annales UMCS Informatica*, vol. AI 3, pp. 131-138, 2005.

W. Suszyński, W. Kuniszyk-Józkowiak, E. Smółka, and M. Wiśniewski, 71] "Automatyczna detekcja wtrąceń," *Varia Informatica, Algorytmy i programy, Polskie Towarzystwo Informatyczne, Instytut Informatyki Politechniki Lubelskiej*, pp. 105-113, 2006.

I. Swietlicka, W. Kuniszyk-Józkowiak, and E. Smółka, "Artificial neural networks in 72] the disabled speech analysis," *Computer Recognition Systems (Advances in Soft Computing)*, Verlag Berlin Heidelberg, Springer, pp. 347-354, 2009.

I. Swietlicka, W. Kuniszyk-Józkowiak, and E. Smółka, "Detection of syllable 73] repetition using two-stage artificial neural networks," *Polish Journal of Environmental Studies*, vol. 17, pp. 462- 466, 2008.

I. Szczurowska, W. Kuniszyk-Józkowiak, and E. Smółka, "Application of Artificial 74] Neural Networks In Speech Nonfluency Recognition," *Polish Journal of Environmental Studies*, 2007 16(4A), pp. 335-338, 2007.

I. Szczurowska, W. Kuniszyk-Józkowiak, and E. Smółka, "Speech nonfluency

75] detection using Kohonen networks," *Neural Computing and Application*, vol. 18, no. 7, pp. 677-687, 2009.

I. Szczurowska, W. Kuniszyk-Józkowiak, and E. Smółka, "The application of  
76] Kohonen and Multilayer Perceptron network in the speech nonfluency analysis," *Archives of Acoustics 2006. 31 (4 (Supplement))*, pp. 205-210, 2006.

I. Szczurowska, E. Smółka, W. Kuniszyk-Józkowiak, W. Suszyński, and M.  
77] Dzieńkowski, "The application of neural networks in the speech nonfluency analysis," *Structures Waves - Human Health*, vol. XIV, no. 1, pp. 173-176, 2005.

R. Tadeusiewicz, *Elementarne wprowadzenie do sieci neuronowych z przykładowymi*  
78] *programami*. Warszawa: Akademicka Oficyna Wydawnicza, 1998.

R. Tadeusiewicz, *Sieci neuronowe*. Warszawa: EXIT, 1993.  
79]

R. Tadeusiewicz, *Sygnal mowy*. Warszawa: Wydawnictwa Komunikacji i Łączności,  
80] 1988.

R. Tadeusiewicz, *Wstęp do sieci neuronowych*. Warszawa: Akademicka Oficyna  
81] Wydawnicza Exit, 2000.

R. Tadeusiewicz, "Zastosowanie sieci neuronowych do rozpoznawania mowy,"  
82] *Analiza, synteza i rozpoznawanie sygnału mowy dla celów automatyki, informatyki, lingwistyki i medycyny*, pp. 137-150, 1994.

R. Tadeusiewicz, W. Wszolek, and A. Izworski, "Sieci neuronowe jako narzędzie do  
83] symulacji przetwarzania informacji akustycznej systemu słuchowego," *X Krajowa Konferencja Naukowa Biocybernetyka i Inżynieria Biomedyczna*, pp. 801-807, 1997.

The MathWorks, "Matlab 7 Help – Wavelets: A New Tool for Signal Analysis".  
84]

H. Traunmüller, "Analytical expressions for the tonotopic sensory scale," *J. Acoust. Soc. Am.* 88, pp. 97-100, 1990.  
85]

H. Wakita, "Direct Estimation of the Vocal Tract Shape by Inverse Filtering of  
86] Acoustic Speech Waveforms," *IEE Transactions on Audio and Electroacoustics*, vol. AU-

21, no. 5, 1973.

M. Wiśniewski and W. Kuniszyk-Józkowiak, "Automatic detection and classification  
87] of phoneme repetitions using HTK toolkit," *Journal of Medical Informatics & Technologies*, vol. 17, pp. 141-147, 2011.

M. Wiśniewski, W. Kuniszyk-Józkowiak, E. Smółka, and W. Suszyński, "Automatic  
88] detection of disorders in a continuous speech with the Hidden Markov Models approach," *Advances in Soft Computing 45, Computer Recognition Systems 2, Springer-Verlag, Berlin Heidelberg*, 2007.

M. Wiśniewski, W. Kuniszyk-Józkowiak, E. Smółka, and W. Suszyński, "Automatic  
89] detection of prolonged fricative phonemes with the Hidden Markov Models approach," *Journal of Medical Informatics and Technologies vol. 11/2007, Computer System Dept. University of Silesia*, 2007.

M. Wiśniewski, W. Kuniszyk-Józkowiak, E. Smółka, and W. Suszyński, "Improved  
90] approach to automatic detection of speech disorders based on the Hidden Markov Models approach," *Journal Of Medical Informatics & Technologies*, vol. 15, pp. 145-152, 2010.

M. Wiśniewski, W. Kuniszyk-Józkowiak, E. Smółka, and W. Suszyński, "Vision  
91] echo," *Annales UMCS, Sectio AI Informatica*, vol. III, pp. 139-144, 2005.

M. Wiśniewski, W. Kuniszyk-Józkowiak, E. Smółka, and W. Suszyński, "Automatic  
92] detection of speech disorders with the use of Hidden Markov Models," *Annales UMCS, Sectio AI Informatica vol. VI-VII*, 2007.

T.P. Zieliński, *Od teorii do cyfrowego przetwarzania sygnałów.:* Wydział EAIiE  
93] AGH, Kraków, 2002.

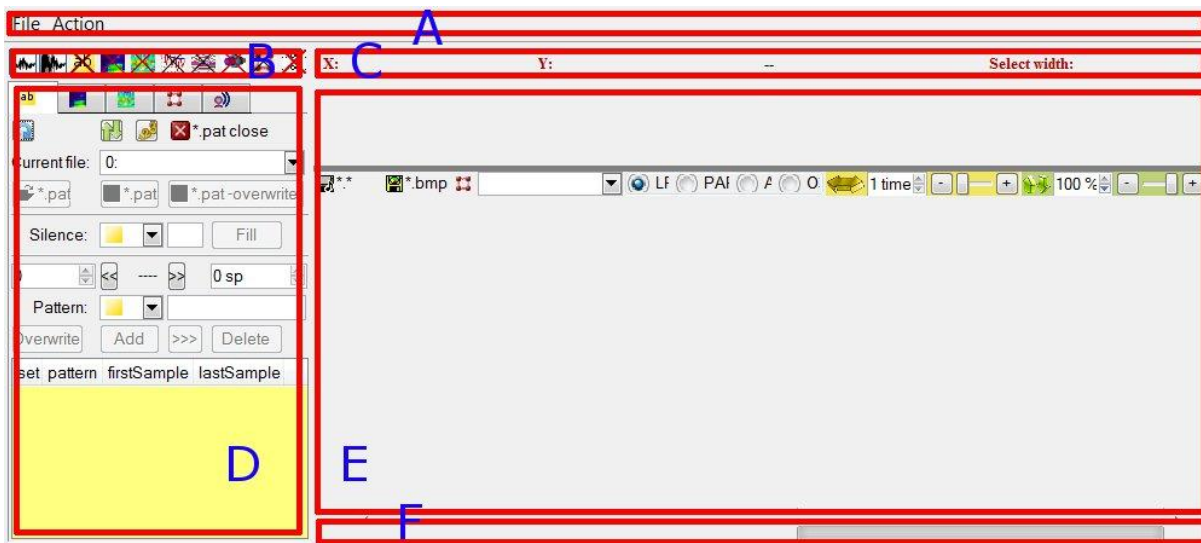
## 10. Dodatek A

Poniżej przedstawiono opis programu WaveBlaster. Nie jest to pełna specyfikacja – ma ona raczej charakter poglądowy, ułatwiający nawigowanie po nim i obrazująca jego możliwości.

