پایان نامه

بنام خدا

تشخیص صوت و گفتار

شامل مباحث

- درک عملکرد دستگاه شنوایی و تکلم (Vocal Tract)
- تشخیص اصوات هارمونیک (هنجار)
- تشخیص زیری و بمی صدای انسان
- تشخیص حروف
- قطعه بندی کلمات (Segmentation)
- شرح پیاده سازی کامل پروژه
- همراه با سورس کامل پروژه و برنامه قابل اجراء

تهیه کننده : سید حسن حسینی قمی نژاد

استاد راهنما : مهندس محمد تقی خیر آبادی

بهار ۱۳۹۰

بسم الله الرّحمن الرّحيم

قدردانی و تشکر

این پروژه بدون شک بدون کمک و الطاف پروردگار و کلیه کسانی که در این راستا بنده را یاری نمودند قابل انجام نبود. از جناب آقای مهندس خیر آبادی در سمت استاد راهنما که همواره راهنمای بنده بودند تشکری بسیار ویژه دارم و همچنین از دست اندر کاران گروه کامپیوتر که طی این سال‌ها بنده را تحمل نمودند بسیار متشکرم.

تقدیم به مادر عزیزم

فهرست

مقدمه

امروزه پردازش و تشخیص زبان طبیعی در جهان از اهمیت ویژه‌ای برخوردار است به طوری که اکثر دانشگاه‌های جهان در نو آوری و کشف روش‌های تشخیص جدید به فعالیت مشغولند و همه روزه شاهد پیشرفت چشم گیر در این علم هستیم. هر چند در دانشگاه‌های معتبر کشور عزیزمان نیز این فعالیت انجام شده ولی محصولات پردازش زبان طبیعی زبان فارسی در بازار به ندرت یافت می‌شود بدون اغراق در این زمینه ده‌ها سال از دیگر کشورهای صنعتی عقب هستیم. در صورتی که برای حتی پیاده سازی نیازی به ادوات الکترونیکی تکنولوژی بالا نداریم. تنها چیزی که نیاز است شناخت پایه برای ورود به این مباحث و الگوریتم‌های کارا برای پردازش زبان طبیعی است. به نظر من یکی از دلایلی که ممکن است وجود داشته باشد این است که اکثر دانشجویان رشته هوش مصنوعی را یا به خاطر قشنگی نام آن و یا به خاطر قبولی آسان در این رشته انتخاب نمودند. غافل از حد اقل پیش‌نیازهایی که در آن‌ها وجود ندارد وارد این رشته شده و تمام هدفشان پس از ترم اول دریافت مدرک تحصیلی می‌باشد. فکر می‌کنم اگر دانشجویان با مفاهیم بسیار پایه هوش مصنوعی، ریاضیات و آمار احتمالات به صورت کاربردی در مقطع کارشناسی آشنایی داشته باشند و نیاز واقعی این علوم را احساس کنند پیشرفت این علوم در کشور ما نیز غیر قابل تصور خواهد بود. گرچه در این پایان نامه هیچ نو آوری دیده نمی‌شود ولی فکر می‌کنم برای باز شدن ذهن دانشجویان در زمینه پردازش زبان طبیعی انشالله موثر خواهد بود.

این پایان نامه شامل مطالبی جهت شناخت اصطلاحات و اطلاعات پیش نیاز اصوات و روش‌های دریافت صوت و همچنین پردازش و تشخیص آن می‌باشد. همچنین پیاده سازی پروژه نیز در بستر دات نت بدون استفاده از هیچ گونه ابزار جانبی انجام شده است. تا جای ممکن سعی شده تا مطالب با بیانی ساده آماده شوند و از همه مهم‌تر از عنوان مطالب ریاضی و آمار اجتناب شده است. ولی دانشجویان به خالی بودن جایگاه آمار احتمالات و هوش مصنوعی پی خواهند برد و امید آنکه با کمک اساتید و دانشجویان، زمینه‌های ورود به علوم ریاضیات و آمار همراه با عمل برای دانشجویان فنی فراهم گردد و انشالله این پایان نامه نیز نقشی هر چند بسیار کوچک در تحقق این امر داشته باشد.

دوستان در صورت داشتن هرگونه ابهام، اصلاحات، پیشنهادات و یا در صورت مقدور به اشتراک گذاری دانش و دست آورد های جدید می‌توانند با پست الکترونیکی بنده به نشانی زیر ارتباط برقرار نمایند :

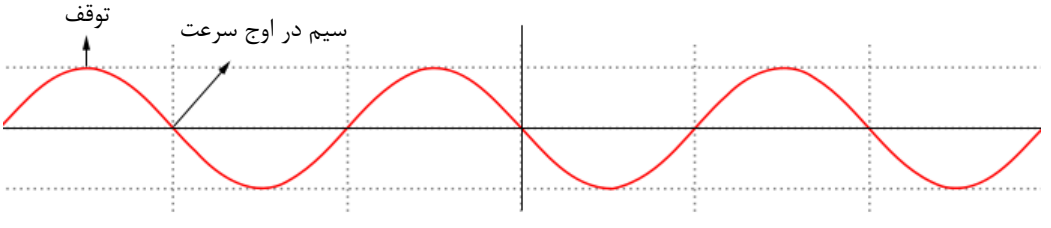
ghominejad@gmail.com

۱ شناخت صدا

صداهایی که ما می‌شنویم

صداهایی که ما می‌شنویم تغییرات سریع فشار هوا است که می‌تواند به روش‌های گوناگون ایجاد شود. برای مثال به وسیله تکان خوردن یک شی فلزی یا بال‌های یک حشره و یا صدایی که از دهان خارج می‌کنیم. صدا حدوداً با سرعت ۳۳۵ متر بر ثانیه در هوا انتقال پیدا می‌کند.

وقتی سیم یک نوع دستگاه موسیقی را می‌کشید و رها می‌کنید حرکت رفت و برگشتی در سیم ایجاد شده و سیم از طریق فشاری که بر هوا وارد می‌آورد صدایی از خود ایجاد می‌نماید. در ذهن خود تصور کنید وقتی که این سیم در حالت رفت به انتها می‌رسد لحظه‌ای مکث نموده و باز می‌گردد و در بازگشت مجدداً لحظه‌ای مکث نموده دوباره سرعت می‌گیرد. اگر بخواهیم این رفت و برگشت را در نمودار به تصویر بکشیم شبیه یک نمودار سینوسی می‌شود که به این حالت، حالت موج گون می‌گوییم.

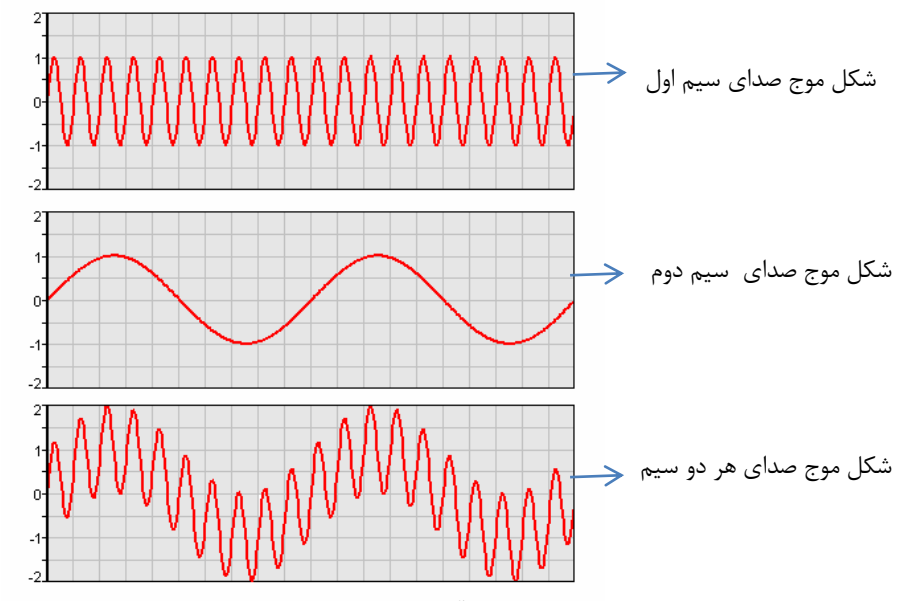


شکل ۱.۱ حرکت رفت و برگشت سینوسی سیم دستگاه موسیقی

این صدا آنقدر ادامه میابد تا رفت و برگشت ساکن شود. در این بین بلندی صدا نیز کاهش میابد. هر چقدر فاصله رفت و برگشت بیشتر باشد صدا قوی‌تر و هرچقدر فاصله کمتری وجود داشته باشد صدا ضعیف‌تر به گوش میرید. همچنین هر چقدر سیم سفت‌تر باشد حرکت رفت و برگشت در آن سریع‌تر انجام شده و صدایی زیرتر تولید می‌کند. به تعداد رفت و برگشت‌ها در ثانیه **فرکانس** میگوییم که با واحد هرتز اندازه گیری می‌شود.

به طور مثال اگر سیم را طوری فشار دهید که ۱۰۰۰ بار در ثانیه رفت و برگشت ایجاد کند میگوییم فرکانس آن ۱۰۰۰ هرتز است. این موج‌های ساده سینوسی معمولاً از حدود ۲۰ تا ۱۶۰۰۰ هرتز قابل درک می‌باشند. که تون نامیده می‌شوند.

موج‌ها می‌توانند فراتر از حالت تون باشند به طور مثال موج ترکیبی زیر را ببینید این موج کمی متفاوت‌تر از موج تون ساده می‌باشد. این موج از دو موج تون ترکیب شده است. مثل اینکه دو سیم را با هم به صدا درآوریم. اگر نوار حساسی را روبروی دو سیم قرار دهید حرکت رفت و برگشت آن را به تصویر بکشیم دیگر شبیه رفت و برگشت سینوسی ساده نیست. سیم اول یک نوع موج ساده به نوار تحمیل نموده و سیم دوم موجی با سرعت متفاوت (فرکانس متفاوت) تحمیل می‌کند. در حقیقت شکل موج روی نوار جمع موج‌های دو سیم خواهد بود.



