

## **Sprawozdanie z laboratorium Podstaw i Algorytmów Przetwarzania Sygnałów**

<b>Ćwiczenie wykonał:</b> Karol Kozłowski (132652) Stanisław Kozuchowicz (132655)	<b>Data :</b> 12 marzec 2006	<b>Prowadzący:</b> Jarosław Lachowski	<b>Ocena:</b>
---	---------------------------------	--	---------------

### **“Rozpoznawanie mowy”**

#### **• Zadanie projektowe**

Celem projektu było napisanie aplikacji (skryptu) w języku programowania Matlab/Octave, który poprzez analizę podanego sygnału mowy będzie rozpoznawał cyfrę, którą dany sygnał reprezentuje.

#### **• Metoda analizy**

W projekcie zastosowaliśmy jedną z najprostszych metod rozpoznawania mowy – analizę gęstości przejść przez zero. Polega ona na podzieleniu badanego sygnału na  $n$  równych przedziałów a następnie wyznaczeniu ilości przejść sygnału przez zero w każdym z tych przedziałów. Utworzony w ten sposób histogram stanowi na tyle wierną reprezentację sygnału, aby móc na jego podstawie zidentyfikować wypowiedzianą cyfrę.

#### **• Aplikacja**

Nasza implementacja opisanej powyżej metody obejmuje zestaw skryptów:

- *wz.m* – plik funkcji generującej wzorzec
- *learn.m* – skrypt generujący bazę wzorców
- *test.m* – funkcja testująca

Pierwszym krokiem algorytmu jest utworzenie bazy wzorców, z którymi porównywane będą analizowane sygnały. Operację tę wykonuje skrypt *learn.m* który zbiera informację o sygnałach wzorcowych (katalogi *./karol/\** oraz *./staszek/\**). Wywołuje on dla każdego sygnału funkcję *wz*, która tworzy jego histogram (ilość przedziałów ustalana jest w pliku *learn.m* zmienną *d* i przechowywana jest w bazie danych razem z wzorcami). Po przeanalizowaniu kilku próbek dźwiękowych dla pojedynczego słowa obliczany jest ogólny wzorzec (histogram) reprezentujący to słowo (przyjętą metodą jest średnia arytmetyczna histogramów składowych). Po utworzeniu całej bazy zapisywana jest ona w pliku *data.fon*. Od tego momentu utworzona jest baza, na podstawie

której można prowadzić rozpoznawanie słów.

Detekcja słów odbywa się za pomocą funkcji *test.m* której jedynym parametrem jest ścieżka do sygnału (pliku) zawierającego słowo, które chcemy rozpoznać (dla zachowania kompatybilności ilość przedziałów na które dzielony jest sygnał pobierana jest z bazy danych). Odbywa się to w następujący sposób: w pierwszej kolejności obliczany jest histogram badanego sygnału, następnie porównywany jest on z każdym z wzorców znajdujących się w bazie. Odległość pomiędzy sygnałami określona jest jako bezwzględna suma współrzędnych wektora, będącego różnicą histogramów wzorca i badanego sygnału. Najmniejsza odległość pomiędzy badanym sygnałem i którymś ze wzorców przyjmowana jest za pozytywny wynik wyszukiwania. Następnie funkcja kończy swoje działanie zwracając wartość rozpoznanej cyfry.

### • Obserwacje i wnioski

W pierwszej wersji skryptu do porównywania wzorców została użyta funkcja korelacji wzajemnej, lecz wyniki jakie dawała były dalekie od oczekiwanych. Dopiero zastosowanie opisanej powyżej metody pozwoliło osiągnąć ok 80% skuteczność w rozpoznawaniu sygnałów.

Ilość przedziałów na jakie dzielony jest sygnał ma duże znaczenie na jakość otrzymywanych wyników. Za mała ilość przedziałów wpływa negatywnie na uzyskiwane rezultaty. Doświadczalnie stwierdzono, że wartością dla której uzyskuje się najlepsze wyniki jest  $p=21$  (8 błędów). Wyniki analizy przedstawione są na *wykresie 1*. Jak łatwo zauważyć zwiększanie ilości przedziałów powyżej pewnej wartości (ok 20) nie wpływa znacząco na poprawę jakości a zwiększa ilość danych potrzebnych do zapisania jednego wzorca.