### 10.1. *Ogólny opis programu WaveBlaster*

Widok programu można podzielić na 6 obszarów (patrz rys. 10.1:1):

- A – obszar menu
- B – obszar przycisków widoczności paneli
- C – obszar informacji pozycji kursora myszy
- D – obszar opcji paneli
- E – obszar paneli
- F – obszar statusu i wskaźnika postępu



rys. 10.1:1 Główny widok programu WaveBlaster



- analizy falkowej
- widm analiz Fouriera, falkowej i liniowej predykcji
- analizy liniowej predykcji
- wizualizacji traktu głosowego
- sieci Kohonena
- komunikatów

Panele te będą opisywane w kolejnych rozdziałach.

### ***10.1.3.C – obszar informacji pozycji kursora myszy***

X: 441pix, 2027234sp, 91938.0ms Y: 17pix, 13.8818 value 1641094(74426.0ms)--2142156(97149.9ms) Selection width: 501062sp(22723.9ms)

rys. 10.1:4 Wyświetlane informacje związane z pozycją kursora myszy oraz związane z zaznaczeniem dowolnego wykresu

Najeżdżając myszką nad dowolny wykres, możemy odczytać wartości osi X i Y, zarówno w pikselach jak i jednostkach osi tego wykresu, związane z tą pozycją na danym wykresie. Tak samo rzecz się ma z tworzeniem zaznaczenia (poprzez przeciąganie prawym przyciskiem myszy nad dowolnym wykresem).

### ***10.1.4.D – obszar opcji paneli***



rys. 10.1:5 Widok zakładek grupujących opcje konfiguracji algorytmów programu WaveBlaster

Większość paneli przedstawia wyniki algorytmów parametryzowanych dużą liczbą zmiennych. Opcje te zostały zgrupowane w tym właśnie obszarze na zakładkach.

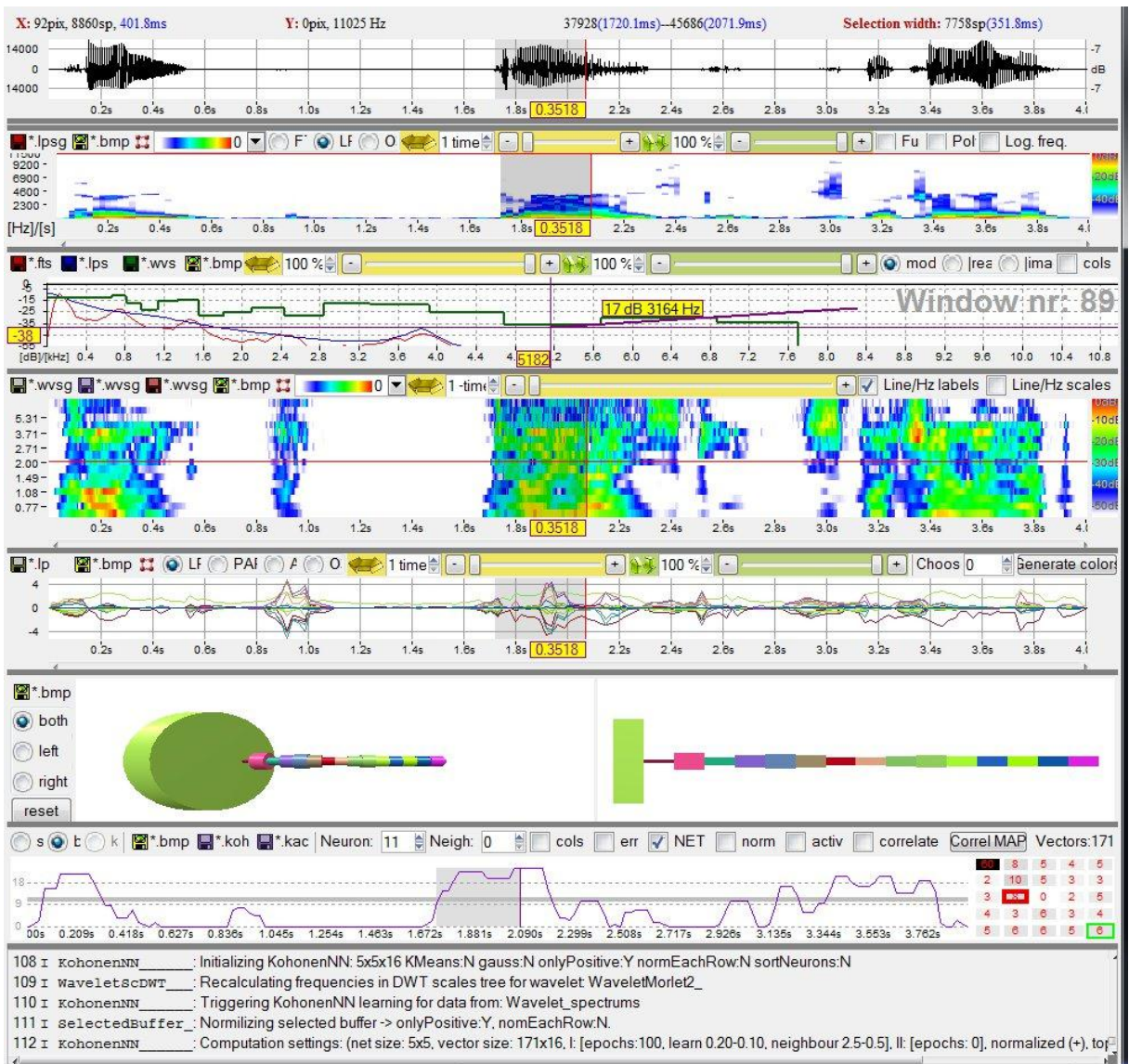
Dzięki takim samym ikonom zakładek i paneli, łatwiej zorientować się, które opcje są używane przez które panele. Tak więc mamy kolejno zakładki opcji:

- wzorców
- analizy Fouriera i liniowej predykcji

- analizy falkowej
- sieci Kohonena
- automatycznego rozpoznawanie niepłynności

Szczegółowy opis opcji znajduje się w rozdziałach z opisem paneli.

### 10.1.5.E – obszar paneli



rys. 10.1:6 Przykładowy widok obszaru z panelami programu WaveBlaster

Każdy panel można pokazać/ukryć, co bardzo ułatwia pracę – w jednym momencie potrzebujemy zazwyczaj tylko 2-3 panele. Efekt pracy ze wszystkimi panelami może być mało czytelny (patrz rys. 10.1:6).

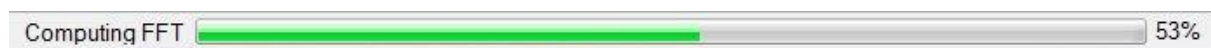


Jak napisano w poprzednim rozdziale, opcje algorytmów są umieszczone w zakładkach po lewej stronie. Ponadto większość paneli posiada jeszcze opcje dotyczące sposobu wyświetlania wyników (opcje zapisu parametrów, skalowania lub edycji obrazu), które są umieszczane już na poszczególnych panelach.

Panele najczęściej posiadają jeszcze obsługę klawiatury i myszy specyficzną dla każdego panelu, czyli obsługują w różnoraki sposób przyciski myszy lub skróty klawiaturowe.

Ponieważ WaveBlaster jest ukierunkowany do obsługi tylko jednego pliku w danej chwili, wszystkie analizy mogą dotyczyć tego samego pliku (wczytanie nowego pliku usuwa dotychczasowe analizy). Dzięki temu założeniu, oś czasu wszystkich paneli jest sprzężona – czyli nawigacja w dowolnym panelu powoduje ustawienie się innych analiz w samy miejscu osi czasu.