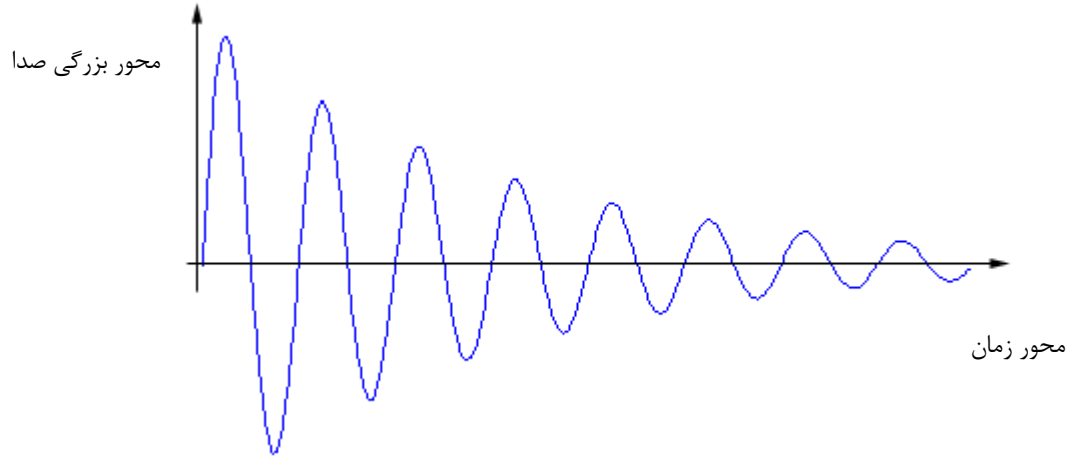
شکل ۱.۲ موج‌های مختلف با هم ترکیب شده و فشار هوای ترکیبی آن‌ها موجی متفاوت ایجاد می‌نماید.

در مثال قبل برای درک آسان‌تر فرض کردیم با به ارتعاش در آوردن سیم یک دستگاه موسیقی صدای آن یک موج سینوسی ساده با یک فرکانس تولید می‌کند. در حقیقت صداهایی که از طریق سیم دستگاه‌های موسیقی به صدا در می‌آیند تنها شامل یک موج تون نمی‌باشند. بلکه بسته به جنس ساز، نوع سیم، کیفیت ساخت، با فشردن یک سیم یک فرکانس پایه (Fundamental) همراه با چند فرکانس اصلی به صورت ترکیبی (Formant) طنینی ایجاد نموده که سبب متفاوت شدن صدای یک دستگاه موسیقی با دیگری می‌گردد. در تصویری که مشاهده نمودیم می‌توانیم فرکانس تصویر را فرکانس پایه دستگاه موسیقی تصور کنیم.

جهت ساده‌تر شدن مطلب جا بجایی سیم را موجب تشکیل صدا فرض نمودیم ولی در حقیقت جا بجایی سیم موجب تغییر فشار هوا شده و سبب حرکت هوا به جلو می‌شود که جای خالی هوا را مجدداً هوا پر می‌کند.

ترسیم صدا

یک صوت شامل فرکانس‌های مختلفی است که ممکن است بلندی (بزرگی) صدای آن و یا فرکانس‌های آن (آمد و رفت) در طول زمان تغییر کند. به راحتی می‌توان اصوات را در نمودارهایی که x آن زمان و y آن بزرگی صدا باشد به تصویر کشید.

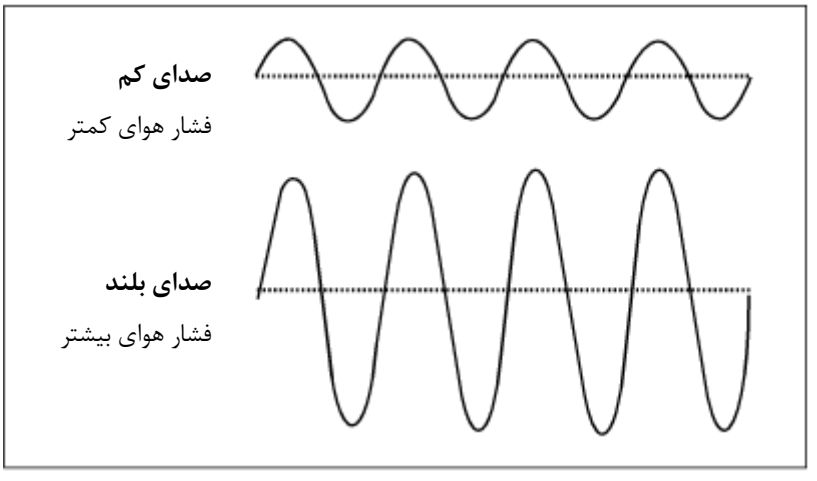


شکل ۱.۳ تصویر موجی را در نمودار دامنه زمان نشان می‌دهد که با گذشت زمان شدت صدای آن کم می‌شود در صورتی که فرکانس آن ثابت است.

این سبک نمایش با نام نمایش صوت در دامنه‌ی زمان معروف است که به راحتی می‌توانیم تغییرات را در طول زمان مشاهده کنیم.

بلندی صدا

صداهای بلند فشار هوای زیاد و صداهای کوتاه فشار هوای کمتری تولید می‌کنند. موج‌های صوتی با ارتعاش یا شدت بزرگ‌تری در فشار هوا صدای بلند تری تولید می‌کنند. در شکل زیر نمونه‌ای از صدای نسبتاً بلند و نسبتاً کوتاه را مشاهده می‌کنید.

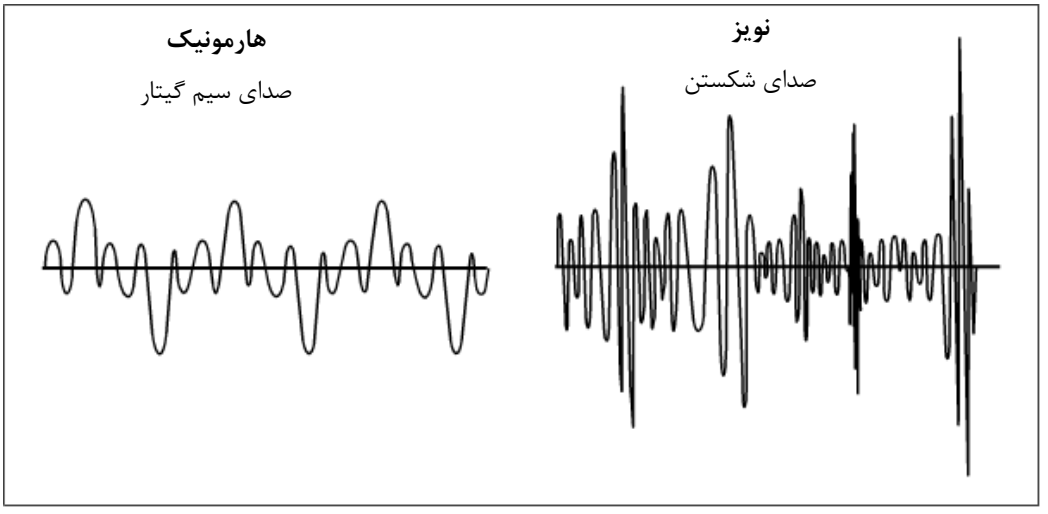


شکل ۱.۴ مقایسه دو موج سینوسی صدای بلند و کوتاه در نمودار دامنه زمانی

بلندی صدا را با بزرگی صدا اشتباه نگیرید. بلندی صدا مربوط به شنوایی انسان می‌باشد و کاملاً نسبی است ولی بزرگی مربوط به اندازه گیری فشار هوای موجود می‌باشد که معمولاً با dB بیان می‌شود. در صورتی که بنده نیز در این پایان نامه از واژه بلندی استفاده نمودم، در خیلی از نقاط منظور همان بزرگی است.

صداهای هنجار و نا هنجار

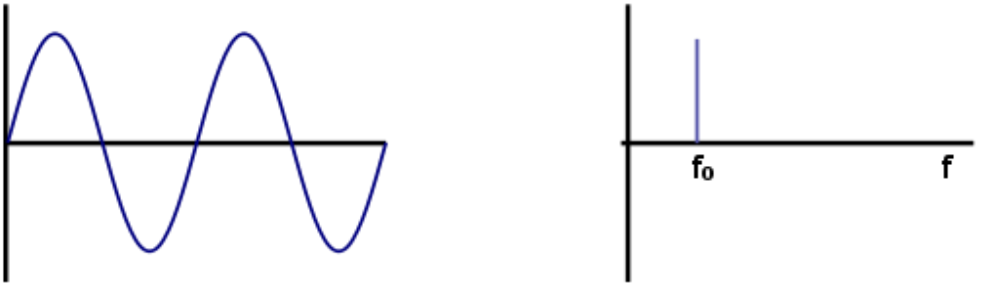
امواجی که رفت و برگشت منظمی در طول زمان داشته باشند صدایی موسیقیایی یا هارمونیک تولید می‌کنند. در طبیعت موج‌هایی وجود دارند که هیچ دوره منظمی در آن نمی‌توان یافت که به این موج‌ها نویز میگوییم. این صداها برای انسان غیر موسیقیایی و بد صدا می‌باشند .



شکل ۱.۵ موج‌های هارمونیک با گذشت زمان الگویی تکرار شونده و منظم دارند ولی نویزها کاملاً تصادفی می‌باشند

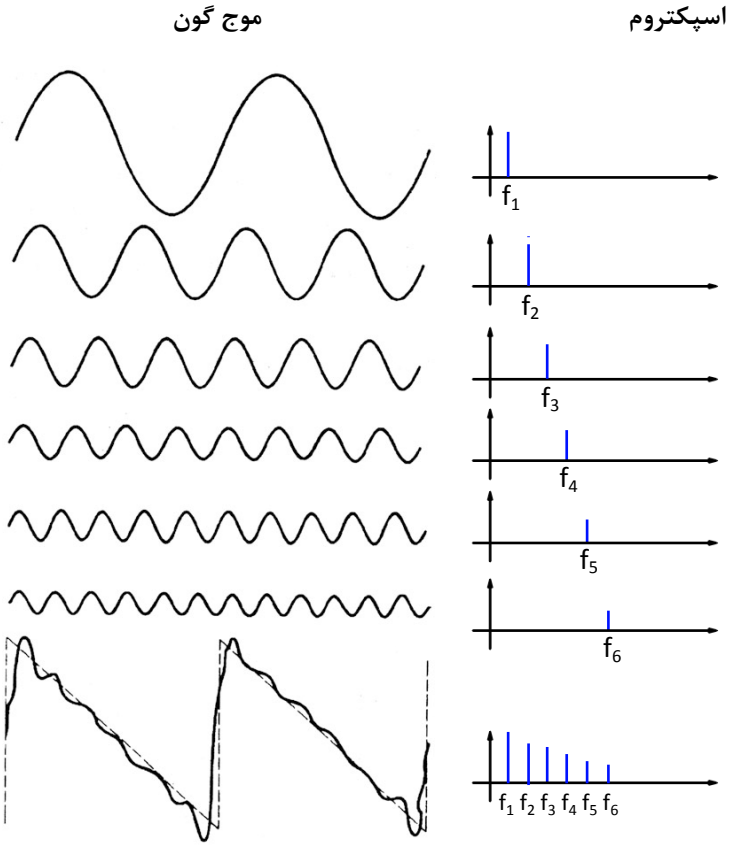
ترسیم صدا در دامنه فرکانس (نمودار اسپکتروم)

نوعی دیگری از نمایش با نام نمایش در دامنه فرکانس نیز بسیار مرسوم می‌باشد. در این نوع نمایش به راحتی می‌فهمیم که یک صوت از چه فرکانس‌هایی تشکیل یافته است. برای مثال در شکل ۱.۵ به راحتی می‌توان فهمید که صدای مورد نظر از یک فرکانس f0 و بلندی صدای آن که با رنگ آبی مشخص شده تشکیل شده است. سمت چپ نمودار دامنه زمان و سمت راست نمودار دامنه فرکانس آن می‌باشد. این نمودار به نمودار اسپکتروم نیز معروف می‌باشد.



