Rafał Cieślak
nr albumu : 34203
kierunek studiów: Informatyka
specjalność: Systemy komputerowe i opramowanie
forma studiów: stacjonarne

Identyfikacja akustyczna rodzaju zdania w systemach dialogowych

Acoustic identification of sentence type in dialogue systems

Praca dyplomowa inżynierska wykonana pod przewodnictwem:
dr inż. Tomasz Mąka

Szczecin 2019

Spis treści

Spis rysunków

Spis tablic

List of equations

1 Wstęp

2 Cel pracy

3 Wprowadzenie teoretyczne

3.1 Sygnał mowy

Mową okreslamy komunikowanie się między sobą ludzi, za pomocą ukształtowanego zbioru dźwięków i reguł, zwanego językiem. Każdy język używa własnych fonetycznych kombinacji zbioru spółgłosek i samogłosek, które tworzą słowa mające semantyczne znaczenie. W czasie mówienia, osoba mówiąca poza samym wypowiadaniem słów, nadaje wypowiedzi znaczenie również za pomocą dodatkowych aspektów, takich jak intonacja, tempo mówienia czy stopień głosnosci. Sama produkcja mowy jest wielokrokowym procesem zamiany myśli w ustną wypowiedź, która może być zarejestrowana jako sygnał mowy.

3.1.1 Powstawanie mowy

Sygnal mowy ludzkiej jest sygnałem akustycznym powstającym podczas przepływu powietrza poprzez aparat mowy, który jest definiowany jako 3 osobne grupy narządów.

Składowymi aparatu mowy są:

1. Aparat oddechowy. Bierze udział w początkowej fazie powstawania mowy, dostarczając kolejnym składowym strumień powietrza, który jest niezbędny do wygenerowania drgań. Dzieje się to podczas wydechu. Elementy, z których jest zbudowany to płuca, oskrzela, przepona oraz tchawica.

2. Aparat fonacyjny, którego głównym elementem jest krtań. Jest to narząd niezbędny do wygenerowania jakiegokolwiek dźwięku, nie tylko mowy. Najważniejszym elementem krtani, w kontekście procesu powtarzania dźwięku, są fałdy głosowe. W ich skład wchodzą więzadła głosowe oraz mięśnie głosowe. Przestrzeń pomiędzy nimi nazywana jest szparą głośni. Struktury te przybliżają się i oddalają od siebie podczas powstawania dźwięku co powoduje zwarcie i rozwarcie szpary głośni. Podczas oddychania oraz przy generowaniu głosek bezdźwięcznych, fałdy są rozsunięte, natomiast zwierają się i rozwierają podczas powstawania głosek dźwięcznych.

Rysunek 1: Fałdy głosowe http://terapiamowy.com/index.php/niedowlad-krtani/

Dzięki tej czynności, strumień powietrza wprowadzany jest w drgania, co postrzegamy jako dźwięczność. Cecha ta występuje wraz z każdą samogłoską oraz przy niektórych spółgłoskach. Podczas drgań generowany jest ton krtaniowy, zwany również częstotliwością podstawową, oznaczany w literaturze jako F0.

3. Aparat artykulacyjny, w którego skład wchodzą jamy przewodu oddechowego, znajdującego się ponad krtanią. Najważniejsze z punktu widzenia artykulacji - nosowa, gardłowa oraz ustna - nazywane są nasadą. Artykulatory znajdujące się w nasadzie dzielone są na ruchome oraz nieruchome. Do ruchomych zaliczamy język, podniebienie miękkie, wargi oraz żuchwę. Nieruchomymi określamy zęby, dziąsła oraz podniebienie twarde. Ich ustawienie ostatecznie determinuje cechy wytwarzanego dźwięku.

Cały proces powstania dźwięku, nazywany jest fonacją. W początkowej fazie jego przebiegu wzrasta cisnienie w płucach, co prowadzi do wydechu. Powietrze dostaje się do tchawicy. Na szczycie tchawicy znajduje się krtań, należąca do aparatu fonacyjnego. W miarę przepływu powietrza przez głosnię, spada lokalne cisnienie, co pozwala mięsniom krtani zamknąć głosnię, przerywając przepływ powietrza. To powoduje wzrost cisnienia, prowadzący do kolejnego oddalenia się strun głosowych. Cały ten cykl zapętla się, tworząc dźwięk, kierowany do aparatu artykulacyjnego. Na tym etapie, poza artykulacją, zachodzi również tłumienie niektórych częstotliwosci, nie będących harmonicznymi fali głosniowej. Nie wytłumione zostają tylko częstotliwosci będace bliskie naturalnemu rezonansowi traktu głosowego. Jako rezultat kompletnego procesu, uzyskiwana jest fala akustyczna, wydostająca się z ust. Ruszając szczęką, ustami lub zmieniając położenie języka, możemy zmieniać uzyskiwany dźwięk, ponieważ zmieni się rezonans traktu głosowego, a zatem inne częstotliwosci zostaną wytłumione. Fakt, że wiele różnych narządów

bierze udział w tworzeniu mowy powoduje, że zaburzenia zdrowotne każdego z nich mają istotny wpływ na cały proces. Zakres powstałych w ten sposób zaburzeń mowy jest szeroki - od drobnych wad wymowy do całkowitej utraty mowy.

3.1.1.1 Dźwięczność głosek

3.1.2 Reprezentacja mowy

W procesie rozwoju technologii związanych z przetwarzaniem mowy, konieczne bylo ustalenie sposobu przedstawienia wypowiedzi za pomocą symboli reprezentujacych wyprodukowany sygnal. Litery, używane w tym celu w języku pisanym, są niewystarczające, ponieważ w różnych wyrazach mogą być wymawiane na rózne sposoby. Często produkowany dźwięk dla danej litery różni się w zależnosci od otaczających ją liter. Dla języka polskiego charakterystyczne jest występowanie tak zwanych dwuznaków, na przykład "rz,sz,ch". Dźwięk produkowany dla tych znaków jest calkowicie odmienny od dźwięków reprezentujących każdą z liter osobno. Jednym ze sposobów reprezentowanie dźwięków powszechnie występujących w danym języku są fonemy. Są to najmniejsze elementy języka mówionego, pozwalające na rozróżnienie poszczególnych słów. Często po zamienieniu jednego z fonemów składowych na inny, znaczenie słowa może ulec zmianie. W lingwistyce istnieją rózne sposoby definiowania czym są fonemy oraz w jaki sposób dany język powinien być przez nie reprezentowany. Najczęsciej jednak fonem jest rozumiany jako często powtarzający się w danym języku zbiór głosek. W języku polskim, w zależnosci od sposobu definiowania, liczba fonemów waha sie od 31 do 42.

3.1.3 Rozumienie mowy

Rozumieniem mowy nazywany jest proces, w trakcie którego wypowiedziana mowa jest słyszana, interpretowana oraz rozumiana przez człowieka. Badania nad postrzeganiem mowy są scisle związane z lingwistyką oraz psychologią poznawczą i próbują odpowiedzieć na pytanie w jaki sposób ludzie rozpoznają dźwięki mowy i na ich podstawie rozumieją mówiony język. Rezultaty tych poszukiwań mają swoje zastosowania w tworzeniu systemów komputerowych służących rozpoznawaniu mowy. Rozumienie mowy w danym języku jest scisle związane z rozpoznawaniem przez mózg fonemów charakterystycznych dla tego języka. Z tego powodu często ludzie uczący się obcego języka znacznie łatwiej przyswajają język w formie pisanej niż mówionej.