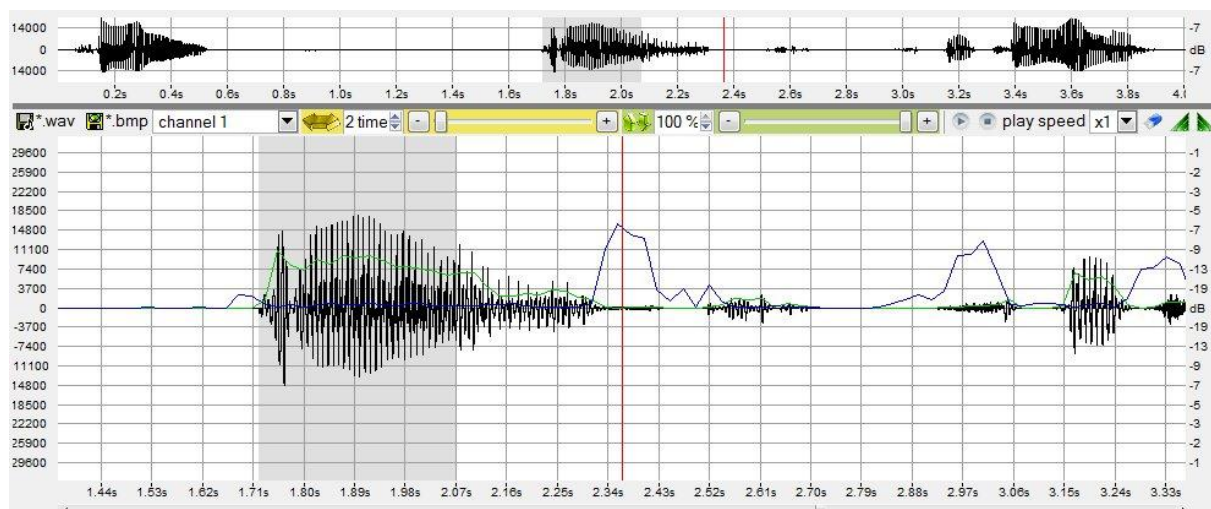
### 10.1.6.F – obszar statusu i wskaźnika postępu



rys. 10.1:7 Pasek statusu i postępu programu WaveBlaster

Status każdego wykonywanego algorytmu oraz jego procentowy stan postępu jest przedstawiany na samym dole okna programu. Niektóre algorytmy są złożone i mogą trwać dość długo (szczególnie dla większych plików) – pasek ten umożliwia śledzenie postępów procesu przetwarzania.

## 10.2. Panel oscylogramu

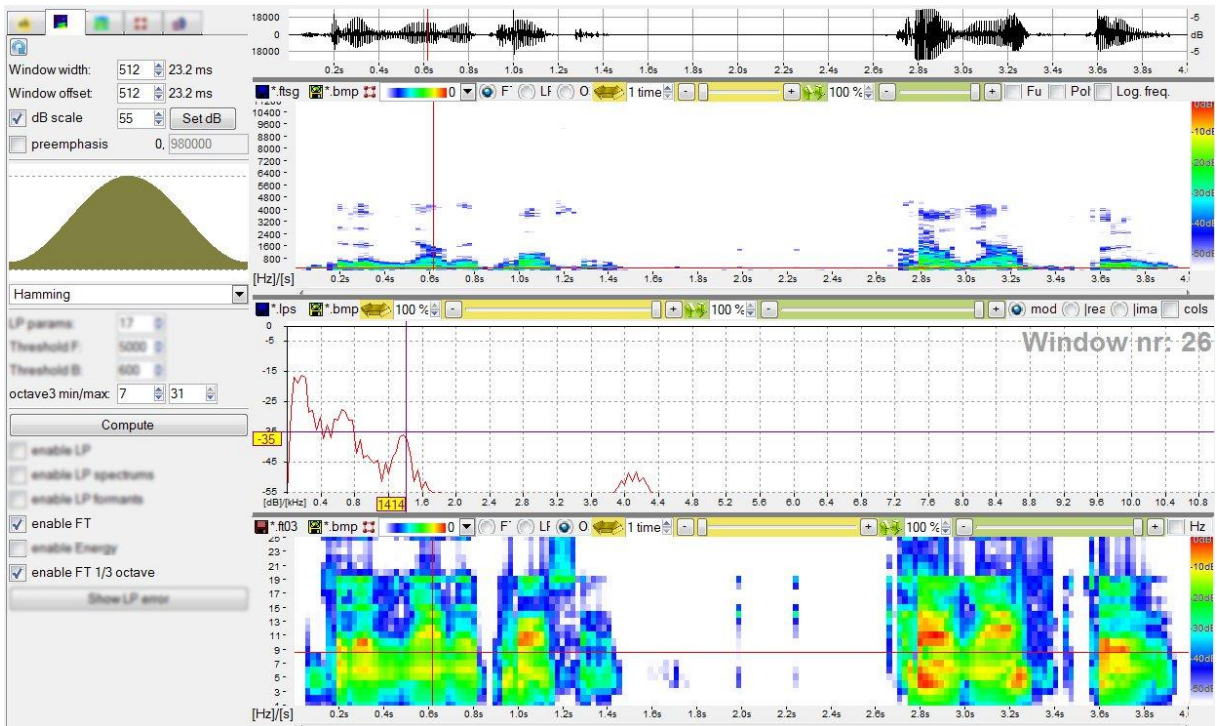


rys. 10.2:1 Panele oscylogramów programu WaveBlaster

Mamy do dyspozycji dwa panele prezentujące dane wejściowe:

- Panel większy - jego zastosowaniem jest dokładna analiza danych wejściowych,
- Panel mniejszy – prezentuje zawsze cały plik i ma on ułatwiać nawigację po nim. Dzięki małym rozmiarom, pozostałe panele mogą być większe i tym samym czytelniejsze.