شکل ۱.۶ شکل سمت چپ نمودار زمانی یک موج سینوسی می‌باشد که در نمودار سمت راست اندازه فرکانس آن (f_0) همراه با نمایش بزرگی فرکانس(خط آبی) در قالب نمودار دامنه فرکانس به نمایش در آمده است.

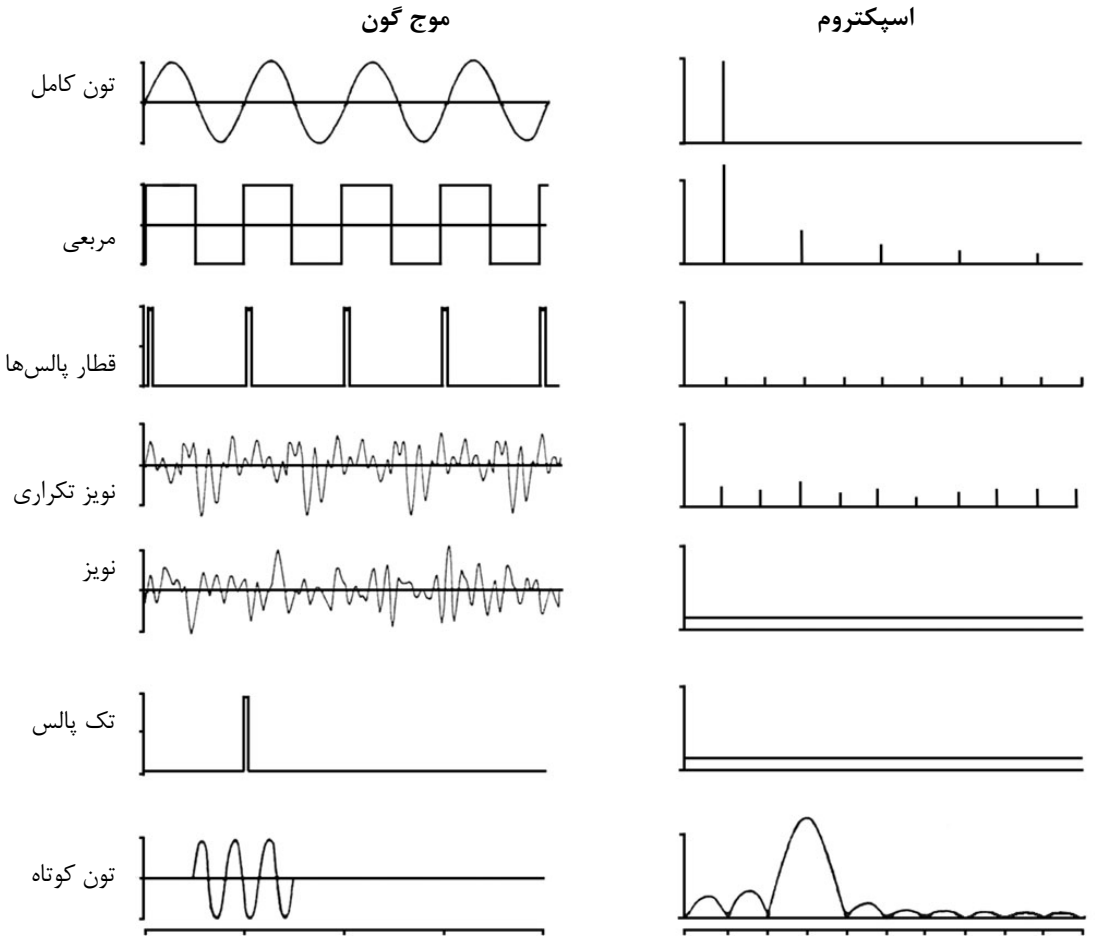
این سبک از نمایش برای تحلیل گران صوت بسیار کارآمد می‌باشد و ما نیز در بخش‌های بعدی به کرات از این سبک نمایش استفاده خواهیم کرد. همچنین با اطلاعاتی که در نمودار دامنه‌ی فرکانس داریم می‌توانیم به نمودار دامنه‌ی زمانی برسیم. فقط کافی است فرکانس‌های موجود در دامنه‌ی فرکانس را با توجه به بلندی هر یک از فرکانس‌ها را به صورت مستقل ترسیم نموده و مجموع آن‌ها در طول زمان نمودار دامنه زمانی خواهد بود.



شکل ۱.۷ نمودار دامنه زمان و دامنه فرکانس شش موج با فرکانس‌های f_1 تا f_6 در تصویر قرار گرفته شده و در انتها یک موج ترکیب شده از این شش تون ترسیم شده است که به دندان اره‌ای معروف است (sawtooth) و در نمودار دامنه فرکانس آن تمام فرکانس‌های تشکیل دهنده آن کاملاً مشخص می‌باشد.

انواع موج‌ها

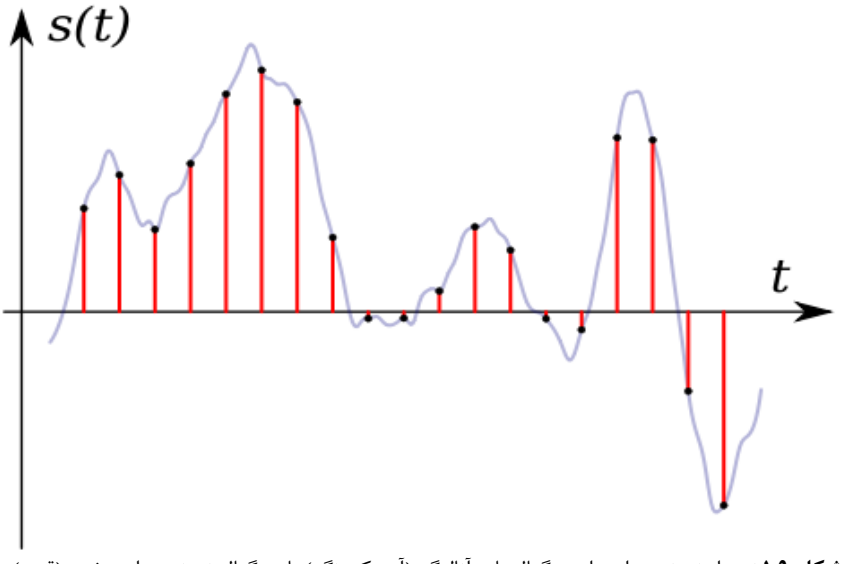
موج ها فقط محدود به شکل سینوسی و نویز نمی باشند، در قسمت گذشته با سبک نمایش تون (یک نمودار سینوسی ساده آسنا شدیم). بجز موج تون ساده موج های دیگری نیز موجود می باشند که تصویر نمودار اسپکتروم آنها متفاوت می باشد. در تصویر ۱.۸ انواع موج‌ها همراه با شکل نمودار اسپکتروم آن‌ها را مشاهده نمایید.



شکل ۱.۸ سمت راست الگوهای موج های متفاوت در دامنه زمانی می باشد که نمودار اسپکتروم مربوطه در سمت چپ آن به نمایش در آمده است.

نمونه برداری صدا

اصوات در دنیای واقعی به صورت پیوسته در جریان می‌باشند. ما قادریم این اصوات را از طریق کارت صدا و میکروفون توسط رایانه دریافت و پردازش نماییم. ولی کارت‌های صدا قادر نیستند که اصوات را به صورت پیوسته (آنالوگ) جهت پردازش در اختیارمان قرار دهند چرا که هم تعدادی نمونه‌های موجود در محیط نامحدود می‌باشد. رایانه‌ها اصوات موجود در محیط را فقط نمونه برداری می‌کنند. کارت‌های صدای قوی‌تر، قادرند نمونه‌های بیشتری از بلندی صدای محیط در یک ثانیه دریافت نمایند. امروزه اکثر کارت‌های صدا قادرند ۱۹۲،۰۰۰ نمونه در ثانیه از محیط دریافت نمایند. هرچقدر نمونه برداری بیشتر باشد ما قادریم اطلاعات بیشتری از جزئیات و ظرافت صدا دریافت نماییم و در نتیجه پردازش مان از کارایی بیشتری برخوردار خواهد بود. هر نمونه از بلندی چندین فرکانس مربوط به آن زمان تشکیل شده است.



شکل ۱.۹ نمودار نمونه برداری از سیگنال‌های آنالوگ (آبی کم‌رنگ) با سیگنال نمونه برداری شده (قرمز) با فواصل ثابت (نرخ نمونه برداری).

پردازش روی صدا

برای آنکه بتوانیم روی صدا پردازش انجام دهیم (تحلیل کنیم) نیاز داریم موج‌های موجود در آن را شناسایی نموده و نمودار دامنه فرکانس آن را ترسیم نماییم. برای این کار می‌توانیم از فرایندی که توسط **جوزف فوریر** در نیمه اول قرن نوزدهم توسعه یافت استفاده کنیم. کسی که توانست یک صوت دوره‌ای را در دامنه فرکانس به نمایش در آورد. و ما نیز برای تحلیل صوت از **تبدیل فوریه** استفاده خواهیم نمود.

ما حدودا عمل تبدیل را بیش از ۵۰ بار در ثانیه انجام می دهیم تا بتوانیم در هر ۲۰ میلی ثانیه صوت را تحلیل کنیم. برای این منظور الگوریتم بسیار کارامد تری مخصوص تبدیل توسط رایانه به نام تبدیل سریع فوریه (Fast Fourier Transfer) طراحی گردیده است. این الگوریتم آرایه‌ای از مقادیر نمونه‌های صوتی را دریافت نموده و پس از پردازش، آرایه‌ای از بلندی هر فرکانس آن را تحویل می‌دهد. در بخش‌های مربوط به پردازش صدا مفصلاً شرح داده شده است.

اندازه گیری شدت صوت

دامنه تغییرات بلندی صدا بسیار بزرگ است. به طوری که اگر یک صدا به بزرگی $۱۰^{۶}$ برابر سطح صدا از منبع صوتی باشد موجب ناراحتی می‌شود.