3.1.4 Rejestrowanie sygnału mowy

Dźwięk opuszczający aparat mowy może zostać zarejestrowany przez mikrofon w celu poddania szczegółowej analizie. Aby możliwe było przetwarzanie sygnału przez program komputerowy, konieczne jest przetworzenie sygnału z postaci analogowej do cyfrowej. W tym celu pobiera się próbki sygnału. Wartość określającą ilość próbek w jednostce czasu nazywamy częstotliwością próbkowania. Najczęściej spotykana wartość to 44,1 kHz. Oznacza to, że podczas sekundy pobierane jest 44100 wartości sygnału ciągłego. Liczba ta została przyjęta jako standard przy nagrywaniu audio na płytach CD. Tak pobrane próbki, po poddaniu procesowi kwantyzacji, tworzą sygnał cyfrowy. Sygnał dźwiękowy może być nagrywany w wersji monofonicznej lub stereofonicznej. Oznacza to użycie jednego lub dwóch (lewy,prawy) kanałów. Nagrania rejestrowane tymi sposobami różnią się od siebie diametralnie, zarówno w kontekście subiektywnych odczuć słuchacza, jak i podczas przetwarzania sygnału. Kanały w wersji stereofonicznej mogą różnić się od siebie wartościami próbek, zwłaszcza w widmie sygnału.

3.2 Ton podstawowy

3.2.1 Definicja tonu podstawowego

W literaturze własnosć bywa również nazywana częstotliwoscią podstawową lub po prostu oznaczana jest jako F0.W zależności od potrzeb, ton podstawowy bywa różnie definiowany. W kontekście przetwarzania sygnału mowy rozumiany jest jako wibracje strun głosowych, towarzyszące powstawaniu głosek dźwięcznych. Powstałe w ten sposób częstotliwosci mieszczą się w zakresie 85-180Hz dla mężczyzn oraz w zakresie 165-255Hz dla kobiet.Wartosci te mogą być wyższe gdy osoba mówiąca znajduje się pod wpływem silnych emocji. Poza płcią oraz stanem emocjalnym, zależne są również od wieku, budowy i kształtu strun głosowych ogólnego stanu zdrowia oraz rodzaju wypowiedzi. Badania nad częstotliwoscią podstawową produkowaną przez mężczyzn pokazały, że jej srednie wartosci spadają po osięgnięciu 35 roku życia, by ponownie ulec wzrostowi po przekroczeniu 55 roku życia. (Hollien and Ship, 1972; Kitzing, 1979; Pegoraro-Krook, 1988). W przypadku kobiet, wartosci F0 zaczynaja spadać w okresie menopauzy, osiagając finalne wartosci okolo 70 roku zycia. (Chevrie-Muller et al., 1971; Kitzing, 1979; Stoicheff, 1981; Pegoraro-Krook, 1988). Badania nad wpływem palenia papierosów na wartosci F0 pokazały, że wieloletnie palenie również doprowadza do obniżenia tych wartosci, jako że nawyk ten wpływa negatywnie na krtań. (Gilbert and Weismer, 1974). Przebieg częstotliwości podstawowej w dużym stopniu odzwierciedla intonację wypowiedzi. Gdyby ten przebieg był stały, mowa byłaby odbierana jako monotonna lub brzmiąca maszynowo. Pełni istotną funkcję w językach tonalnych, w których wielu słów jest zapisywanych tak

samo, a jedynie nadawany im ton pozwala rozróżnić ich znaczenie. Z tego powodu też poprawna estymacja F0 jest konieczna w systemach rozpoznawania mowy dla języków tonalnych. Dla idealnie okresowego sygnału, częstotliwość podstawowa byłaby po prostu odwrotnością okresu. Okresem nazywamy czas pomiędzy kompletnym cyklem otwarcia i zamknięcia głosni. Jednak sygnał mowy jest sygnałem bardzo dynamicznym, co sprawia, że estymacja F0 przestaje być zadaniem trywialnym. Dodatkowo transformacja sygnału analogowego do postaci dyskretnej, wiążąca się zawsze z utratą danych oraz towarzyszący nagranemu głosowi szum wpływają negatywnie na dokładność estymacji.

3.2.2 Formanty

Częstotliwosć podstawowa powiązana jest w największej mierze z intonacją. Jednak w badaniach związanych z technologią przetwarzania mowy, wyznaczane z sygnału mowy sa również inne częstotliwosci, związane z rezonansem innych częsci traktu głosowego. Nie są one bezposrednio związane z intonacją, lecz wiedza na ich temat jest istotna dla każdych badań związanych z sygnałami mowy. Są to formanty. Pod tym pojęciem rozumiane sa skupiska energii akustycznej, zgromadzone wokół konkretnej częstotliwosci w sygnale mowy. Istnieje kilka formantów, lecz zazwyczaj wyznaczane sa cztery - F1, F2, F3, F4. Każdy z nich występuje na innej częstotliwosci. W dużym przybliżeniu można stwierdzić, że F1 występuje w okolicach 500 Hz, a kolejne formanty są zlokalizowane na częstotliwosciach będących kolejnymi nieparzystymi wielokrotnosciami pierwszego formantu.

3.2.3 Przegląd metod estymacji

Prowadzone badania nad częstotliwością podstawową doprowadziły do wynalezienie wielu algorytmów estymacji o różnej skuteczności, zarówno w dziedzinie czasowej jak i widmowej. Przykłady metod czasowych:

1. Analiza funkcji autokorelacji, polegająca na badaniu korelacji między danymi wejściowymi sygnału przy różnych opóźnieniach. Implikuje to wiele operacji mnożenia oraz dodawania. Estymacja F0 z wykorzystaniem tej metody związana jest z wykrywaniem maksimów lokalnych funkcji autokorelacji.

2. AMDF (Average Magnitude Difference Function) będąca odmianą funkcji autokorelacji. Polega na analizie relacji sygnału do jego opóźnionej w czasie wersji. Jako, że nie występują tu operacje mnożenia, złożoność czasowa tego algorytmu jest niższa.

Przykładem metody widmowej jest metoda cepstralna. „W metodzie tej obliczana jest odwrotna transformata Fouriera logarytmu widma amplitudowego analizowanej ramki sygnału wg wzoru:” Dzięki analizie pozycji maksimum w dziedzinie cepstralnej, możliwe jest oszacowanie F0.

3.2.4 Definicja algorytmu YIN

Metody widmowe umożliwiały lepszą dokładność estymacji częstotliwości podstawowej, do momentu opracowania algorytmu YIN. W podstawowej wersji bazuje na analizie funkcji autokorelacji w dziedzinie czasu. Jego autorami są Hideki Kawahara oraz Alain de Cheveigne, którzy zaprezentowali te podejście w 2002 roku. Algorytm ten posiada kilka własności, dających mu przewagę nad konkurencyjnymi metodami. Nie posiada górnego limitu frekwencji, dla których działa poprawnie, dzięki czemu wyniki nie są zakłamywane dla wysokich głosów. Ta cecha jest również znacząca w użyciu algorytmu do analizy muzyki. Ważną własnością jest fakt, że algorytm ten jest relatywnie prosty, co pozwala na efektywną implementację, bez dużych opóźnień. Na jego prostotę istotnie wpływa niewielka liczba wymaganych parametrów.