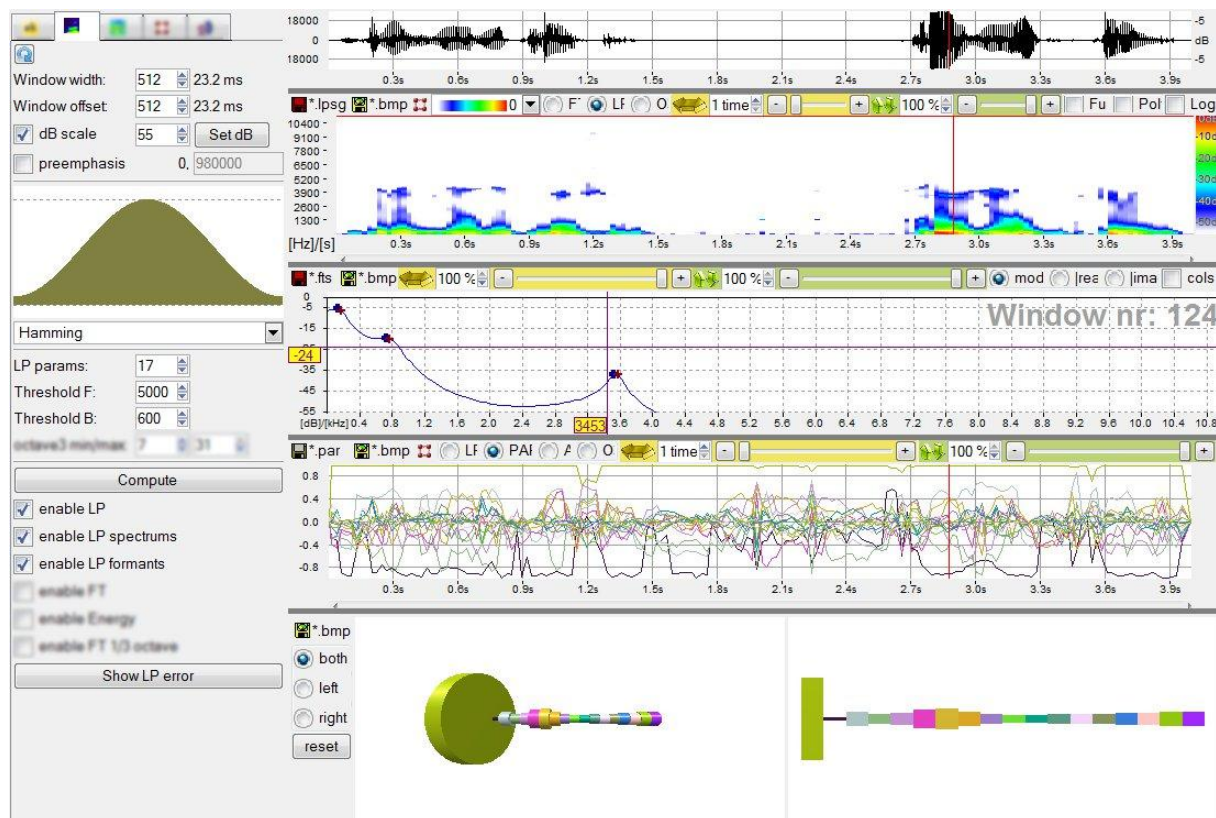
### 10.3. Panele analizy Fouriera



rys. 10.3:1 Panele prezentujące wyniki analizy Fouriera programu WaveBlaster

Analizę Fouriera można skonfigurować standardowymi parametrami: szerokość i przesunięcie okna, funkcja okna, poziom odcięcia szumów. Oprócz użycia FFT/DFT (jeżeli szerokość okna jest potęgą liczby 2, wykonuje się FFT) mamy też opcję obliczania filtrów terejowych. Ustawiając się w dowolnym miejscu oscylogramu, możemy dokładnie obejrzyć jego przekrój (czyli widmo) – pokaże się ono na „panelu widm”.

## 10.4. Panele analizy liniowej predykcji

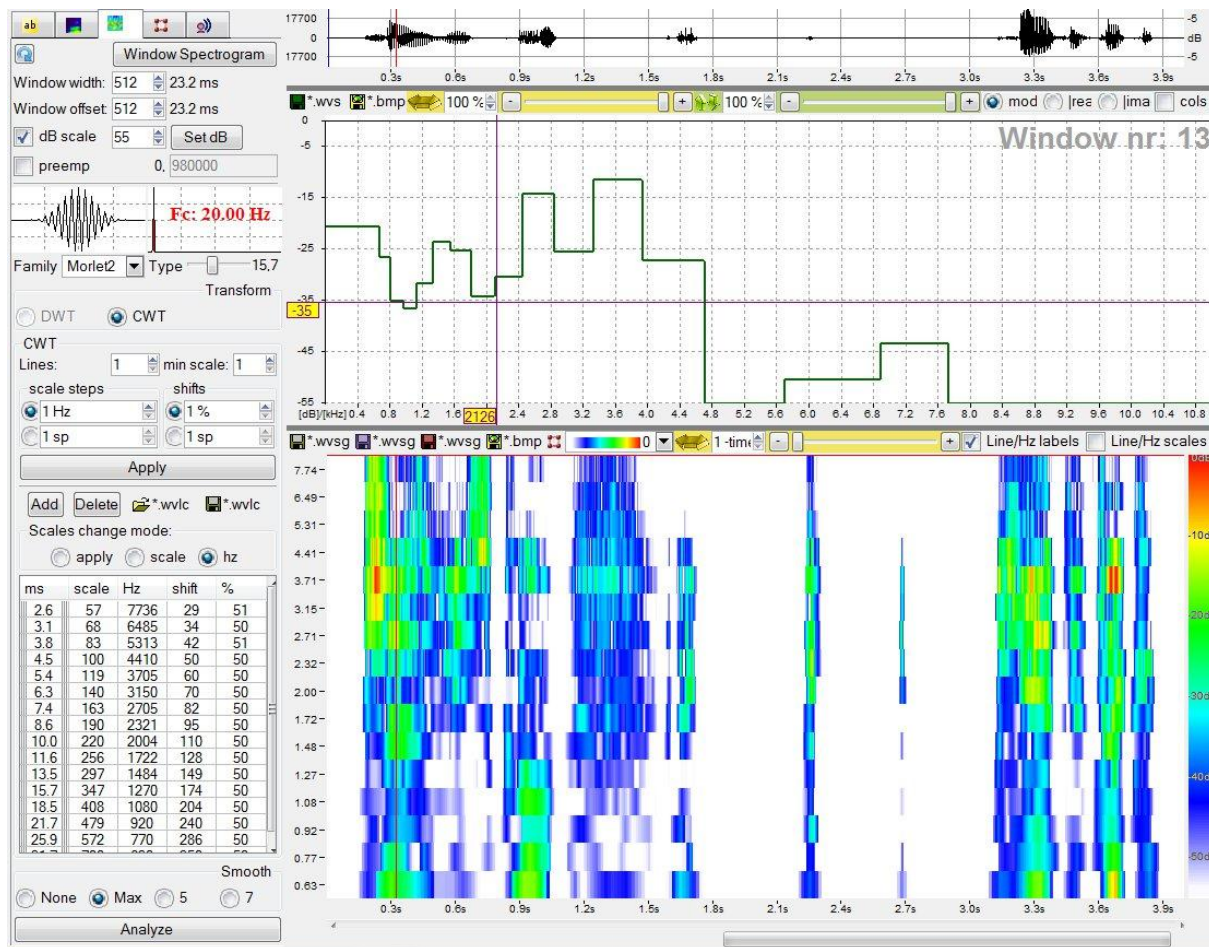


rys. 10.4:1 Panele prezentujące wyniki analizy liniowej predykcji programu WaveBlaster

Analizę liniowej predykcji można skonfigurować parametrami: szerokość i przesunięcie okna, funkcja okna, rząd predykcji oraz poziom odcięcia szumów. Oprócz prezentacji spektrogramu lub jego poszczególnych przekroi (widm), możemy również zobaczyć wykresy współczynników liniowej predykcji oraz współczynników PARCOR. Dostępna jest też wizualizacja modelu traktu głosowego wyliczanego na podstawie współczynników PARCOR.

Można też skorzystać z algorytmu wyznaczania formantów: parametrami algorytmu są współczynniki F i B, natomiast wyniki są wizualizowane zarówno na spektrogramie, jaki i na widmach.