درک اصوات در مکان‌های مختلف متفاوت می‌باشد. تصور کنید که نوازنده‌ای آهنگ زیبایی را در محیطی آرام می‌نوازد. اگر همین آهنگ قرار باشد در محیطی پر سر و صدا نواخته شود برای اینکه با همان کیفیت شنیده شود، نوازنده باید انرژی بسیار بیشتر صرف کند تا بلندی صدای آهنگ به قدری باشد که سرو صدای محیط دیگر قابل تشخیص نباشد. اجازه دهید با یک سری اعداد و ارقام ساده مثالی بزنیم.

نوازنده‌ای در محیطی آرام آهنگی را به بلندی ۱۰۰۰ واحد می‌نوازد و قصد دارد همین آهنگ را در محیطی که شدت سرو صدای آن نیز ۱۰۰۰ واحد می‌باشد بنوازد. اگر قرار باشد که نوازنده آهنگ را با همان شدت ۱۰۰۰ واحد بنوازد این آهنگ درست شنیده نخواهد شد. او باید به ازی ۱ واحد سرو صدا ۱۰۰۰ واحد به بلندی آهنگ بیفزاید. در نتیجه به ازی ۱۰۰۰ واحد سرو صدا باید آهنگ را به بلندی ۱۰۰۰۰۰۰ واحد بنوازد.

به همین دلیل وقتی صحبت از اندازه گیری استاندارد می‌شود باید مقدار صوت را با ۲ صوت محاسبه کنیم. ۱- صوت اصلی (نوازنده) ۲- صوت فرعی (سر و صدا)

$$\text{اندازه} = \frac{\text{اندازه اصلی}}{\text{اندازه فرعی}}$$

واحد اندازه گیری صدا بل می‌باشد (جهت قدر دانی از الکساندر گراهان بل) و یک دهم بل دسی بل نامیده می‌شود و در اندازه گیری دسی بل (dB) بسیار مرسوم‌تر می‌باشد.

به دلیل اینکه اندازه دامنه صوتی بسیار بزرگ می‌باشد آن را به صورت لگاریتمی بیان می‌کنیم.

$$\text{بل } B = \log_{10} (I_1 / I_2)$$

$$dB = \frac{1}{10} \; Log_{10} \; (I_1/I_2)$$

در مثال قبل I_1 برابر $۱۰^{۶}$ (صدای آهنگ) و I_2 برابر 10^3 (صدای محیط) می‌باشد. در نتیجه

$$B = \text{Log}_{10} \frac{10^6}{10^3} = \text{Log}_{10}\ 10^3 = 3 \quad B = 0.3 \text{ dB}$$

همان‌طور که مشاهده نمودید با به‌کارگیری مقیاس dB اندازه‌های بسیار بزرگ صوتی را در قالب مقیاس‌های بسیار کوچک‌تر بیان می‌کنیم.

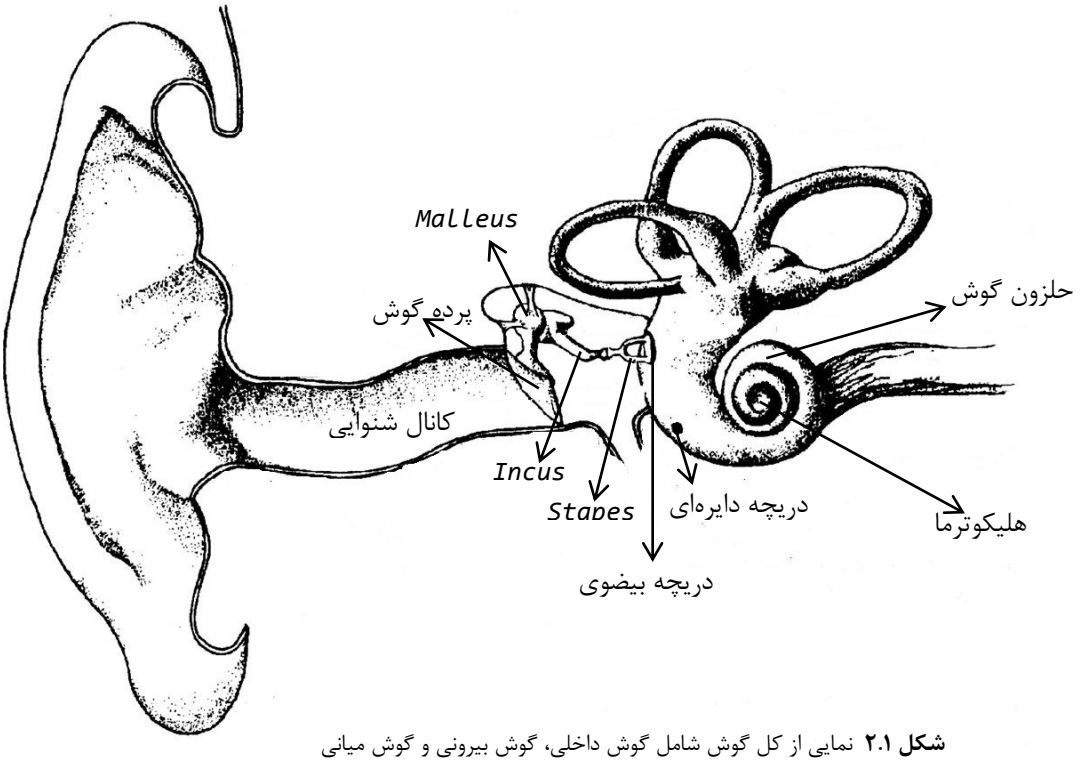
۲ دستگاه شنوایی و تکلم انسان

دستگاه شنوایی انسان

مرسوم است که دستگاه شنوایی انسان، به سه قسمت گوش بیرونی، گوش میانی و گوش داخلی تقسیم می‌شود که در ادامه به شرح هر یک خواهیم پرداخت.

ساختار گوش بیرونی و گوش میانی

شیارهای آکوستیکی که در گوش بیرونی تعبیه شده به ما کمک می‌کند تا فرکانس‌های بالا را به وسیله کنترل ماهیچه‌ها، بسیار با کیفیت‌تر دریافت نماییم.پس از نقل و انتقالات آکوستیکی که به صورت بازتابی توسط گوش بیرونی تولید شده . صدا از میان کانال شنوایی که به پرده گوش منتهی می‌شود عبور می‌کند. این کانال بیش از یک گذرگاه ساده می‌باشد. طول آن حدود ۲.۵ سانت ، چیزی شبیه یک لوله که صدا در آن پیچیده شده (*Resonant*) عمل می‌نماید. تأثیر این تشدید (*Resonance*) چیزی حدود ۵ دسی بل یا بیشتر، از بازه فرکانس حدود ۲۰۰۰ هرتز تا ۵۵۰۰ هرتز می‌باشد. که بیشترین تشدید چیزی حدود ۱۱ دسی بل بر روی محدوده فرکانس ۴۰۰۰ هرتز صورت می‌پذیرد. تغیرات فشار هوا در انتهای لوله سبب می‌شود تا پرده گوش به لرزش در آید. پرده گوش توسط سه استخوان به گوش میانی متصل می‌باشد.



شکل ۲.۱ نمایی از کل گوش شامل گوش داخلی، گوش بیرونی و گوش میانی

زنجیره‌ای از سه استخوان بسیار کوچک (*Incus, Malleus, Stapes*) رابط گوش داخلی و گوش خارجی می‌باشد. استخوان *Stapes* به دریچه بیضی شکل (*Oval Window*) متصل می‌باشد. این دریچه در ابتدای حلزون گوش که با مایع پر شده قرار گرفته است. از این دریچه به بعد (حلزون گوش) گوش داخلی را تشکیل می‌دهد. گوش میانی صدایی که در هوا موجود است را به داخل مایع موجود در گوش داخلی بدون از دست دادن کیفیت صدا هدایت می‌کند. وقتی که صدا از داخل هوا به طور مستقیم وارد مایع گوش می‌شود ۹۹.۹ درصد از قدرت آن کاسته شده و مقدار زیادی از صدا به هوا بازگشت داده می‌شود.

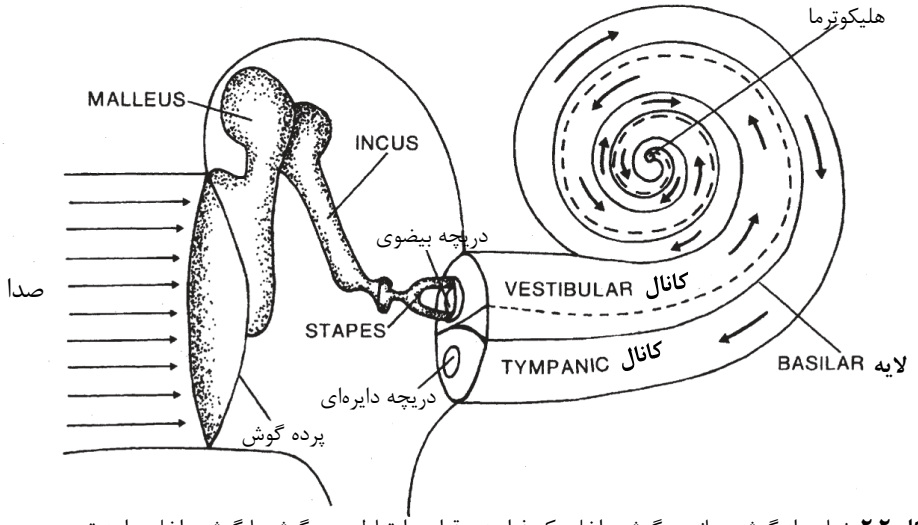
سه قانون فیزیکی برای افزودن کارایی در کیفیت انتقال به وسیله گوش میانی موجود می‌باشد.

۱- خمیدگی پرده‌ی گوش سبب انتقال مکانیکی کاراتری می‌شود
۲- حالت اتصال زنجیره‌ای سه استخوان همانند اهرم عمل نموده و سبب یک سری از مزایای مکانیکی می‌شود
۳- کوچکی *Stapes* چسبیده به دریچه بیضوی

دو ماهیچه درون گوش میانی وجود دارد که می‌توانند شدت اصوات بسیار بلند را جهت جلوگیری از صدمات گوش داخلی کم کنند که یکی از آنان به *Malleus* و دیگری به *Stapes* متصل می‌باشد. تأثیرات این ماهیچه‌ها در برخی موارد با تأثیرات قرنیه چشم قابل مقایسه می‌باشد. صداهای بسیار بلند سبب انقباض این ماهیچه‌ها شده و صوت به اندازه ۰.۶ یا ۰.۷ بابت هر ۱ دسی بل کاسته خواهد شد. محققی به نام *Laurence* اظهار دارد که تحریکات ماهیچه‌های دو گوش مستقل از یکدیگر عمل می‌کند و همین موضوع می‌تواند در تشخیص مسیر صداهای مختلف السمت کمک کند.