3.3 Intonacja

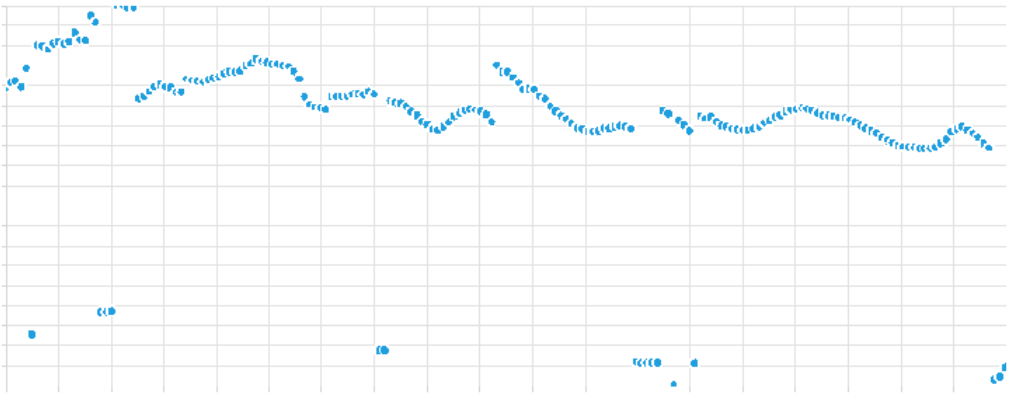
Intonacja jest zmianą tonu podstawowego, nie wpływającą na rozpoznawanie słów. Jest jedną z trzech głównych brzmieniowych właściwości mowy, obok akcentu i iloczasu. Najczęściej jest dodawana podczas wypowiedzi w celu oddania emocji. W wielu językach, w tym także w polskim, nadawanie wypowiedzi określonej intonacji może determinować jej typ. W pewnych sytuacjach modulacja intonacyjna może być jedyną informacją pozwalającą rozmówcy zrozumiec czy wypowiedź była twierdzeniem czy pytaniem. Przykład takiego zdania:
Musisz jutro wcześnie wstać.
Musisz jutro wcześnie wstać?
Jako, że taki szyk zarówno zdania jak i pytania jest całkowicie poprawny w języku polskim, bez nadania wypowiedzi odpowiedniej intonacji odbiorca nie jest w stanie zrozumieć intencji osoby mówiącej.

3.3.1 Typy intonacji

Najczęściej rozróżniane są dwa typy intonacji, opadająca oraz rosnąca. Opadająca, zwana kadencją zwyczajowo kojarzona jest ze zdaniami twierdzącymi, natomiast intonacja rosnąca, znana jako antykadencja, określana jest jako pytanie. W rzeczywistości podział nie jest tak klarowny. Należy wziąć również pod uwagę kontury z intonacją będącą połączeniem dwóch podstawowych zmian, czyli intonację rosnąco – opadająca oraz opadająco – rosnąca. Rozróżniany jest również brak wyraźnych zmian w przebiegu tonu podstawowego, zwany progrediencją. Jest on charakterystyczny dla tekstu czytanego.

3.3.2 Przebiegi intonacji dla poszczególnych rodzajów zdań

Ogólna tendencja tonu podstawowego dla zdań twierdzących jest rzeczywiście spadkowa. Powodowane jest to najprawdopodobniej kładzeniem większego akcentu na pierwszą część zdania, zwłaszcza na pierwszy wyraz

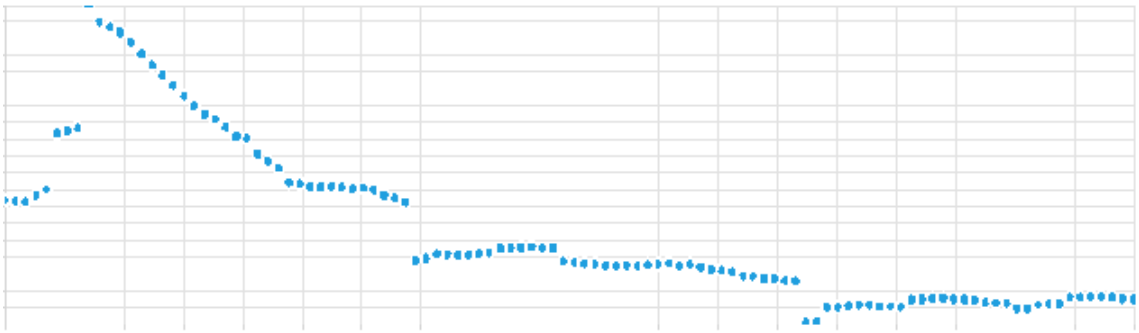


Rysunek 2: Przebieg konturów intonacyjnych dla zdania „Intonacja w zdaniach twierdzących jest opadająca". Próbkowanie 44kHz, estymacja z wykorzystaniem algorytmu YIN

W pytaniach sytuacja ma się jednak inaczej, wbrew opinii powtarzanej w wielu źródłach, nie można wprost zakładać, że dla pytania intonacja będzie miała przebieg rosnący. Istnieje duża grupa pytań, dla których w tym przypadku intonacja również może mieć przebieg opadający lub będący połączeniem opadającego oraz rosnącego, w różnej kolejności. Powodowane jest to faktem, że w języku polskim rozróżniamy dwa rodzaje pytań. Wyróżnił je Kazimierz Ajdukiewicz w podręczniku „Logika Pragmatyczna" wydanym w 1965 roku. Na podział ten składają się:

1. Pytania rozstrzygnięcia. Są to pytania, na które można udzielić odpowiedzi tak lub nie. Przykładem może być pytanie: „Czy byłeś dzisiaj w pracy?" Cechują się silną antykadencją, umiejscowioną zazwyczaj w ostatnim wyrazie. Mogą występować również bez partykuły „czy", na przykład pytanie „Mógłbyś to zrobić?" również jest pytaniem rozstrzygnięcia. Poprawna klasyfikacja takiej wypowiedzi na podstawie intonacji jest zadaniem stosunkowo prostym.

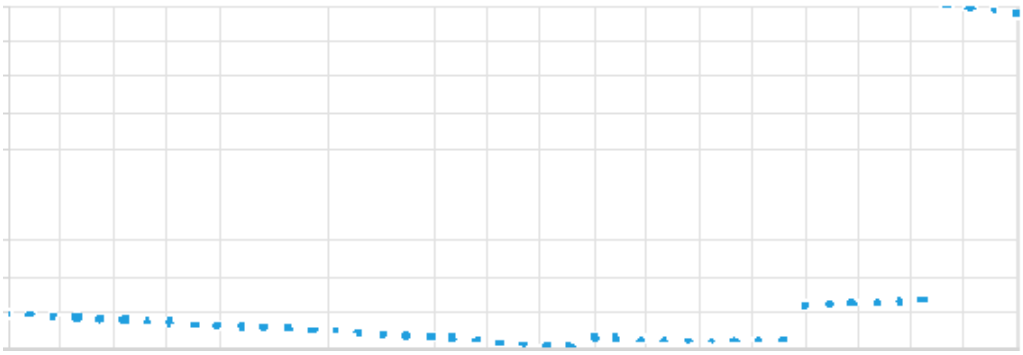
2. Pytania uzupełnienia. Nazywamy tak pytania, na które nie można udzielić odpowiedzi twierdzącej lub przeczącej. W wypowiedziach tych, akcent intonacyjny, mogący wskazywać, ze jest to pytanie kładziony jest na zaimek pytajny, będący najczęściej pierwszym wyrazem.



Rysunek 3: Przykład pytania o uzupełnienie : „Jaką intonację ma to pytanie?”