## 10.5. Panele transformaty falkowej

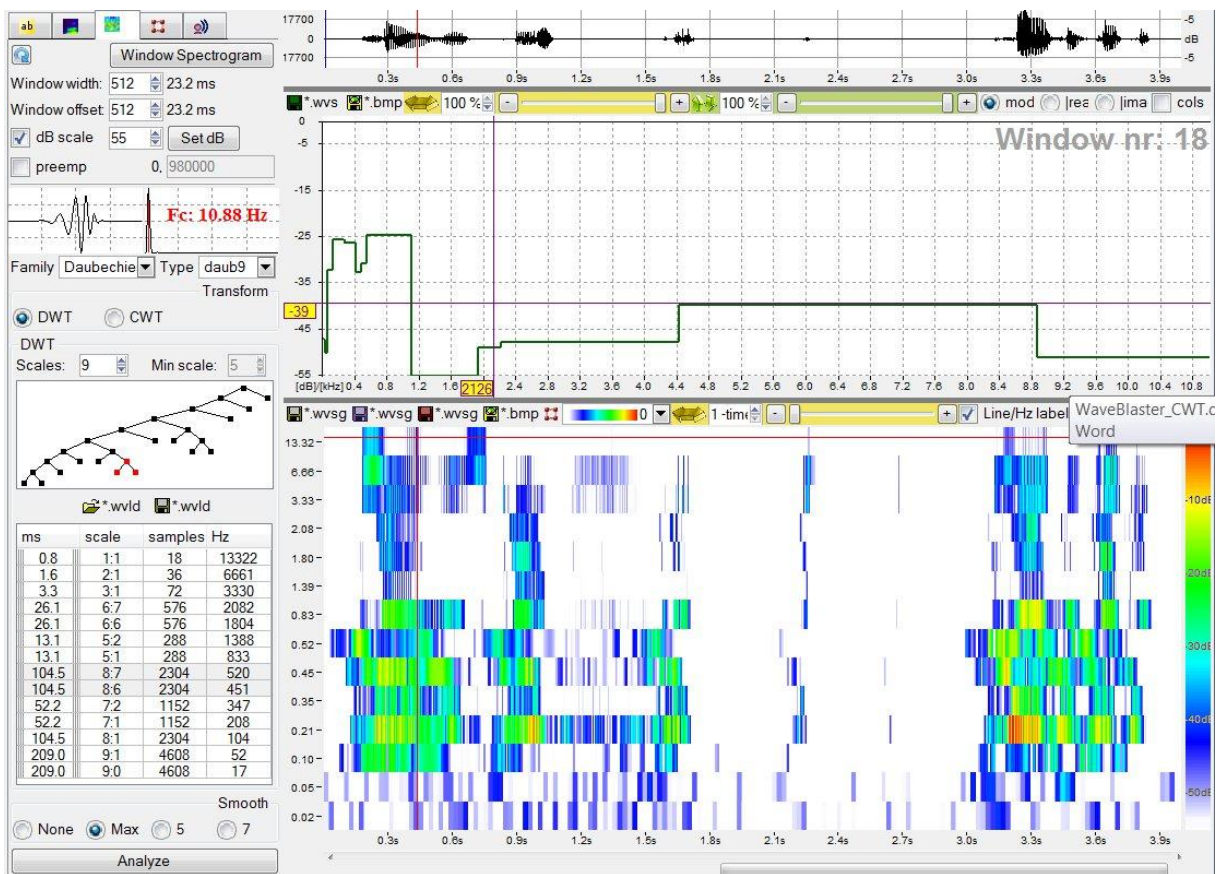


rys. 10.5:1 Panele prezentujące wyniki analizy CWT programu WaveBlaster

Najważniejszymi parametrami jest oczywiście rodzaj i typ falki. Mamy też do dyspozycji na przykład szerokość okna, potrzebną do wyznaczania wektorów (rozdział 6.2), czy wartość odcięcia szumów.

Większą część panelu opcji zajmują komponenty edycji skal, które możemy:

- wczytywać/zapisywać do pliku,
- generować na podstawie reguł takich jak ilość skal czy krok skali,
- edytować każdą komórkę tabeli ze skalami, co spowoduje automatyczne przeliczenie pozostałych wartości skal.



rys. 10.5:2 Panele prezentujące wyniki analizy DWT programu WaveBlaster

Falki dyskretne mogą być użyte w dyskretnej transformacji falkowej. Ponieważ warunki, jakie muszą spełniać wartości skal, są ściśle określone – tabelka ze skalami jest tylko do odczytu. Do tworzenia i usuwania skal służy komponent edycji drzewa binarnego (stworzony od podstaw przez autora programu).

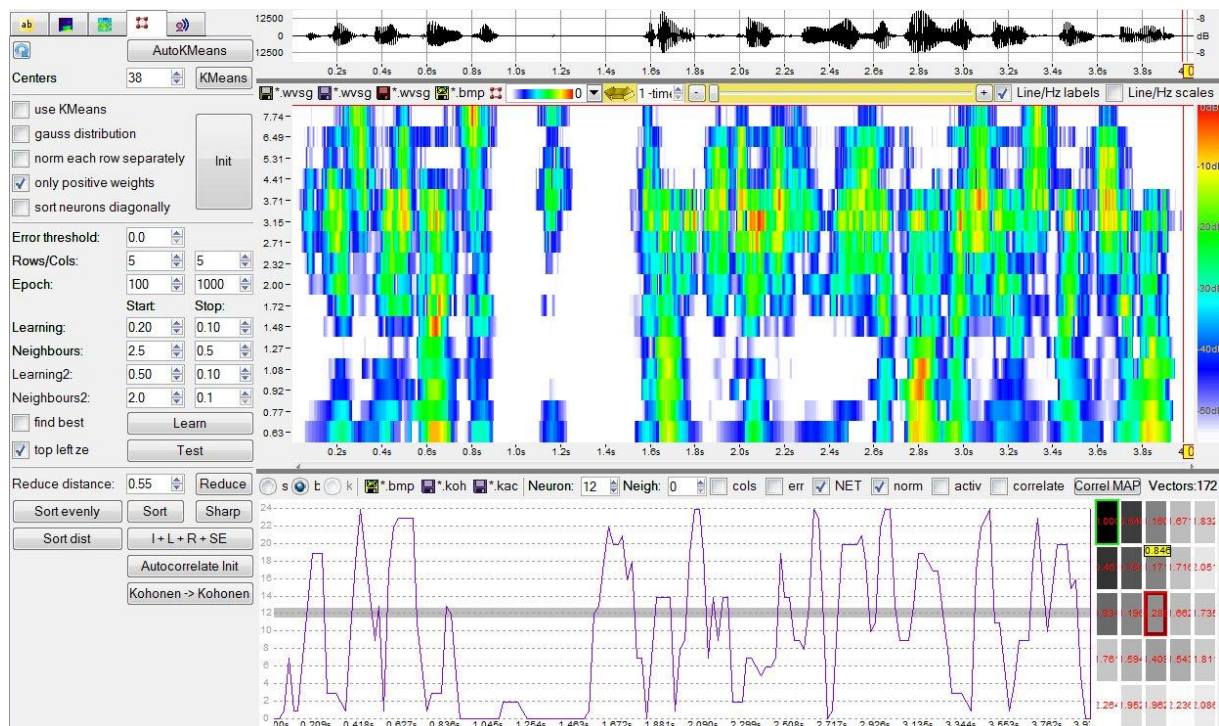
## 10.6. Panel widm



rys. 10.6:1 Panel widm Fouriera, liniowej predykcji i analizy falkowej programu WaveBlaster

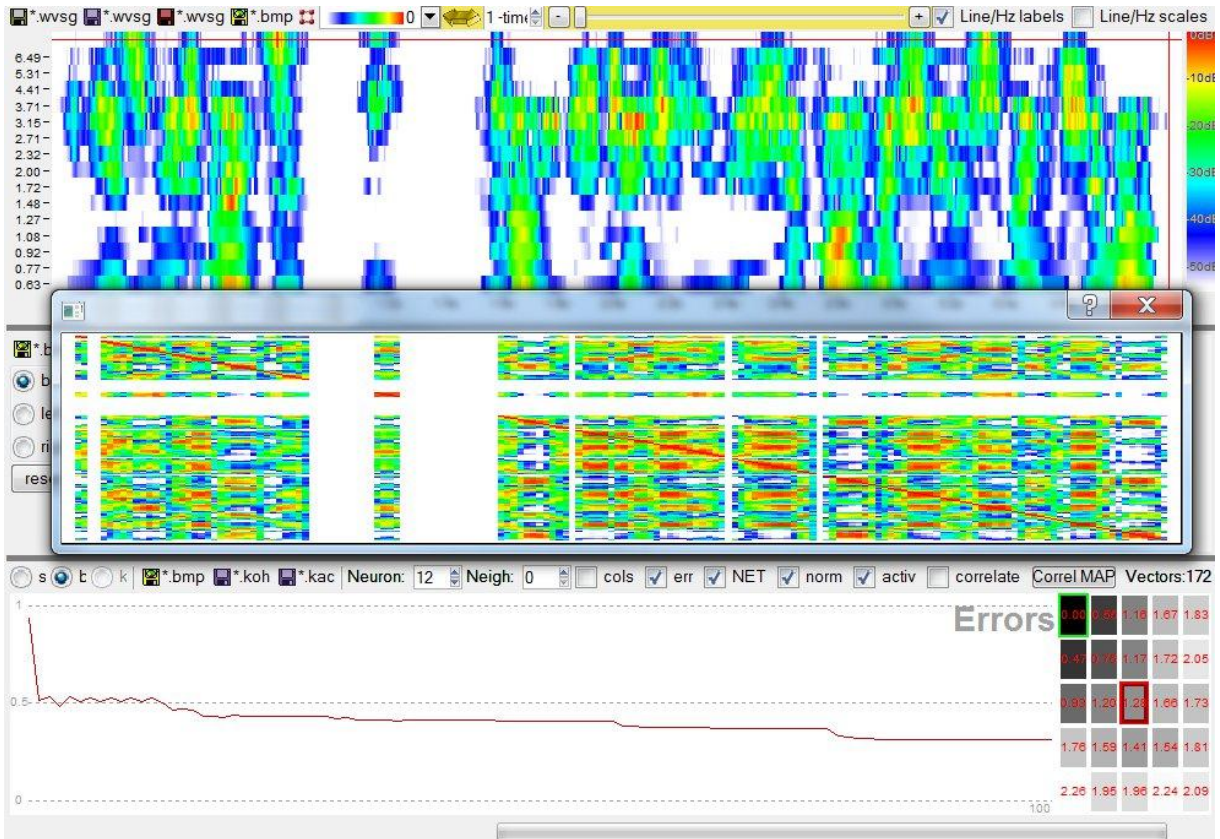
Ponieważ analiza częstotliwościowa sygnału jest podstawą wszystkich badań zawartych w tej rozprawie, na funkcjonalność tego panelu położono specjalny nacisk. Umożliwia on dokładne powiększanie, mierzenie, porównywanie i zapis widm wszystkich analiz.