ساختار گوش داخلی

ساختار حلزونی استخوانی گوش شامل اعضایی جهت شنیدن می‌باشد. این لوله مارپیچی شامل ۲.۵ دور و طولی حدود ۳.۵ سانت می‌باشد. درون این لوله به سه کانال که با مایع مخصوص پر می‌باشند تقسیم شده است .

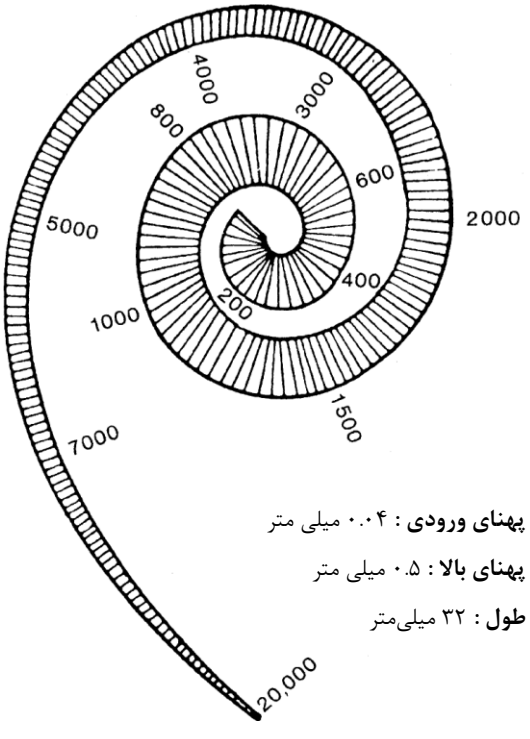


شکل ۲.۲ نمایی از گوش میانی و گوش داخلی که فرایند برقراری ارتباط پرده گوش با گوش داخلی را به تصویر کشیده. پرده گوش استخوان چه‌ها را تحریک نموده و *Stapes* به دریچه بیضوی شکل فشار وارد می‌کند و گوش داخلی این فشار را دریافت نموده، فرکانس‌های مختلف آن را به مغز ارسال می‌نماید.

صوت از طریق پرده گوش به واسطه Stapes به دریچه بیضوی شکل ضربه وارد می‌نماید. و از طریق این ضربه‌ها وارد مایع موجود در کانال Vestibular می‌شود. صوت کل کانال را می‌پیماید تا به دریچه کوچکی به نام هلیکوترما می‌رسد و از این دریچه عبور نموده و وارد کانال Tympanic می‌شود. صوت از طریق مایع درون این کانال باز می‌گردد تا به انتهای آن، که دریچه دایره‌ای قرار دارد برخورد می‌کند. وقتی دریچه

بیضوی شکل به واسطه Stapes به داخل هدایت می‌شود دریچه‌ای دایره‌ای به بیرون هدایت می‌شود که این عمل فشار موجود در گوش را متعادل می‌کند. بدین‌صورت که فشار از کانال Vestibular عبور کرده و از طریق دریچه کوچک هلیکوترما به کانال Tympanic هدایت شده و در نهایت به دریچه دایره‌ای منتقل می‌شود و باعث می‌شود این دریچه کمی به بیرون قوس پیدا کند. بین دو کانال Tympanic و Vestibular، **کانال میانی** قرار گرفته شده. که کانال میانی توسط دو لایه‌ی Basilar و Ressner از دو کانال جدا شده است.

داخل کانال میانی کانال پیچیده‌ای به نام Organ Of Corti بر روی لایه Basilar قرار گرفته که شامل سلول‌های مویی می‌باشد. و نقش اصلی تحریک عصب‌های شنوایی را به عهده دارد. این سلول‌ها سراسر حلزون را در بر دارند. و در مقابل فرکانس‌های مختلف موجود در مایع عکس‌العمل نشان می‌دهند. به این صورت که سلول‌های نزدیک به دریچه بیضوی شکل با فرکانس‌های بالا تحریک می‌شوند و سلول‌های مویی نزدیک به هلیکوترما با فرکانس‌های پایین. در واقع این ارگان نقش یک تبدیل فوریه را بازی می‌کند تا بتواند یک صوت را تجزیه نموده و بر اساس فرکانس‌های مختلف دستورهای مختلفی به مغز ارسال نماید.



شکل ۲.۳ نمایی از لایه *Basilar* در حلزون گوش که ابتدای ان خیلی باریک می‌باشد و با فرکانس‌های بالا تحرک شده و انتهای آن پهن‌تر و با فرکانس‌های پایین تحریک می‌شود.

++++ در اثر فرکانس‌های مختلف نقاط مختلف لایه Basilar تحریک می‌شود. ضخامت این لایه در ابتدا (نزدیک پنجره بیضوی) بسیار باریک و به تدریج با نزدیک شدن به پنجره هلیکوترما پهن‌تر می‌شود.

هر سلول مویی شامل چندین Stereocilia به شکل میله‌های انعطاف پذیر و بسیار باریک می‌باشد که با تغییر فشار در مایع بر اثر امواج صوتی تحرک شده و به لرزش در می‌آیند و از طریق فیبرهایی به مغز فرمان می‌دهد. سلول‌های مویی به دو دسته داخلی و خارجی تقسیم می‌شوند که به گفته برخی از محققان سلول‌های خارجی مسئولیت فرمان دادن به مغز و سلول‌های داخلی مسئولیت فرمان گرفتن از مغز را دارند.

دستگاه تکلم انسان

وقتی قصد داریم روی سیستم توسعه تشخیص صدا کار کنیم، در صورتی که با نحوه تولید صدای انسان آشنایی داشته باشیم، ممکن است کارمان را بسیار آسان‌تر کند.

مهم‌ترین اعضای انسان که مسئولیت تولید صدا را بعهده دارند ریه‌ها ، حنجره ، حلق، بینی و قسمت‌های مختلفی از دهان که در شکل زیر به نمایش در آمده است.

+++++ تصویر

نیروی عضلانی که سبب خروج هوا از ریه‌ها می‌گردد منبع انرژی را جهت صحبت مهیا می‌نماید. صدای ایجاد شده از طریق حلق قبل از خروج توسط دیگر اعضای صوتی انسان تغییر می‌نماید که معمولاً این اعضا به مجاری صوتی انسان معروف هستند. که از حلق شروع شده و از طریق گلو و دهان به لب‌ها ختم می‌شود. از طرفی وقتی به حفره‌های بینی آمیخته می‌شود صدای انسان طنین بسیار پیچیده‌تری تولید می‌کند. فرکانس‌های این طنین صدا که در طول زمان تغییر می‌کند کلید تشخیص اینکه چه چیزی بیان شده می‌باشد.

چند فرکانس اصلی که توسط مجاری صوتی تشکیل شده فرکانس‌های سازنده (formants) می‌نامیم. این فرکانس‌ها در رنج فرکانس‌های پایین (معمولاً از ۲۵۰ هرتز تا ۳۰۰۰ هرتز) قرار دارند. که به اختصار با F_1, F_2, F_3 و غیره مشخص می‌شوند که به طور قابل توجهی در تشخیص حروف نقش دارند. ولی برخی از فرکانس‌های سازنده در فرکانس‌های بالا نیز می‌توانند نقش مهمی در شناسایی برخی حروف داشته باشند.

++ بخش‌های مهمی از این قسمت هنوز تکمیل نشده

۳ پردازش صدا

تشخیص صدا

امروزه جهت تشخیص صدا از مدل سازی‌های مدرن آماری که مدل سازی آکوستیک (صوتی) و مدل سازی زبانی از اجزاء بسیار مهم آن می‌باشند استفاده می‌شود. نویز، محیط‌های متفاوت و حتی میکروفون‌های مختلف سبب متفاوت شدن نمونه‌های صوتی گشته و کار تشخیص را برایمان دشوار می‌کند. اصولاً محیط تشخیص صدا در هوش مصنوعی **تا حدودی قابل مشاهده** (**Partially Observable**) می‌باشد و استفاده از **مدل مخفی مارکوف** بسیار مرسوم می‌باشد. و به همین دلیل از الگوریتم‌های ویژه و حتی ترکیبی جهت طبقه بندی و تشخیص برای این منظور استفاده می‌شود.

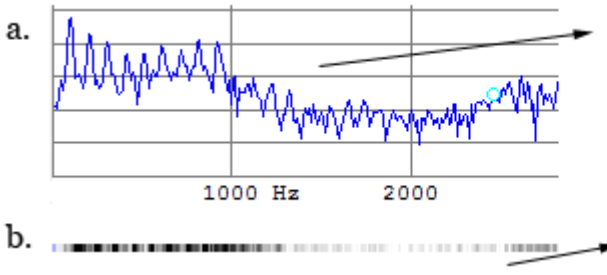
اما هدف این پروژه باز شدن راه ورود دانشجویان مقطع کاردانی و کارشناسی به این زمینه است و سعی نمودیم تا الگوها را با ساده‌ترین روش ممکن تشخیص دهیم و به پیاده سازی آن بپردازیم.

نمودار اسپکتروم

همان‌طور که در بخش اول توضیح دادیم هر نمونه از نمونه‌های دریافتی توسط کارت صدا شامل فرکانس‌های ترکیبی (جمعی از فرکانس‌های آن زمان) می‌باشد. جهت تحلیل روی نمونه‌های دریافتی مجبوریم فرکانس‌های آن را بازیابی نماییم که این کار را توسط عمل تبدیل فوریه انجام می‌دهیم و نمودار دامنه زمانی را به نمودار دامنه فرکانس تبدیل می‌کنیم که به آن نمودار اسپکتروم می‌گوییم. برای اطلاعات بیشتر به بخش اول رجوع نمایید.

نمودار اسپکتروگرام

جهت تحلیل آسان‌تر صوت در قدم اول ما نیاز داریم تغییرات نمودار اسپکتروم را با گذشت زمان مشاهده کنیم. برای این منظور جهت ترسیم هر اسپکتروم بهتر است بجای امتداد بلندی فرکانس در محور y (محور بلندی صدا)، نقطه فرکانس را پر رنگ و یا کم‌رنگ کنیم. تصویر زیر هر دو حالت نمودار اسپکتروم را به نمایش در می‌آورد.