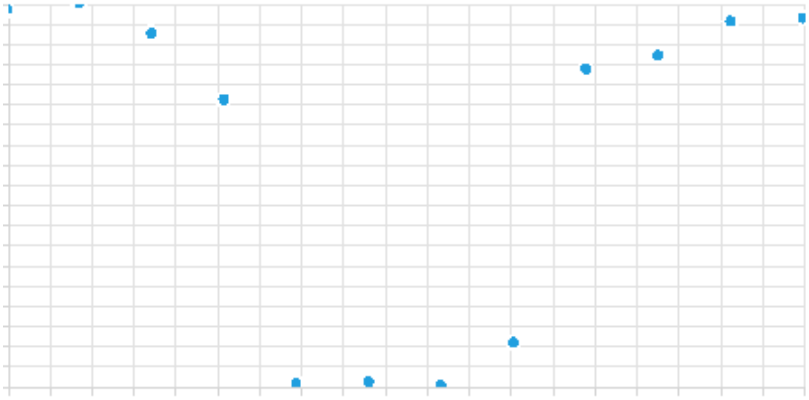
Silny wzrost intonacji występuje na samym początku sygnału, następnie widoczny jest niemal ciągły spadek. Istnieją również pytania o uzupełnienie, dla których wzrost intonacji ma swoje miejsce na końcu wypowiedzi, obrazując typową antykadencję:

(a) Pytania odwrócone, np. W zeszłym roku byłem we Francji. A w Belgii? (= Kiedy byłeś w Belgii?)



Rysunek 4: Jest to pytanie odwrócone, które brzmi „A w Belgii?”

(b) Antykadencja występuje także w pytaniach nawiązujących upewnienia. Występuje wówczas zarówno w pytaniach o rozstrzygnięcie, jak i o uzupełnienie, a jej funkcją jest domaganie się powtórzenia i potwierdzenia informacji, np. Byłem tam wczoraj. Kiedy? Wczoraj."[(Leokadia Dukiewicz, Irena Sawicka, "Fonetyka i fonologia", Kraków 1995)]



Rysunek 5: Przykład pytania nawiązującego upewnienie : „Kiedy?”

Przebieg intonacji dla takiego pytania jest opadająco-rosnący.

3.4 Analiza dotychczasowych badan

4 Implementacja detekcji konturów częstotliwości podstawowej

Celem praktycznym pracy jest stworzenie aplikacji desktopowej umożliwiającej wczytanie próbki zawierającej nagrane zdanie oraz dokonanie identyfikacji rodzaju tej wypowiedzi

4.1 Język programowania oraz środowisko

Pierwszym rozważanym zagadnieniem był wybór języka programowania oraz środowiska. Należało wziąć pod uwagę zawartość bibliotek związanych z przetwarzeniem dźwięku, oferowanych przez poszczególne języki. Mimo rozpatrywania możlwości wielu języków, główny wybór zawarty był między Javą oraz C++. Dla obu języków dostępna jest mnogość gotowych funkcji wspierających pracę z dźwiękiem. Jako, że projekt zakładał stworzenie graficznego interfejsu użytkownika, konieczny był również wybór odpowiedniego środowiska, umożliwiającego stworzenie takiej aplikacji. Dla języka Java jako środowisko spełniające takie wymagania postrzegany był Eclipse wraz z frameworkiem JavaFx. Nie posiadają one wbudowanych pomocy do pracy z próbkami dźwięku, lecz dla Javy stworzone zostało Java Sound API. API te zawiera podstawowe funkcjonalności, jest pomocne przy wczytywaniu plików wav. W celu korzystania z tego rozszerzenia, należy je po prostu zaimportować. Dla C++ sytuacja wygląda zgoła inaczej. Pracując z tym językiem, można korzystać z możliwości obszernego frameworka - Qt. Oferuje on wiele węwnetrznych klas ułatwiających pracę z dźwiękiem. Działają one niskopoziomowo, wszelkie zadania wykonywane są dużo szybciej niż w przypadku Javy. System sygnałów i

slotów, charakterystyczny dla Qt, jest bardzo wygodny przy wczytywaniu kolejnych próbek dźwieków. Umożliwia to aktualizowanie wykresów przedstawiających odczytane lub obliczone wartości na bieżąco. Dodatkowo, tworzenie graficznego interfejsu użytkownika w tym środowisku jest bardziej intuincyjne. Biorąc pod uwagę argumenty, wybór padł na język C++ z wykorzystaniem frameworka Qt.

4.2 Opis możliwości aplikacji

W pierwotnym założeniu aplikacja miała umożliwiać nagrywanie wypowiedzi, która następnie miała zostać poddana rozpoznaniu. Jednak w trakcie implentacji nie sposób było nie zauważyć, że znacznie lepsze wyniki rozpoznania są uzyskiwane, gdy do programu zostanie wczytana wypowiedź nagrana zewnętrznym programem, oraz poddana w nim obróbce wstępnej. Spowodowało to porzucenie tej funkcjonalności, jako że nie jest ona konieczna do osiągnięcia zakładanego celu, jakim jest poprawne rozpoznawanie rodzaju wypowiedzi.

Aplikacja umożliwa wczytanie pojedynczego nagrania lub całego katalogu z nagraniami. Program wyświetla nazwę wczytanego pliku, oraz rodzaj zdania do jakiego dana wypowiedź została sklasyfikowana. Po kliknięciu w tabeli na wybrany wiersz, a następnie po kliknięciu na jeden z dowolnych przycisków w dolnym pasku, program wyświetli na wykresie odpowiednio energię nagrania, przebieg widma, przebieg wartości próbki w dziedzinie czasu (waveform) lub przebieg wyestymowanej częstotliwości podstawowej.

4.3 Wczytanie próbki

Pierwszym krokiem na drodze do rozpoznania rodzaju zdania, jest wczytanie calego nagrania przez program. Wykonuje sie to z wykorzystaniem mozliwosci oferowanych przez Qt. Framework oferuje do tego klase QAudioDecoder. Nagranie jest wczytywane w 100 milisekundowych fragmentach. Jako że częstotliwość próbkowania wynosi 44100Hz, na jeden fragment przypada 4410 wartości. Każda część jest odczytana jako obiekt klasy QAudioBuffer. Wektor typu QAudioBuffer zawiera cale wczytane nagranie.

```
std::vector<QAudioBuffer>audioBuffers;
QAudioDecoder *audioDecoder;

audioDecoder = new QAudioDecoder();
connect(audioDecoder, SIGNAL(bufferReady()), this, SLOT(readBuffer()));
connect(audioDecoder,SIGNAL(finished()),this,SLOT(decodingFinished()));
audioDecoder->start();
```

Po wczytaniu kazdej z ramek emitowany jest sygnal. Laczac sygnal ze slotem, mozliwe jest przechwycenie aktualnie wczytanych wartosci, zanim zostana zastapione wartosciami kolejnej ramki. Zostaja one dodane do wektora ramek.

```
void MainWindow::readBuffer()
{
    audioBuffers.emplace_back(audioDecoder->read());
}
```

Gdy cale nagranie zostanie odczytane, QAudioDecoder emituje sygnal finished(). Po jego przechwyceniu, a wiec otrzymaniu informacji o zakonczeniu dekodowania, program umieszcza w jednym wektorze próbki ze wszystkich 100 milisekundowych buforów.