## 10.7. Panel sieci Kohonena




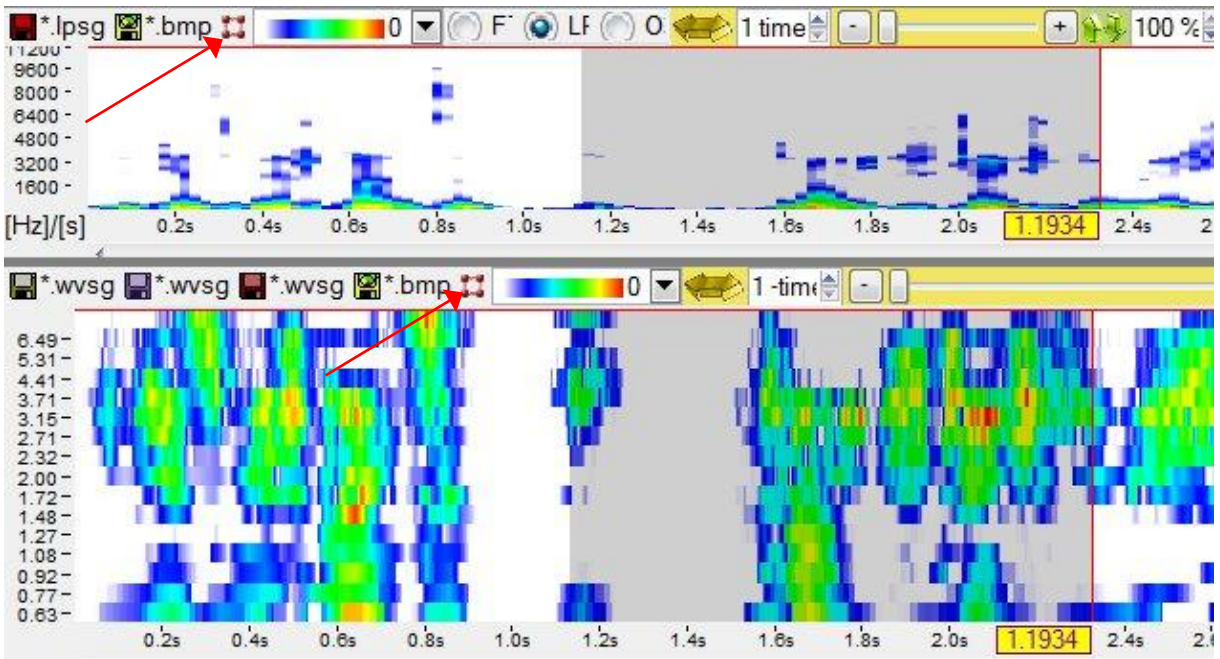
rys. 10.7:1 Panele związane z funkcjonalnością sieci Kohonena programu WaveBlaster

Funkcjonalność związana z siecią Kohonena jest również bardzo rozbudowana. Szeroki zakres sposobów inicjowania, parametrów uczenia jak i metod post-procesowania sprawiają, że panel opcji jest rozbudowany. Liczny jest też zbiór opcji na samym panelu, umożliwiając na przykład dokładne oglądanie wartości błędu aktywacji neuronów, odległości wag między nimi czy nawet wartości korelacji obrysu wygrywającego neuronu (patrz rys. 10.7:1).



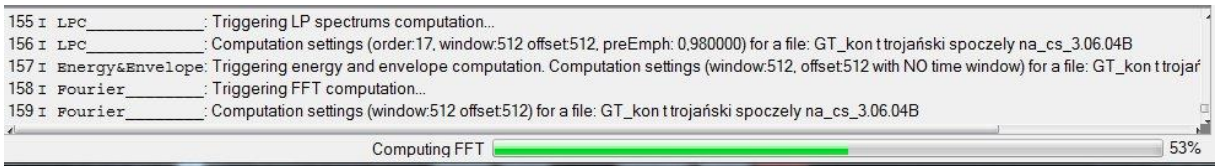
rys. 10.7:2 Panele związane z funkcjonalnością sieci Kohonena programu WaveBlaster

Do panelu Kohonena można zaimportować dane z innych komponentów – wystarczy zaznaczyć pożądany fragment i nacisnąć przycisk  znajdujący na każdym panelu.



rys. 10.7:3 Prezentacja sposobu eksportu danych do panelu sieci Kohonena programu WaveBlaster

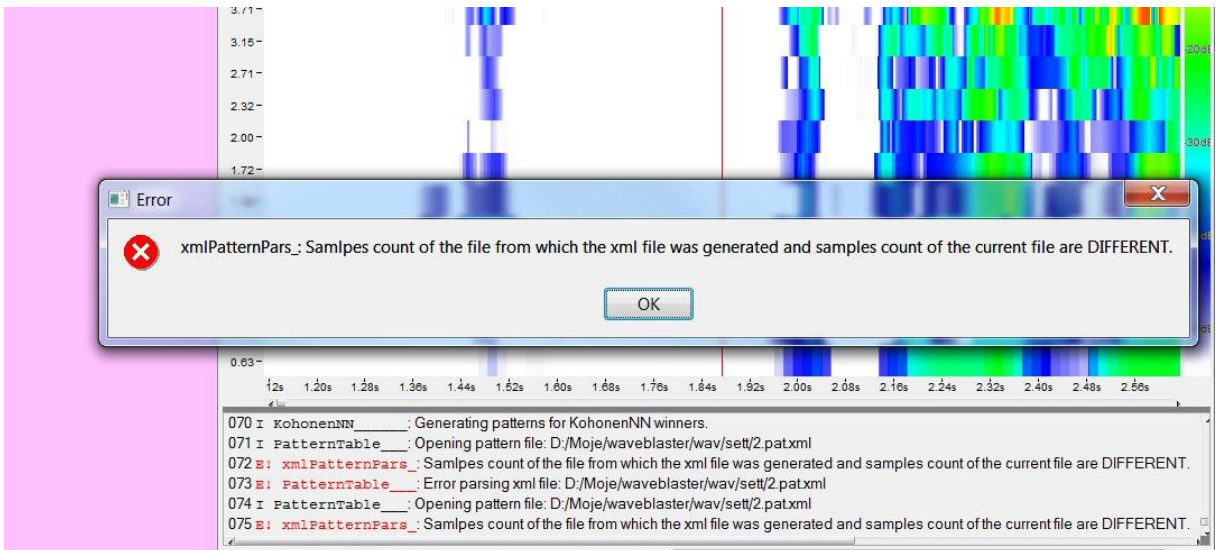
## 10.8. Panel komunikatów



rys. 10.8:1 Panel komunikatów wraz ze wskaźnikiem postępu programu WaveBlaster

Jak opisano w poprzednich rozdziałach, każdy uruchamiany program jest monitorowany – to znaczy, że na dole okna pojawia się status, czyli nazwa aktualnie uruchomionego algorytmu, oraz jego wskaźnik postępu. Przy długiej pracy i sprawdzaniu wielu zestawów parametrów wejściowych istnieje duże ryzyko pomyłki. Można wtedy otworzyć ten panel, a następnie prześledzić, jaki algorytm i z jakimi parametrami został właśnie uruchomiony. Ponadto komunikaty mają typy – mogą to być:

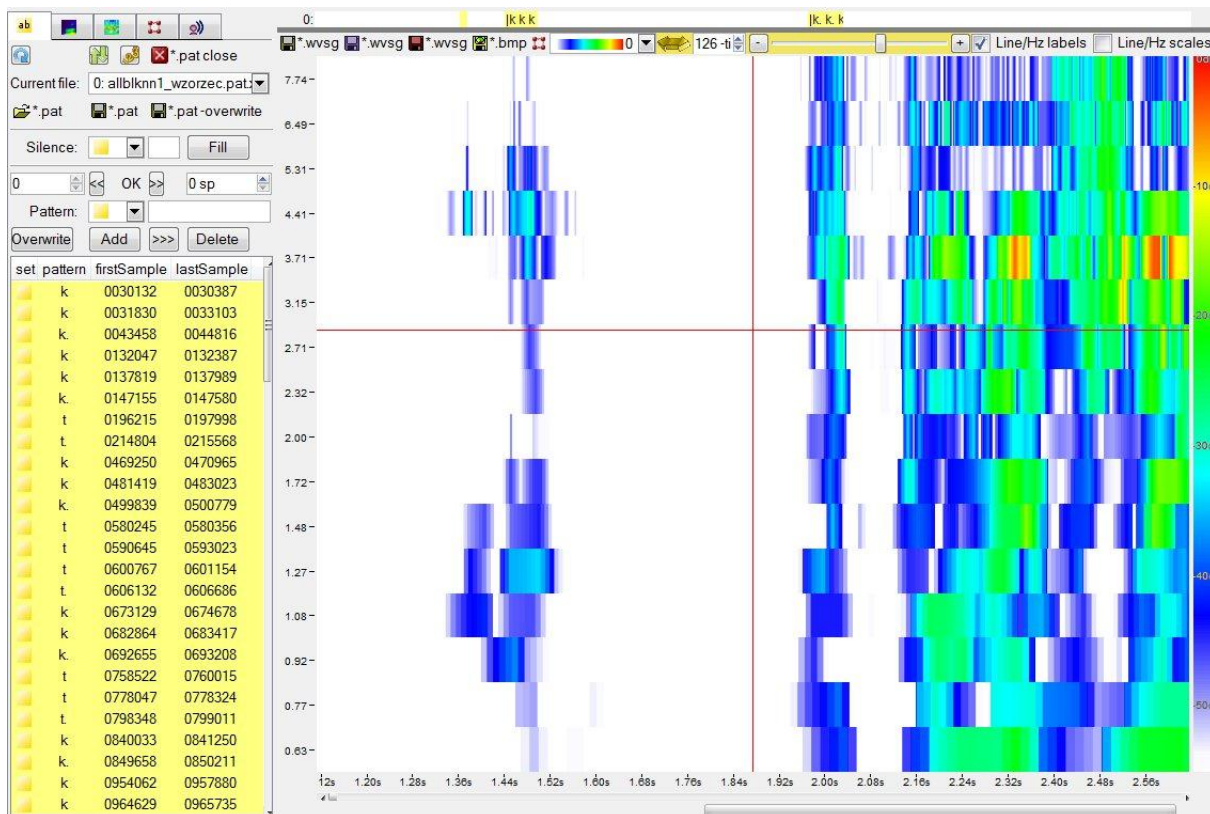
- Informacje – informuje, jaka operacja została wykonana,
- Komunikaty – są to informacje, które dodatkowo wyświetlają się w osobnym okienku (patrz rys. 10.8:2),
- Ostrzeżenia – są to informacje wyróżnione kolorem,
- Błędy – są to ostrzeżenia, które dodatkowo wyświetlają się osobnym okienku (patrz rys. 10.8:2). Ich następstwem jest zawsze przerwanie wykonywania operacji.



rys. 10.8:2 Panel komunikatów z liniami błędów oraz sprzężonym z tą funkcjonalnością okienkiem komunikatów



## 10.9. Panel wzorców

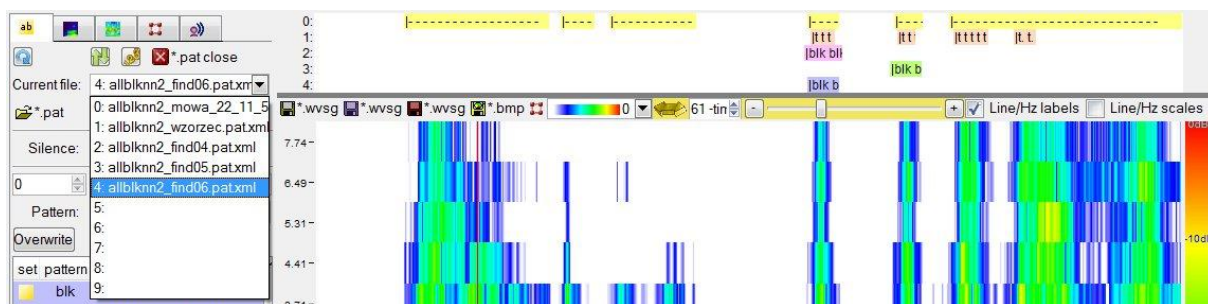


rys. 10.9:1 Panel wzorców (u góry) wraz z zakładką tworzenia wzorców (po lewej) programu WaveBlaster

Aspekty:

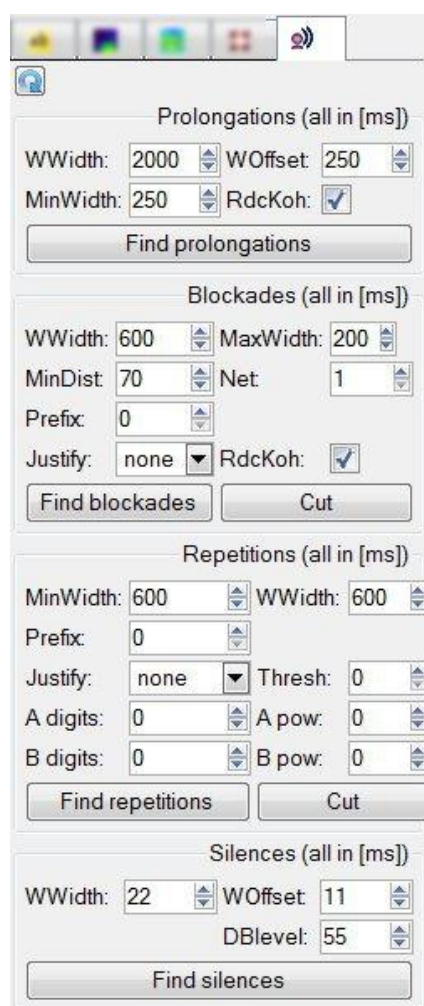
- manualnego (odsluchowego) oznaczania niepełności,
- automatycznego oznaczania i manualnego weryfikowania fonacji,
- automatycznego oznaczania i manualnego weryfikowania niepełności,
- automatycznego zliczania poprawnie i błędnie wykrytych niepełności

są elementami, na których opierają się badania prezentowane w tej rozprawie, zatem tworzenie, przeglądanie, edycja i zapisywanie wzorców to bardzo istotne funkcjonalności. W zakładce tworzenia wzorców możemy dowolnie tworzyć, edytować, listować wzorce. Zaznaczenie linii w tabelce wzorców automatycznie powoduje pojawienie się tych obszarów na widocznych panelach z danymi. Oczywiście wzorce można zapisywać i odczytywać z pliku. Możliwa jest praca aż z dziesięcioma jednocześnie otwartymi zestawami wzorców – jest to szczególnie przydatne przy porównywaniu różnych wyników detekcji fonacji lub niepełności (patrz rys. 10.9:2).