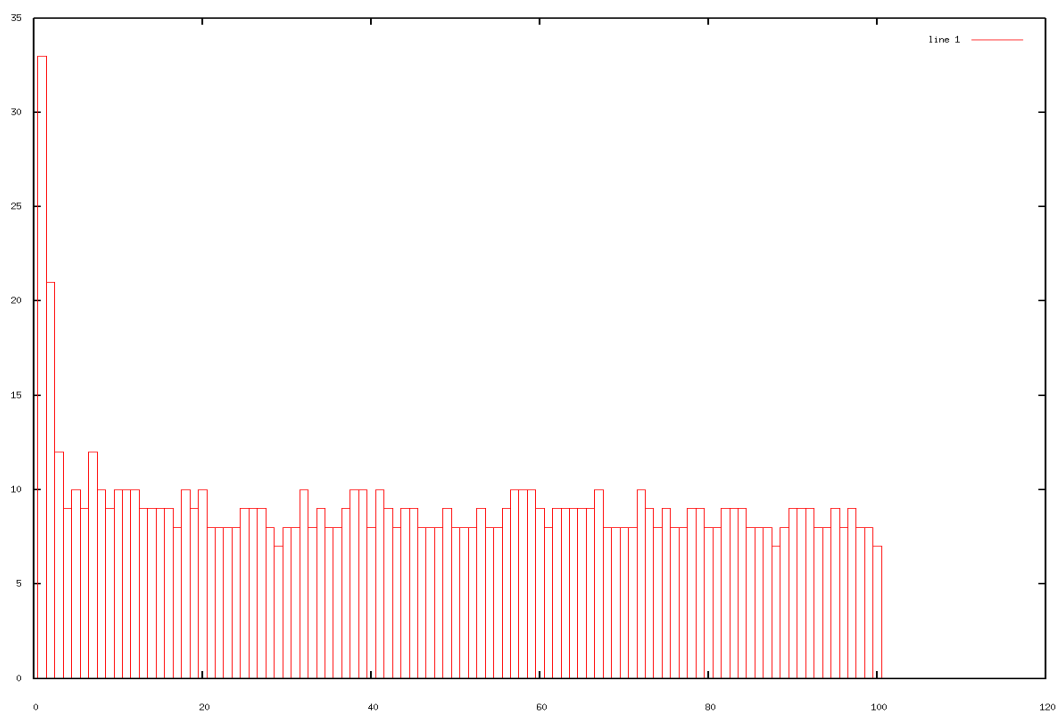
Na *wykresie 2* przedstawiono zależność odległości wzorców dla poszczególnych słów. Zgodnie z teorią detekcji idealna krzywa powinna mieć duży kąt nachylenia do osi x, jak widać w tym przypadku odbiega to od założeń teoretycznych. Spowodowane to jest najprawdopodobniej niedokładnym przygotowaniem próbek wzorcowych, bądź szumem występującym w nagraniach. Wpływ na to ma również wybrana metoda porównywania wzorców.

Do realizacji algorytmu użyliśmy 6 serii danych, które zostały przeanalizowane przez skrypt “uczący” i na ich podstawie utworzony został wzorzec. Następnie każdy z elementów serii został przetestowany pod kątem poprawności rozpoznania. Przy dobraniu optymalnej liczby interwałów czasowych (minimalizacja ilości błędów) udało nam się uzyskać poprawność rzędu 80%.

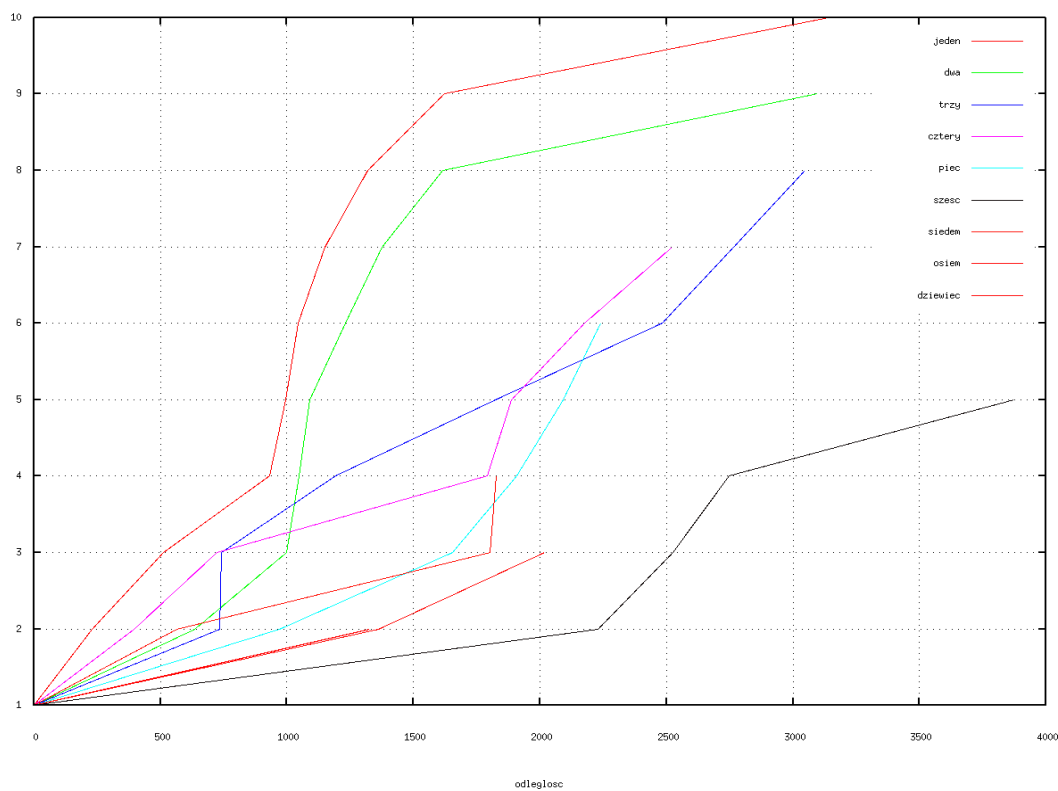
Aplikacja zawiera w sobie również dodatkowe skrypty nie wymienione wcześniej, należą do nich:

- *bestp.m* – wyznacza wartość  $p$  dla której występuje najmniej błędów detekcji,
- *bestp\_bar.m* – ilustruje wyniki działania *bestp.m* zapisane w pliku *bestp.dat* (*wykres 2*),

- *far.m* – wyświetla wykres odległości wzorców (wykres 1),
- *stats.m* – analizuje wzorzec i pliki źródłowe w celu odnalezienia błędu detekcji.



Wykres 1: wykres zależności błędnych rozwiązań od ilości interwałów



Wykres 2: Wykres ilustrujący odległość pomiędzy poszczególnymi słowami