Listing 1: Funkcja dodająca do wektora wszystkie odczytane próbki

```
void MainWindow::putValuesIntoVector()
{
    sampleRate = audioBuffers[0].format().sampleRate();
    frameSize = audioBuffers[0].format().sampleRate()/40;

    for (QAudioBuffer audioBuffer : audioBuffers)
    {
        const qint16 *data = audioBuffer.constData<qint16>();
        for(int j=0;j<audioBuffer.sampleCount();j++)
        {
            wholeBuffer.emplace_back(data[j]);
        }
        delete data;
    }
}
```

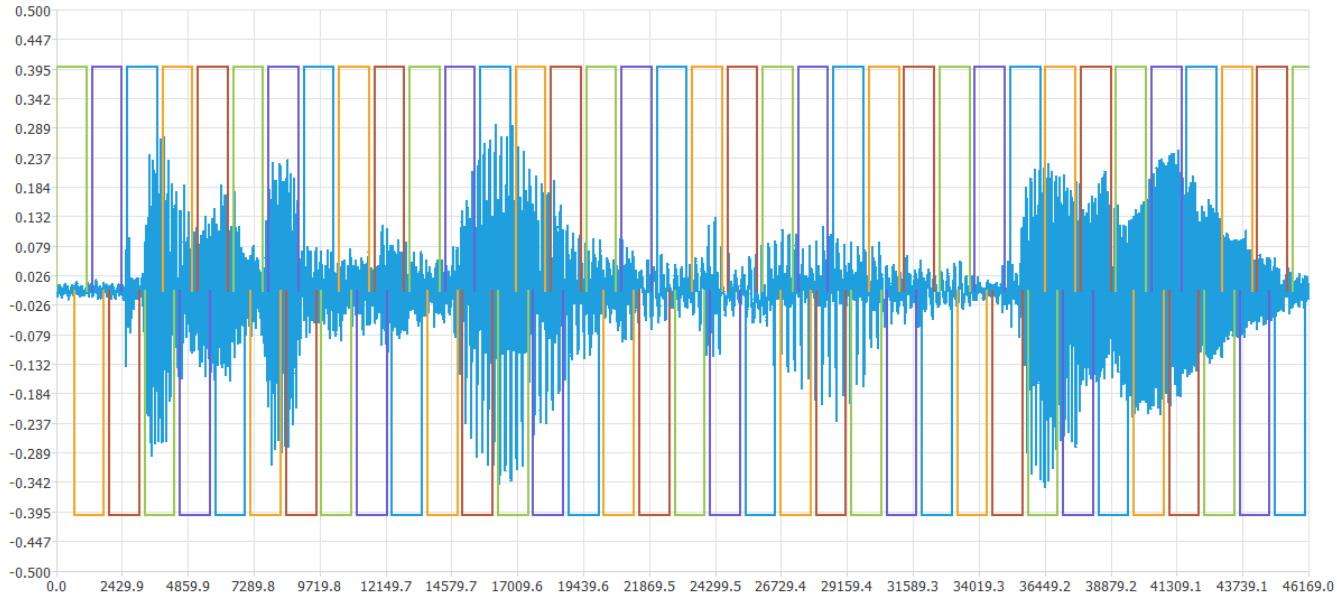
W powyższej funkcji, najpierw pobierana jest ilość próbek przypadających na jedną sekundę, oraz na 25 milisekundową ramkę. Następnie wartości kolejno z każdego obiektu typu QAudioBuffer, znajdującego się w wektorze audioBuffers, są dodawane do wektora wholeBuffer. Jest to wektor przechowujący zmienne zmiennoprzecinkowe, o podwójnej precyzji, tj.double.

4.4 Ekstrakcja tonu podstawowego

W pierwotnym założeniu program, poza estymacją częstotliwości podstawowej miał również dokonywać ekstrakcji niskopoziomowych cech. W toku implementacji zostały one jednak pominięte, z powodu posiadania małego wpływu na cel pracy. Pierwszym zagadnieniem, które powinno być rozważone, jest długość fragmentów sygnału, na które powinien być podzielony. Sygnały mowy nie są sygnalami stacjonarnymi, co oznacza, że ich częstotliwość istotnie zmienia się w czasie, znacznie obniżając dokładność obliczeń, opierających się na rezultatach transformaty Fouriera. W przetwarzaniu mowy korzystne jest dzielenie sygnału na części, celem uzyskania fragmentów sygnału bliskich byciu stacjonarnymi. Głośnia, odpowiedzialna za zmiany częstotliwości głosu, nie zamyka i nie otwiera się natychmiastowo, co oznacza, że w małych odstępach czasu wartości częstotliwości są do siebie zbliżone. Odpowiednio dzieląc sygnał możliwe jest uzyskanie krótszych quasi-stacjonarnych fragmentów. Proces ten nazywa się ramkowaniem.

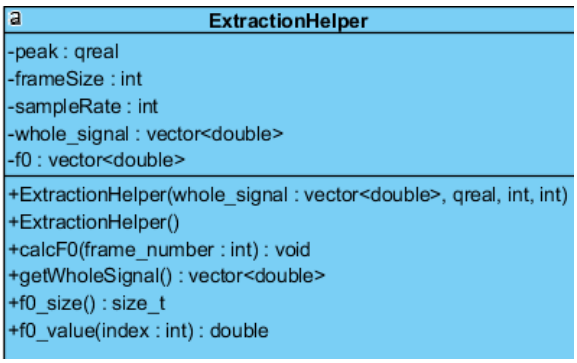
4.4.1 Ramkowanie oraz ekstracja wartości F0

Sygnał najczęśniej dzielony jest na 20-50ms ramki. W tym projekcie ustalona długość ramki wynosi 25ms. Oznacza to, że każda ramka składa się z 1102 wartości. Pojawia się jednak problem związany z wartościami brzegowymi. Dzieląc sygnał na przystające do siebie, lecz nie zachodzące na siebie ramki istnieje duże ryzyko nie wykrycia pewnych cech, które mogą znajdować się pomiędzy dwoma kolejnymi ramkami. Taka sytuacja mogłaby wystąpić podczas analizy sygnału w celu wykrycia konturów częstotliwości podstawowej. Jeżeli relatywnie krótki kontur zaczynałby się w jednej ramce i kończył w drugiej, mógłby nie zostać wykryty. Rozwiązaniem jest nakładanie ramek na siebie (overlapping).Określona część każdej ramki, zawarta jest również w ramce kolejnej. Najczęściej jest to 20-50% segmentu.



Rysunek 6: Zobrazowany podział sygnału na ramki wraz zastosowaniem 30-procentowego overlappingu. Opracowanie własne

Do ekstrakcji cech niskopoziomowych 30 procentowe nakładanie się ramek jest wystarczające. Jednak algorytm YIN, wykorzystany w projekcie do estymacji F0, wymaga znacznie większego zachodzenia fragmentów na siebie. W tym przypadku 90% danej ramki znajduje się również w ramce kolejnej. Oznacza to, że ramki przesuwane są jedynie o 2,5ms. Spowodowane jest to faktem, że algorytm YIN opiera swoje działanie na funkcji autokorelacji. Do ekstrakcji cech stworzona została klasa ExtractionHelper.