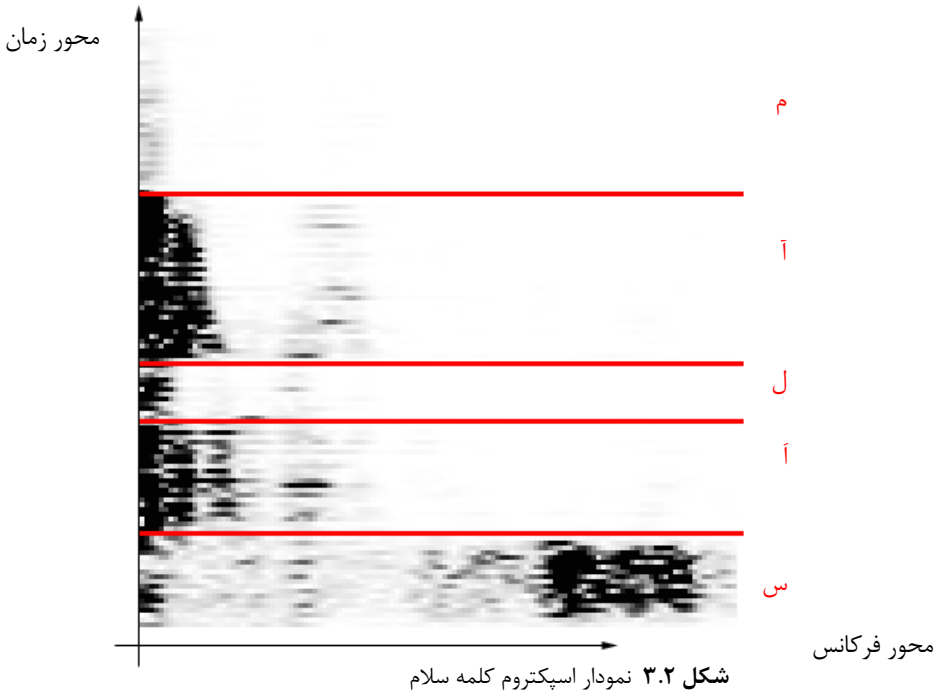
الف) ترسیم از طریق امتداد در محور بلندی فرکانس

ب) ترسیم از طریق پر رنگ و کم رنگ نمودن نقطه فرکانس

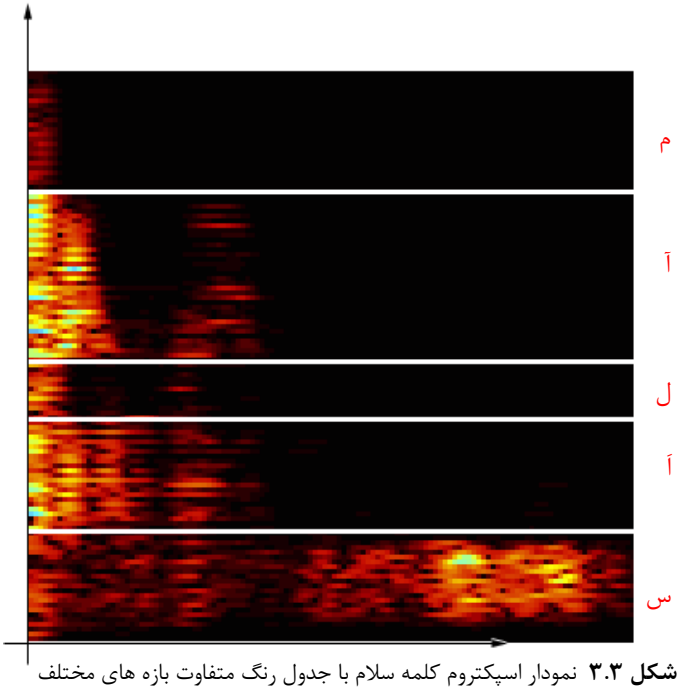
شکل ۳.۱ طریق دیگری از نمایش نمودار اسپکتروم (شکل *b*) که جهت ترسیم نمودار اسپکتروگرام بکار می‌رود

حالا توانستیم نمودار اسپکتروم را در تنها در یک ردیم ترسیم نماییم. جهت مشاهده تغییرات در طول زمان فقط کافی است ردیف‌های اسپکتروم مربوط به هر ردیف را کنار هم قرار دهیم. این نمودار به نمودار اسپکتروگرام معروف می‌باشد. در شکل ۳.۲ نمودار اسپکتروگرام کلمه سلام را مشاهده نمایید.

اصولاً نمودارهای اسپکتروگرام جهت وضوح بیشتر بازه‌های مقادیر مختلف با طیف رنگی مختلف به نمایش در می‌آیند. تصویر ۳.۳ نمودار سلام را با جدول رنگ متفاوت نمایش می دهد.



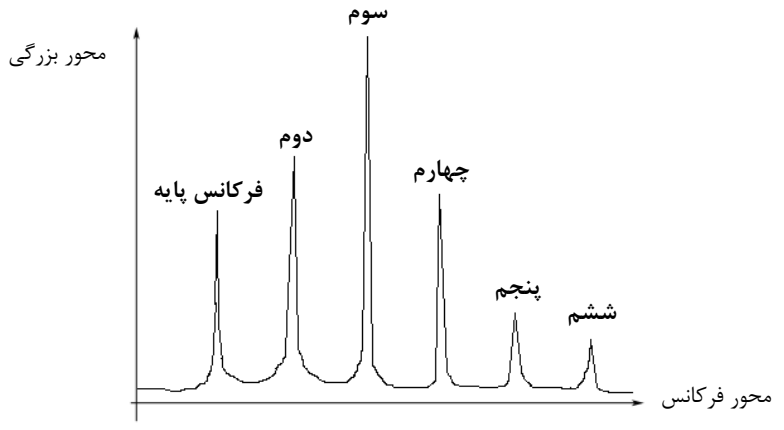
شکل ۳.۲ نمودار اسپکتروم کلمه سلام



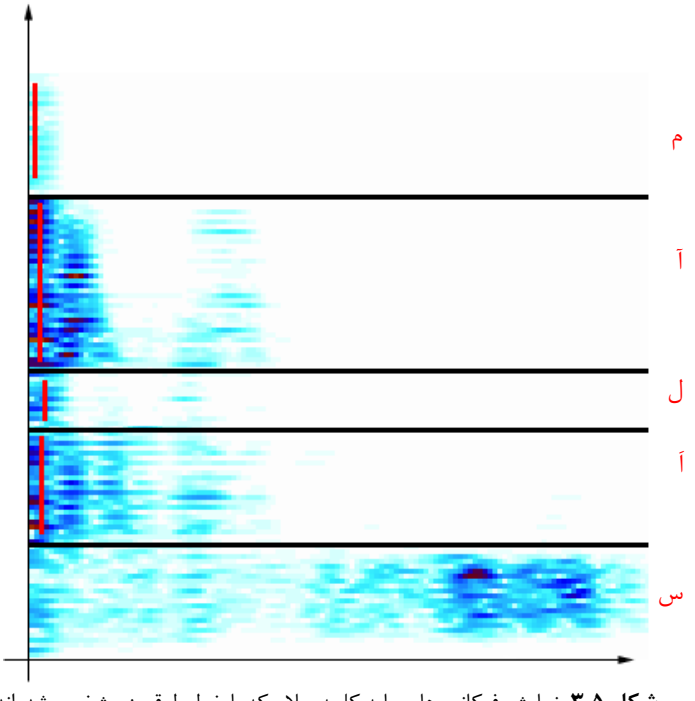
شکل ۳.۳ نمودار اسپکتروم کلمه سلام با جدول رنگ متفاوت بازه های مختلف فرکانس را واضح تر نمایش می دهد.

فرکانس پایه

در صداهای هارمونیک یا پریودیک فرکانسهای هارمونیک مختلفی در نمودار اسپکتروم رویت می شود. اصولا این فرکانسها در محدوده فرکانسهای پایین می باشند که به پایین ترین فرکانس آن (Fundamental Frequency) میگوییم.



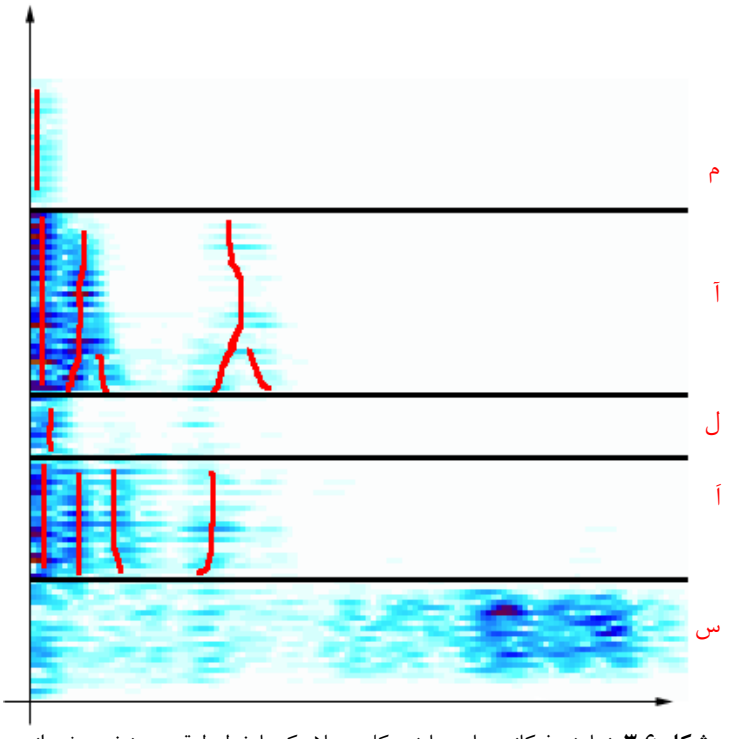
شکل ۳.۴ نمودار اسپکتروم یک صدای هارمونیک که فرکانس پایه در آن مشخص شده.



شکل ۳.۵ نمایش فرکانس‌های پایه کلمه سلام که با خطوط قرمز مشخص شده‌اند. حرف "س" شامل فرکانس پایه نمی‌باشد. چون فرکانس‌های هارمونیک در آن دیده نمی‌شود.

فرکانس‌های سازنده

صداها بجز فرکانس پایه شامل برخی دیگری فرکانس می‌باشند که جزء فرکانس‌های اصلی یک صدا جهت شناسایی می‌باشد. برای تشخیص صدا اصولاً ۳ یا ۴ فرکانس سازنده آن به غیر از فرکانس پایه در نظر می‌گیریم که به آن فرکانس‌های سازنده (Formant) میگوییم. اصولاً فرکانس‌های سازنده در صداهای هارمونیک و حروف صدا دار به وضوح دیده می‌شوند. در شکل زیر فرکانس‌های سازنده با خطوط آبی در اسپکتروگرام سلام مشخص شده‌اند.