rys. 10.9:2 Panel wzorców programu WaveBlaster wyświetlający jednocześnie pięć zestawów danych

## 10.10. Zakładka automatycznej detekcji nie płynności



rys. 10.10:1 Zakładka automatycznej detekcji nie płynności programu WaveBlaster

Jest to miejsce uruchamiania wszystkich algorytmów automatycznej detekcji opisywanej w tej rozprawie. Na rys. 10.10:1 od góry mamy kolejno ramki:

- automatycznej detekcji przedłużeń – prolongations,
- automatycznej detekcji powtórzeń głosek – tu nazwanej blockades,
- automatycznej detekcji powtórzeń sylab – tu nazwanej repetitions,

- automatycznej detekcji ciszy – silences. Po dokonaniu inwersji zaznaczeń otrzymujemy detekcję fonacji.

Wynikiem działania każdego w/w algorytmów automatycznej detekcji jest jeden zestaw wzorców. Aby go zachować należy go zapisać do pliku korzystając z zakładki tworzenia wzorców. Takiego zestawu wzorców można użyć do dalszego przetwarzania np. obliczania statystyk lub inwersji wzorców.

### 10.11. Opcja uruchamiania skryptów

Pod względem programistycznym, konstrukcja programu ma bardzo dobrze odseparowany interfejs użytkownika od logiki (algorytmów) aplikacji. Oznacza to, że w łatwy sposób można stworzyć kod programu, który odwzorowuje każdą operację (sekwencję operacji) uruchomioną z poziomu interfejsu użytkownika. Wartości wszystkich opcji programu WaveBlaster (tych pokazanych w interfejsie użytkownika i tych niewidocznych) składowane są w jednym, głównym pliku konfiguracyjnym (patrz rys. 10.11:1)

```

<comp_settings>
  <common>
    <enableLP>0</enableLP>
    <enableLPformants>0</enableLPformants>
    <enableLPSpectrums>0</enableLPSpectrums>
    <enableEnergy>1</enableEnergy>
    <enableFT>0</enableFT>
    <forceFTcomputations>0</forceFTcomputations>
    <forceFTO3computations>0</forceFTO3computations>
    <forceENERGYcomputations>0</forceENERGYcomputations>
    <forceWAVELETcomputations>0</forceWAVELETcomputations>
    <forceLPcomputations>0</forceLPcomputations>
    <forceLPSPECTRUMScomputations>0</forceLPSPECTRUMScomputations>
    <forceLPFORMANTcomputations>0</forceLPFORMANTcomputations>
    <windowWidth>512</windowWidth>
    <windowOffset>256</windowOffset>
    <timeWindow>1</timeWindow>
    <logScale>1</logScale>
    <maxDB>55</maxDB>
  </common>
  <FT>
    <ftMinO3>7</ftMinO3>
    <ftMaxO3>29</ftMaxO3>
  </FT>
  <LP>
    <LPCparamCount>5</LPCparamCount>
    <LPCformantSensitivity>1</LPCformantSensitivity>
    <isPreemphasis>1</isPreemphasis>
    <preemphasisParam>937500</preemphasisParam>
  </LP>
</comp_settings>

```

rys. 10.11:1 Plik konfiguracyjny programu WaveBlaster. Przechowywane są w nim wartości wszystkich opcji interfejsu użytkownika.

Jest to szczególnie przydatna funkcjonalność programu WaveBlaster. Po zakończeniu wstępnych testów (tj. wymyślono procedurę detekcji, która może dać sensowne wyniki rozpoznawania) można przystąpić do systematycznego przeszukiwania przestrzeni danych

wejściowych algorytmu (na przykład sprawdzenie wyników detekcji przedłużeń dla 20-stu różnych konfiguracji parametrów). W tym celu nie trzeba ‘wyklikiwać’ wszystkich operacji.

Po zaimplementowaniu sekwencji operacji w kodzie (nazywaj tutaj skryptem) i nadaniu jej unikalnej nazwy, można stworzyć plik konfigurujący dane wejściowe dla tego skryptu (w formacie xml – patrz rys. 10.11:2). Dla każdego elementu <item> uruchamiany jest ten właśnie skrypt. Opcje algorytmu wczytywane są z pliku (takiego jak na rys. 10.11:2) oznaczonego tagiem <setting> dla każdego elementu osobno, dzięki czemu każdy element może być uruchomiony z innymi opcjami.

```
<?xml version="1.0" encoding="windows-1250"?>
<cos>
  <scriptEnum>CWtblkFind</scriptEnum>

  <settingDir>E:\Praca\VS_LPC wersja II\VC_LPC\LinearPredictiveCoding\wav\sett</settingDir>
  <BIGsettingFile></BIGsettingFile>
  <scaleDir>E:\Praca\VS_LPC wersja II\VC_LPC\LinearPredictiveCoding\wav\sett</scaleDir>
  <BIGscaleFile></BIGscaleFile>
  <resultDir>E:\Praca\VS_LPC wersja II\VC_LPC\LinearPredictiveCoding\wav\results\allblknn\ibiza\dist_30ms\
  </resultDir>
  <BIGresultFile>statystyki - pięć sprawdzeń - ponownie.txt</BIGresultFile>
  <waveDir>E:\Praca\VS_LPC wersja II\VC_LPC\LinearPredictiveCoding\wav\blknn</waveDir>
  <patternDir>E:\Praca\VS_LPC wersja II\VC_LPC\LinearPredictiveCoding\wav\sett</patternDir>

  <item>
    <wave>allblknn1.wav</wave>
    <pattern>allblknn1_wzorzec.pat.xml</pattern>
    <pattern2>allblknn1_mowa_22_11_54_20Hz.pat.xml</pattern2>
    <setting>CWtblkFind.sett.xml</setting>
    <scale>barki_bez_1_5.wvlc.xml</scale>
    <attribute>13</attribute>
  </item>
  <item>
    <wave>allblknn2.wav</wave>
    <pattern>allblknn2_wzorzec.pat.xml</pattern>
    <pattern2>allblknn2_mowa_22_11_54_20Hz.pat.xml</pattern2>
    <setting>CWtblkFind.sett.xml</setting>
    <scale>barki_bez_1_5.wvlc.xml</scale>
    <attribute>13</attribute>
  </item>
  <item>
    <wave>allblknn3.wav</wave>
    <pattern>allblknn3_wzorzec.pat.xml</pattern>
```

rys. 10.11:2 Przykładowy plik konfigurujący dane wejściowe dla skryptu o nazwie CWtblkFind (pierwsza linia konfiguracji)

## 11. Dodatek B



### BEST PAPER AWARD

TO

**IRENEUSZ CODELLO, WIESŁAWA KUNISZYK-JÓŹKOWIAK,  
ELŻBIETA SMOŁKA AND ADAM KOBUS**

FOR THEIR PAPER:

**AUTOMATIC DISORDERED SYLLABLES REPETITION RECOGNITION IN  
CONTINUOUS SPEECH USING CWT AND CORRELATION**

IN THE 8<sup>TH</sup> INTERNATIONAL CONFERENCE ON COMPUTER  
RECOGNITION SYSTEMS, MAY 27-29 2013, MIŁKÓW, POLAND.

Organized by Department of Systems and Computer Networks

Miłków, May 29, 2013

Sincerely,  
Dr hab. inż. Michał Woźniak, prof. PWr  
Chair of the Organizing Committee



Wrocław University of Technology