Rysunek 7: Klasa stworzona w celu ekstracji F0, obliczenia energii sygnału oraz przechowywania tych wartości

Listing 2: Przedstawienie sposobu dokonywania podziału na ramki, wraz z zastosowaniem overlappingu

```
void ExtractionHelper::calcF0(int numberOfFrames)
```

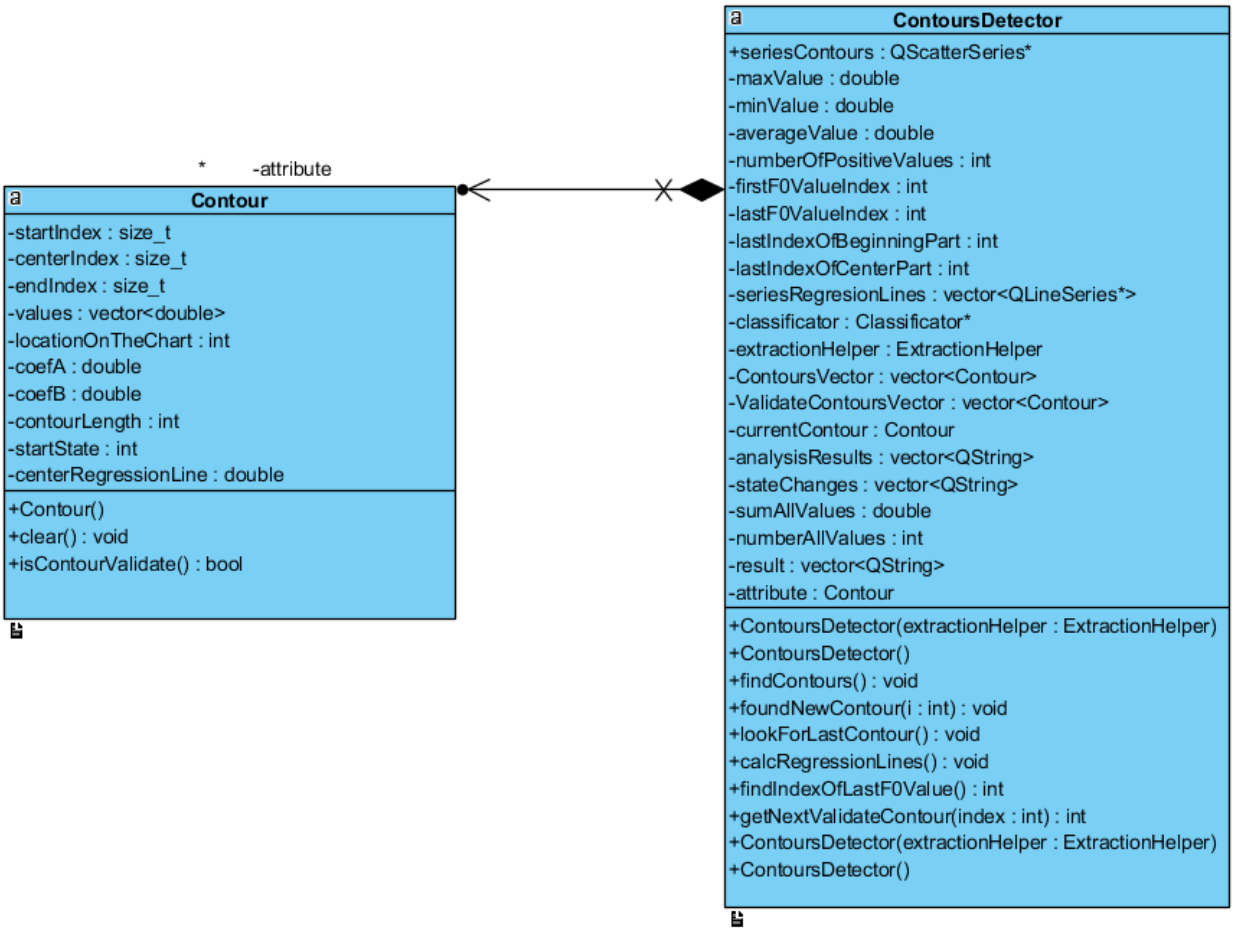
```
{

    int numberOfShifts=10;

    Yin m_yin(frameSize, sampleRate);


    int frameStartIndexAfterShifting = 0;
    int shift= frameSize/numberOfShifts;

    while(frameStartIndexAfterShifting < (whole_signal.size()))
    {
        double *shift_frame =new double [frameSize];
        int index=0;
        frameStartIndexAfterShifting +=shift;
        for(int k=frameStartIndexAfterShifting;
                k<frameStartIndexAfterShifting+frameSize;k++)
        {
            if(k>=whole_signal.size())
                shift_frame[index] = 0;
            else
                shift_frame[index] = whole_signal.at(k);
            index++;
        }
        Yin::YinOutput f0_struct=m_yin.process(shift_frame);
        if (f0_struct.f0 <F0_MAX && f0_struct.f0 >F0_MIN)
            f0.emplace_back(f0_struct.f0);
        else
            f0.emplace_back(0);
        delete shift_frame;
    }

}
```

W funkcji wykorzystywana jest klasa Yin, pochodząca z ogólnodostępnej implementacji algorytmu YIN. Konstruktor obiektu tej klasy jako argumenty przyjmuje długość pojedynczej ramki oraz częstotliwośc próbkowania. W tym przypadku wartości te wynoszą kolejno 1102 i 44100. W ciele funkcji calcF0 obiekt ten będzie wykorzystywany do

estymacji konturów F0 dla pojedynczych ramek. Z racji zastosowania wysokiego overlappingu, proces dzielenia sygnału na fragmenty nie wygląda jak typowe ramkowanie. Okno sygnału przeznaczone do estymacji będzie przesuwane jedynie o 2,5ms. W tym celu zadeklarowane zostały dwie zmienne, frameStartIndexAfterShifting przechowuje początkowy indeks obecnie przetwarzanej ramki, a zmienna shift przechowuje wartość pojedynczego przesunięcia. Warunkiem kończącym działanie głównej pętli funkcji jest przekroczenie przez początkowy indeks ramki rozmiaru całego sygnału. Oznacza to, że końcowa ramka może być dowolnie mała. W wewnętrznej pętli wartości rozpatrywanej ramki są przypisywane do dynamicznie zadeklarowanej tablicy. Jeżeli indeks tej pętli przekroczy rozmiar całego sygnału, reszta pól tablicy wypełniona jest zerami. Powodem tego jest wymaganie implementacji algorytmu YIN, aby wszystkie ramki miały jednakowy rozmiar. Po zakończeniu estymacji wartości F0 dla danej ramki, wartość ta jest dodawana do wektoru jeżeli mieści się w zdefiniowanym zakresie. Musi być większa niż 60 i mniejsza niż 450. W przeciwnym razie do wektoru zostanie dodana wartość zerowa. Po obliczeniach zadeklarowana dla ramki pamięć zostaje zwolniona.

4.5 Wykrywanie konturów

Wszystkie wyestymowane wartości częstotliwości podstawowej na tą chwilę przechowywane są w jednym wektorze. Aby umożliwić analizę przebiegu intonacji, konieczne jest wydzielenie poszczególnych konturów. Segmentacji można dokonać analizując wartości pod kątem wartości odstających. Do tego celu zostały stworzone dwie klasy.



Rysunek 8: Klasy stworzone do wykrycia poszczególnych konturów intonacyjnych, na podstawie wszystkich wartości F0

Dla każdej ze zmiennych istnieją funkcje typu get i set, odpowiednio zwracające wartość zmiennej oraz przypisujące dana wartość. Zostały one pominięte w celu zwiększenia czytelności diagramów. Główna funkcjonalność zawarta jest w funkcji findContours() w klasie ContoursDetector. Wykryte kontury będą umieszczane jako obiekty typu Contour, w wektorze contoursVector. W wektorze tym będą również umieszczane fragmenty z wartościami zerowymi, dla których nie wykryto występowania intonacji. Będą one pomijane w dalszej analizie, dodawane są w celu ułatwienia przejrzystego wyświetlania konturów na wykresie, w miejscu w którym rzeczywiście się znajdują.