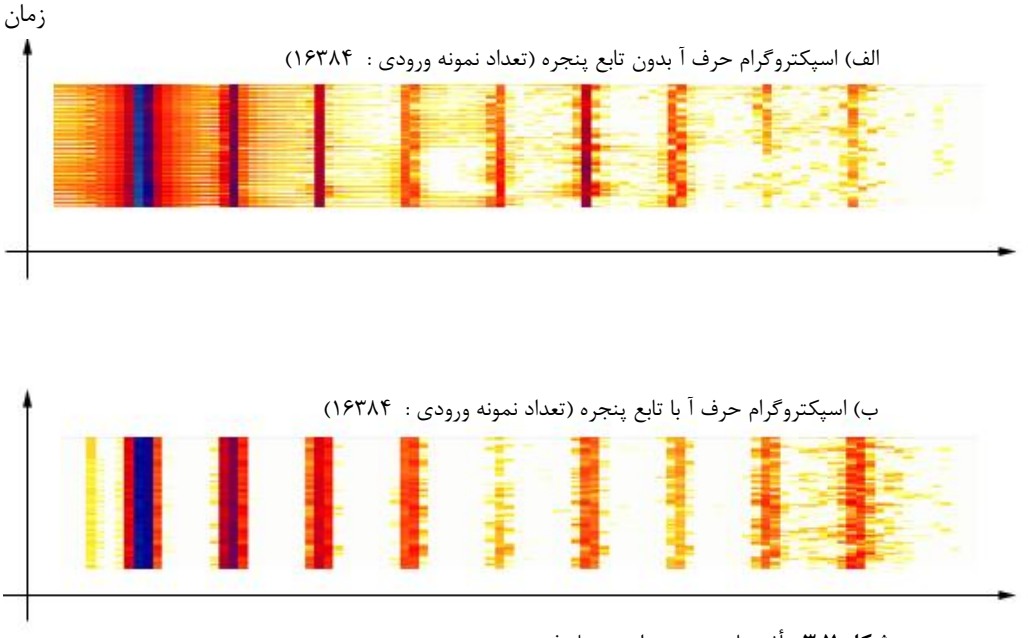


شکل ۳.۶ نمایش فرکانس‌های سازنده کلمه سلام که با خطوط قرمز مشخص شده‌اند.

توابع پنجره

در محاسبات تبدیل فوریه جهت بدست آوردن فرکانس مشکل خاصی دیده می‌شود. از مهم‌ترین مشکلات می‌توان دخالت فرکانس‌های سازنده با قدرت بالا به همسایه‌های مجاور ذکر نمود. توابع پنجره یک سری فرمول ریاضی می‌باشند که قبل از عمل تبدیل روی داده‌های صوتی لحاظ می‌کنیم و به طور چشم

گیری مشکل حل خواهد شد. از جمله توابع پنجره معروف ، پنجره هان، پنجره همینگ، پنجره تاکی، پنجره‌های گاوسشن و پنجره‌های بلکمن می‌باشند که ما برای رفع این مشکل از نوع بلکمن استفاده نمودیم که این تابع شامل یکی دو خط کد برنامه نویسی، به داده‌های ورودی اعمال می‌کنیم. برای اطلاعات بیشتر می‌توانید به سایت ویکیپدیا رجوع نمایید.



شکل ۳.۷ تأثیر تابع پنجره برای تبدیل فوریه

جهت اطلاعات بیشتر در مورد کدهای این قسمت به بخش توضیح کدهای برنامه فیلترها رجوع نمایید:
`SoundAnalysis\Filters\WindowFilter.cs`

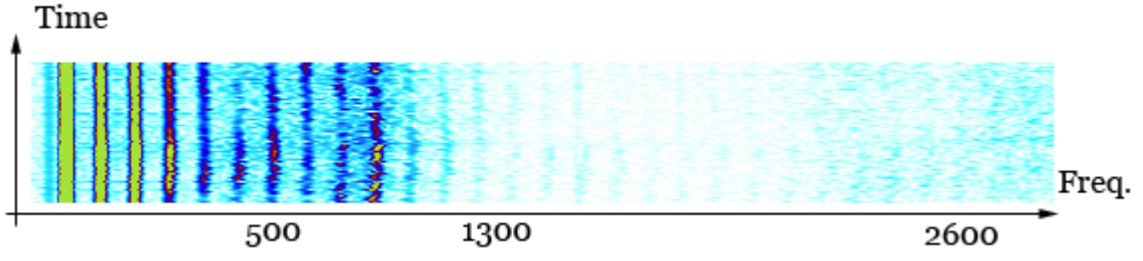
طول پنجره در تبدیل فوریه

ما از کارت صدا تعداد زیادی نمونه‌های صوتی را در واحد زمانی یک ثانیه دریافت می‌کنیم. که با انجام تبدیل فوریه بر روی این نمونه‌های صوتی قادریم انواع فرکانس‌های صوتی موجود در این یک ثانیه را بسیار دقیق بدست آوریم. ولی مشکل اینجاست که در این یک ثانیه تعداد زیادی حروف تلفظ خواهند شد و ما نمی‌توانیم بفهمیم کدام فرکانس مربوط به کدام حرف می‌باشد. یک راه حل این است که نمونه‌های در یافت شده را به اندازه‌های کوچک‌تر تقسیم کنیم و بعد از آن تبدیل فوریه بگیریم. مسلماً هرچقدر تعداد نمونه‌ها جهت تبدیل کمتر باشد جزئیات فرکانس کم‌تری دریافت خواهیم نمود.به طور مثال اگر ما ۱۹۲۰۰۰

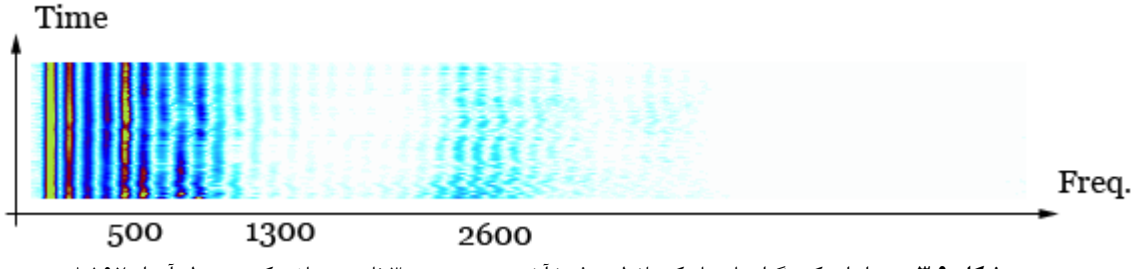
نمونه از کارت صدا در ثانیه دریافت نماییم اگر بخواهیم فرکانس‌ها را در بازه‌های ۱۰۰ میلی ثانیه بدست آوریم باید نمونه‌های ورودی را به ۱۰ قسمت تقسیم کنیم. که به این قسمت‌ها طول پنجره می‌گوییم.

باند پهن (Wide Band) در مقابل باند باریک (Narrow Band)

جهت دریافت اطلاعات فرکانسی بیشتر باید طول پنجره را بیشتر انتخاب نماییم که در این صورت فرکانس‌های دقیق‌تری دریافت خواهیم نمود ولی در عوض تغییرات صحبت در زمان را کمتر متوجه خواهیم شد. که خروجی آن اسپکتروم باند باریک (Narrow Band) نامیده می‌شود. ولی در صورتی که نیاز دارید از فرکانس‌های موجود در زمان‌های بسیار کوتاه با خبر شوید باید طول پنجره را کمتر در نظر بگیرید. در این صورت خروجی از کیفیت فرکانسی پایین‌تری برخوردار خواهد بود که به خروجی آن اسپکتروم باند پهن (Wide Band) میگوییم. جهت دریافت فرکانس‌های گفتار به دلیل اینکه در ثانیه ممکن است چندین حرف تلفظ شود مجبوریم از اسپکتروم‌های باند پهن استفاده نماییم. باید طول پنجره را حدود ۳۰ میلی ثانیه در نظر بگیریم . در ادامه شکل چند اسپکتروگرام را با طول‌های پنجره متفاوت مشاهده می‌نمایید. کلیه تصاویر زیر با نرخ نمونه برداری ۱۹۲۰۰۰ نمونه در ثانیه توسط برنامه آزمایشگاه خودمان تهیه شده اند.

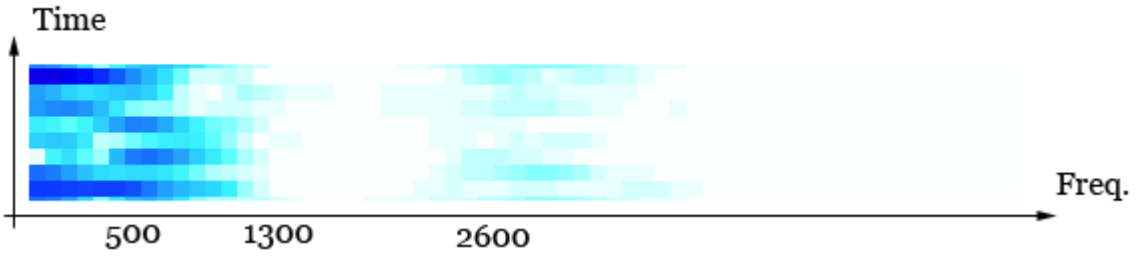


شکل ۳.۸ نمودار اسپکتروگرام باند باریک تلفظ حرف (آ) به مدت حدود ۶ ثانیه می‌باشد که هر سطر آن از ۱۶۳۸۴ نمونه صوتی بدست آمده. و فرکانس‌های هارمونیک به وضوح مشخص می‌باشند.



شکل ۳.۹ نمودار اسپکتروگرام باند باریک تلفظ حرف (آ) به مدت حدود ۳ ثانیه می‌باشد که هر سطر آن از ۸،۱۹۲ نمونه صوتی بدست آمده. و فرکانس‌های هارمونیک به وضوح مشخص می‌باشند.

هر چند فرکانسهای هارمونیک در هر دو تصویر ۳.۸ و ۳.۹ به وضوح قابل رویت می باشند ولی شکل ۳.۸ اطلاعات دقیقتری از فرکانس به ما نمایش می دهد.



شکل ۳.۱۰ نمودار اسپکتروگرام باند پهن تلفظ حرف (آ) به مدت زمان بسیار کوچک ۳۰ میلی ثانیه تهیه شده که هر سطر آن از ۱۰۲۴ نمونه صوتی بدست آمده و فرکانس‌ها از وضوح بسیار کمتری قرار دارند.

ما در این پروژه نمونه‌ها را به اندازه‌های ۱۰۲۴ تایی جهت تبدیل فوریه انتخاب می‌کنیم چرا که قادر خواهیم بود کوچک‌ترین تغییرات تلفظی در زمان بدست آوریم.