Listing 3: Początkowa faza funkcji wykrywającej kontury

```
#define TRANSITION 15

void ContoursDetector::findContours()
```

```
{
    currentContour.setStart(1);
    lastValueIndex = findIndexOfLastF0Value();
    for(size_t i=1;i<extractionHelper.f0_size();i++)
    {
        double value =extractionHelper.f0_value(i);
        double previousValue = extractionHelper.f0_value(i-1);
        seriesContours->append(i,value);
        if (value > maxValue) maxValue = value;
        if (value < minValue && value > F0_MIN) minValue = value;
        if(std::abs(value - previousValue) > TRANSITION)
        {
            currentContour.setEnd(i-1);
            currentContour.setCenter();
            foundNewContour();
            currentContour.setStart(i);
            currentContour.addValue(value);
        }
        else
        {
            currentContour.addValue(value);
        }
    }
```

Początkowy indeks pierwszego konturu jest ustawiony jako 1. Główna pętla przebiega po wszystkich wyestymowanych wartościach tonu podstawowego. Oprócz poszukiwania konturów, wartości są również sprawdzane pod kątem wykrycia wartości maksymalnej i minimalnej. Funkcja wykrywanie danego konturu za zakończone, gdy aktualnie rozpatrywana wartość rózni się od poprzedniej o 15 jednostek. Metodą obserwacji ustalono taki przeskok za wystarczający do stwierdzenia, że dana wartość należy już do nowego konturu. Poprzedzający indeks jest uznawany za koniec danego konturu. Aktualny licznik pętli zostaje przekazany do funkcji foundNewContour. Z uwagi na obszerność tej funkcji, będzie ona omawiana fragmentami.

4.5.1 Analiza wstępna wykrytego konturu

Listing 4: Funkcja zajmująca się analizą wstępną wykrytego konturu

```
void ContoursDetector::foundNewContour()
```

```
{
    if (!currentContour.isContourValidate())
    {
        currentContour.clear();
        return;
    }

    if (lastIndexOfFirstPart == 0)
    {
        firstValueIndex = currentContour.getStartIndex();
        double occurenceRange = lastValueIndex-firstValueIndex;
        lastIndexOfFirstPart = currentContour.getStartIndex()
                                            +occurenceRange/4;
        lastIndexOfCenterPart = lastValueIndex - occurenceRange/4;
    }
```

Najpierw kontur jest poddawany walidacji. Sprawdzane jest, czy nie występują w nim wartości zerowe oraz czy jego długość jest większa niż 1. Przyjęta implementacja segmentacji traktuje wartości zerowe jako przerwy między konturami i nie powinny one być dodawane do wektoru przechowującego wykryte kontury. Do określania czy dany obiekt jest przerwą między konturami, wystarczy sprawdzić jego pierwszą wartość. Metodą obserwacji zauważono, że kontury składające się tylko z jednej wartości, często są błędami estymacji, lub powstają w wyniku róznego rodzaju zanieczyszczeń w nagraniu. Mogą zaburzać wyniki późniejszej klasyfikacji, dlatego są pomijane.

Listing 5: Funkcja dokonująca walidacji konturu

```
bool isContourValidate()
{
    if (values.size()<2) return false;
    if (values.at(0) == 0)  return false;
    return true;
}
```

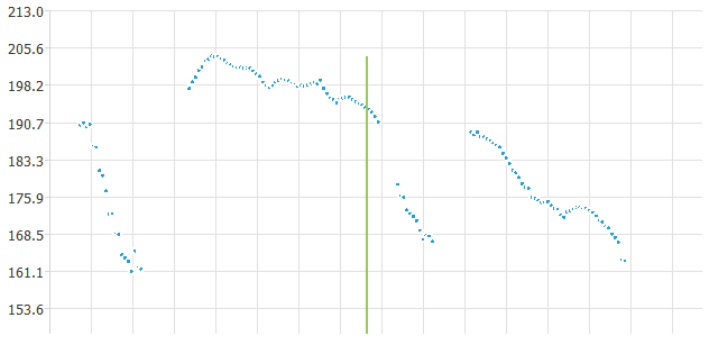
Późniejsza analiza konturów w celu wykrycia rodzaju danego zdania, oparta jest w dużej mierze na położeniu danego konturu w przestrzeni przebiegu całej intonacji. Przebieg intonacji zawiera wartości od pierwszego poprawnego konturu do ostatniego. Wykres jest dzielony na 3 części. Na część początkową oraz końcową przypada po 25% całości, podczas gdy część środkowa zawiera pozostałą połowę. W celu przydzielenia konturom odpowiedniej lokalizacji, używane są zmienne typu całkowitego, lastIndexOfFirstPart

oraz lastIndexOfCenterPart. Wyznaczają one końce początkowej oraz środkowej części.

Listing 6: Dalsza część funkcji foundNewContour

```
 if (ContoursVector.size()>0)
{
    if ((currentContour.getFirstValue()
            -ContoursVector.back().getLastValue())
            >(currentContour.getFirstValue()/6))
    {
        currentContour.setStartState(GROWTH);
    }
    else if ((ContoursVector.back().getLastValue()
            - currentContour.getFirstValue())
             >(ContoursVector.back().getLastValue()/4))
    {
        currentContour.setStartState(DROP);
    }
}
ContoursVector.push_back(currentContour);
currentContour.clear();
}
```

W dalszej części kodu funkcji foundNewContour, dokonywana jest analiza położenia danego konturu względem poprzednika. Jeżeli początkowa wartość analizowanego konturu jest znacząco mniejsza lub większa od ostatniej wartości konturu poprzedzającego, zapisywana jest informacja o gwałtownym przeskoku w przebiegu intonacji.



Rysunek 9: Przykładowy przeskok(wzrost) między pierwszym i drugim konturem, zlokalizowanymi w początkowej części

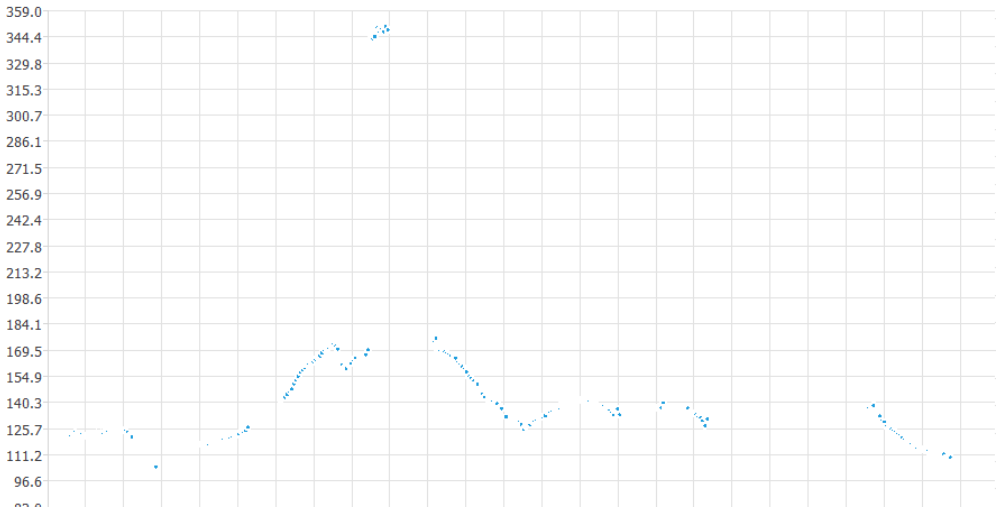
Następnie kontur zostaje dodany do wektoru, zmienna currentContour zostaje wyczyszczona w celu poszukiwania kolejnego konturu. Na tym funkcja foundNewContour

kończy swoje działanie.

Listing 7: Dalsza część głównej funkcji findContours

```
    double averageWithoutCurrentContour;
    for(int i = 0;i<ContoursVector.size();)
    {
        averageWithoutCurrentContour = sumAllValues -
                        ContoursVector.at(i).getCenterOfRegressionLine();
        averageWithoutCurrentContour /= (ContoursVector.size()-1);
        if((ContoursVector.at(i).getCenterValue()
                > (averageWithoutCurrentContour*1.6))
                && (ContoursVector.at(i).getContourLength()<10))
        {
            ContoursVector.erase(ContoursVector.begin()+i);
        }
        else
        {
            setContourLocation(i);
            i++;
        }
    }
    calcRegressionLines();
}
```

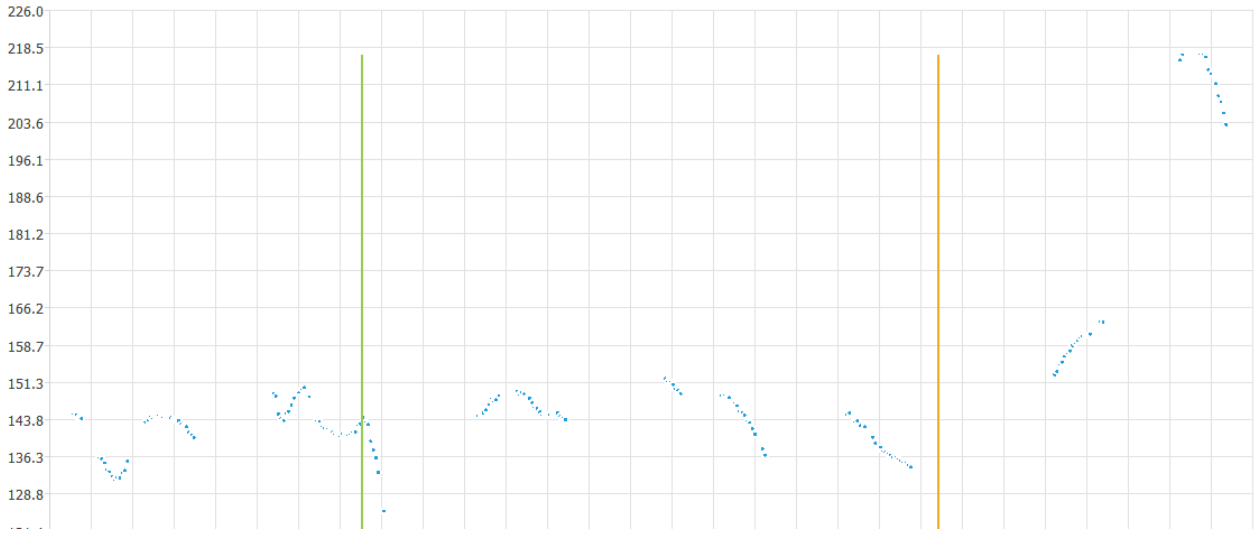
W czasie implementacji wykrywania konturów oraz przy późniejszej analizie, zauważano występowanie krótkich, wyraźnie odstających konturów. Pojawiały się w miejscach, w których nie było logicznego uzasadnienia ich występowania. Miały wyraźny wpływ na zaburzenia procesu wykrywania rodzaju zdania. Podjęto decyzję o usuwaniu ze zbioru takie kontury, których wartości są bardzo wyraźnie większe od średniej oraz jednocześnie są bardzo krótkie. Pierwotnie zostało to zaimplemetowane w celach testowych, lecz okazało się, że zabieg ten znacząco poprawia stopień poprawnego rozpoznawania.



Rysunek 10: Przykład usuniętego konturu.

Na rysunku 14 przedstawiony został przykład usuniętego konturu. Jest to kontur, którego wartości oscylują około 345 jednostek. Jest to liczba ponad dwukrotnie większa od innych konturów, do tego kontur ten jest bardzo krótki. Słuchając nagrania, nie sposób było uzasadnić jego występowanie w tym miejscu, dlatego został uznany za błąd estymacji i usunięty ze zbioru. Jeżeli warunek usunięcia konturu nie jest spełniony, wywoływana jest funkcja określająca jego położenie w przebiegu intonacji. Jak zostało już wspomniane, przebieg intonacji jest podzielony na 3 części. W zależności od położenia środka konturu, zostaje mu przypisane odpowiednie makro, zawierające informację o lokalizacji konturu.

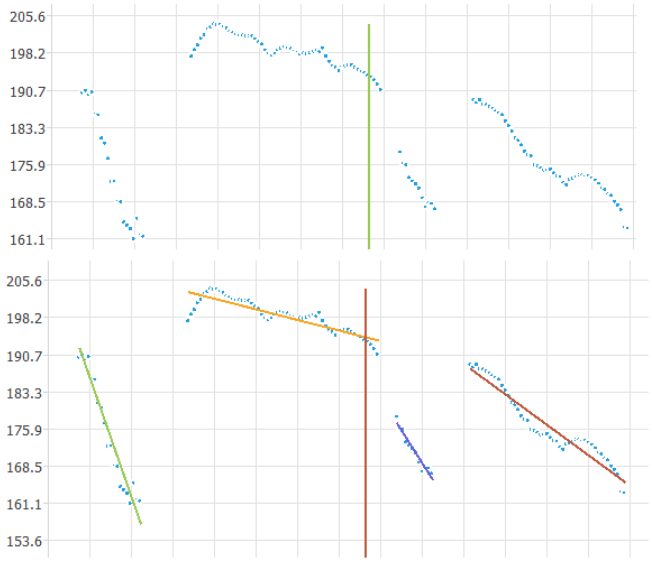
```
void ContoursDetector::setContourLocation(int i)
{
    if (ContoursVector.at(i).getCenter() < lastIndexOfFirstPart)
        ContoursVector.at(i).setLocation(BEGINNING);
    else if (ContoursVector.at(i).getCenter() < lastIndexOfCenterPart)
        ContoursVector.at(i).setLocation(CENTER);
    else
        ContoursVector.at(i).setLocation(END);
}
```



Rysunek 11: Przykład podziału przebiegu intonacji na 3 części

4.5.2 Współczynniki regresji liniowej

Dla każdego wykrytego konturu obliczane są współczynniki regresji liniowej. W tym celu została zaimplementowana metoda najmniejszych kwadratów, szerzej opisana we wstępie teoretycznym. Kod tej funkcji nie został umieszczony w pracy, z uwagi na jego obszerność oraz fakt, ze jest to implementacja gotowego wzoru. Wewnątrz funkcji, każdemu konturowi zostają przypisane wartości obliczonych współczynników A i B oraz obiekt typu QLineSeries. Obiekt ten, bazując na obliczonych współczynnikach, słuzy do zobrazowania na wykresie przebiegu linii regresji dla danego konturu.



Rysunek 12: Fragment przebiegu intonacji przed i po nałożeniu linii regresji

5 Analiza wykrytych konturów