همان‌طور که گفتیم ما نمونه‌ها را ۱۹۲۰۰۰ نمونه در ثانیه از کارت صدای رایانه دریافت می‌کنیم. این تعداد نمونه برای تبدیل فوریه و دریافت فرکانس آن‌ها مناسب ما نمی‌باشد. چون ممکن است در یک ثانیه چندین حرف تلفظ شود. بنابراین این تعداد نمونه‌ها را به ترتیب به تکه‌های ۱۰۲۴ تایی تقسیم می‌کنیم و فرکانس‌های هر یک از این دسته‌ها را به صورت مستقل محاسبه خواهیم نمود.

fft bin

وقتی ما قصد داریم از طریق تبدیل فوریه فرکانس‌های تعدادی از نمونه‌های صوتی را بدست آوریم فرمول تبدیل فوریه سریع به گونه ایست که آرایه‌ای به تعداد n عنصر به عنوان نمونه‌های صوتی که از کارت صدا دریافت نمودیم را دریافت کرده و آرایه‌ای به تعداد n عنصر به عنوان بلندی فرکانس به عنوان خروجی به ما باز می‌گرداند. به کد زیر توجه کنید :

```
freqData = Fft(sampleData) ;
```

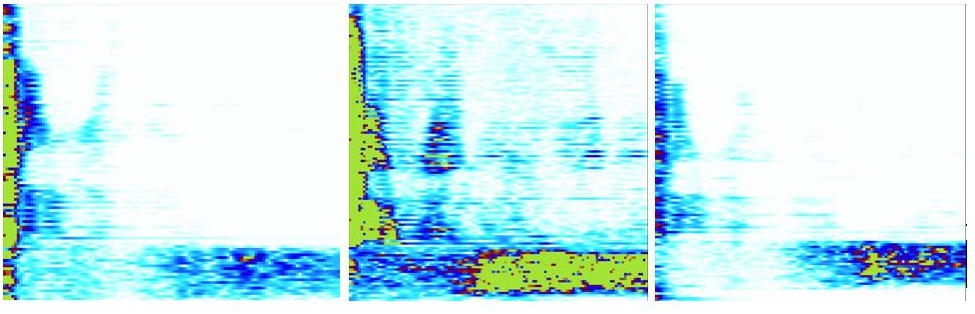
در کد فوق `sampleData` نمونه‌های ورودی و `freqData` آرایه‌ای از مقادیر بلندی صدای فرکانس می‌باشد. نکته قابل توجه اینجاست که ما به عنوان خروجی تنها مقادیر بلندی فرکانس‌ها را داریم پس خود

فرکانس را از کجا بدست آوریم؟ آیا شماره آرایه فرکانس می‌باشد؟ خیر شماره آرایه `fft bin` نامیده می‌شود و ما باید بر اساس تعداد فرکانس‌های ممکن به صورت نسبی از روی `fft bin` آن را محاسبه کنیم. برای مثال اگر نمونه گیری شما ۱۹۲۰۰۰ در ثانیه است. پس می‌توان تصور نمود همین تعداد فرکانس وجود دارد. اگر تعداد نمونه‌های مان جهت محاسبه تبدیل سریع فوریه ۱۰۲۴ عدد می‌باشند. نسبت ۱۰۲۴ به ۱۹۲۰۰۰ را بدست می‌آوریم. پس هر شماره آرایه باید در این نسبت ضرب شود تا فرکانس مربوطه مشخص شود و مقدار موجود در شماره آرایه بلندی مربوط به این فرکانس می‌باشد.

منظور از بلندی فرکانس بزرگی صدای آن فرکانس است، همان طور که میدانید هر یک از فرکانس‌های صدای موجود در محیط یک بزرگی صدا دارد.

محیط گوینده، نویز، میکروفون

قسمت مهمی از مشکلات در تشخیص صدا محیط و حتی نوع میکروفون می‌باشد که با تغییر هر یک و حتی با تغییر ساعات در یک محیط نمونه گیری را نسبت به یکدیگر متفاوت می‌سازد. به طور مثال یک میکروفون حساسیت بسیار زیادی دارد در حالی که میکروفون دیگر به سختی در مقابل صداهای بیرونی عکس‌العمل نشان می‌دهد. دو نمونه میکروفون که در یک زمان حرف آ را رکورد نمودند در زیر به نمایش در آمدند:



الف) میکروفون ۱ ب) میکروفون ۲ ج) میکروفون ۳

شکل ۳.۱۱ نمودار اسپکتروگرام تلفظ سلام با سه میکروفون متفاوت در یک

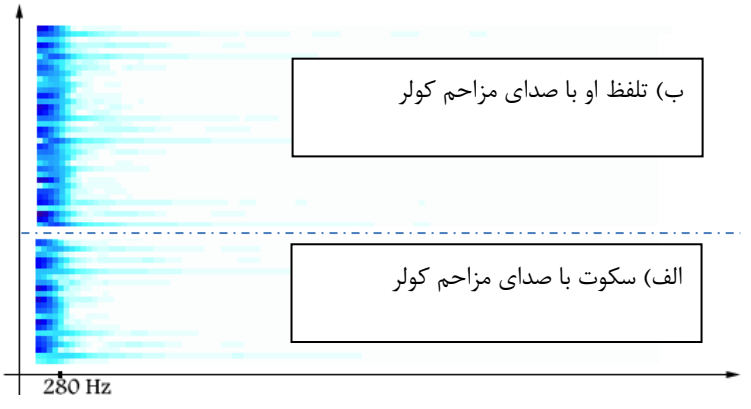
برخی اوقات اصوات مزاحم به قدری زیاد می‌باشند که تشخیص حروف را برایمان مشکل می‌نمایند. البته ممکن است گوش شما با نویز محیط خو گرفته باشد و شما احساس نکنید ولی وقتی به اسپکتروگرام نگاهی

می‌اندازید چیزی می‌بینید که زیاد برایتان خوش آیند نیست. به هر حال تمام این مسائل قابل حل می‌باشند. ما نیز در این پروژه هرچند مختصر به این مسائل می‌پردازیم.

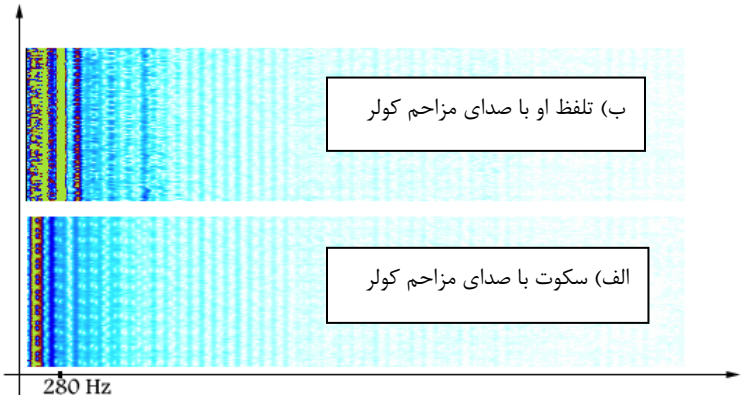
تشخیص صحبت / سکوت

چه وقت کاربر در حال صحبت است؟ در این قسمت باید راه کاری ارائه دهیم تا سکوت کاربر را از صحبت آن تشخیص داده تا بتوانیم در هنگام سکوت از تشخیص‌های اشتباه خود داری نماییم. به عبارت دیگر وقتی که کاربر به سکوت می‌نشیند. از محیط صداهایی دریافت می‌کنیم هر چند ضعیف باشد ولی با پردازش آن صداها ممکن است به الگوهای قابل فهم برخورد کنیم در صورتی که مد نظر کاربر نیست. البته نکته مثبت اینکه وقتی ما شروع به صحبت می‌کنیم قسمت مهمی از نویزها با انرژی بیشتر صحبت ما از بین می‌رود ولی در هنگام سکوت نویزهای معنا دار و مزاحم را چه کنیم؟

بر اساس تئوری معروف نیکوئست (Nyquist) اینکه فقط به تعداد نیمی از نمونه‌های ورودی فرکانس‌ها قابل تشخیص می‌باشند. یعنی وقتی ما ۱۰۲۴ نمونه داریم فقط ۵۱۲ فرکانس آن قابل شناسایی می‌باشند. و از این ۵۱۲ فرکانس ما اغلب به یک پنجم فرکانس‌های پایین جهت تشخیص صداهای هارمونیک نیاز داریم که قسمت بسیار مهم ان با صوت‌های مزاحم مخلوط باشد! شناسایی تعدادی حروف واقعاً مشکل خواهد شد. ولی در صورتی که نمونه‌های ورودیمان بالا باشد باشند ما توانایی شناسایی رنج بیشتری از فرکانس آن را خواهیم داشت و به راحتی صداهای مزاحم قابل شناسایی خواهند بود. در زیر



شکل ۳.۱۲ نمودار اسپکتروگرام باند پهن الف) تلفظ حرف "او" با نمونه‌های ۱۰۲۴ تایی و ب) سکوت همراه با وجود صدای مزاحم (صدای کولر).



شکل ۳.۱۳ نمودار اسپکتروگرام باند باریک الف) تلفظ حرف "او" با نمونه‌های ۱۶۳۸۴ تایی و ب) سکوت همراه با وجود صدای مزاحم (صدای کولر).

یکی از راهکارها می‌تواند این باشد که صدای محیط دائماً بررسی شود و اگر ماندگاری آن بیشتر از حد معمول بود حتماً آن مقدار ماندگار در فرکانس خاص اندازه نویز است. و ما محاسبات را نسبت به پارامتر اندازه نویز فعلی بسنجیم. راه کار دیگر که ما در این پروژه استفاده نمودیم اینست که در بازه‌های ۵ ثانیه سطح نویز را اندازه گیری می‌کنیم. به این صورت که سعی می‌کنیم کم‌ترین مقدار را در محدوده فرکانس خاص ثبت کنیم. و هر ۵ ثانیه برای اطمینان کم‌ترین مقدار را نا معتبر می‌کنیم تا وقتی سطح نویز تغییر نمود مجدداً قابل محاسبه شود. در این بین به مشکلی برخورد نمودم که به دلیل برخی دیگر از مشکلات در تبدیل فوریه ، از میان هر چند تبدیل برخی مواقع فرکانس‌ها با مقادیر کمتری محاسبه می‌شوند. شمارش نمودن کم‌ترین‌ها راه حل خوبی بود که بکار گرفته شد. به این صورت که کم‌ترین مقدار باید حد اقل چند بار تکرار شود تا نویز پایین در نظر گرفته شود.

از این تکنیک می‌توانیم جهت شناسایی اینکه آیا کاربر در حال صحبت است یا نه استفاده کنیم. بدین صورت که اگر صدا از حد نویز حدود ۲ برابر بالاتر رفت نشان دهنده آن است که کاربر در حال تکلم است. در شکل زیر برنامه صوت‌های مزاحم را تشخیص داده و فرکانسی دیده نخواهد شد.

جهت اطلاعات بیشتر در مورد کدهای این قسمت به بخش توضیح کدهای برنامه فیلترها رجوع نمایید:
`SoundAnalysis\Filters\NoiseAnalyzerFilter.cs`

تشخیص حروف

در قدم بعدی باید بتوانیم تا جای ممکن حروف الفبا را در سطح یک اسپکتروم شناسایی نماییم. البته لازم نیست تمام حروف الفبا شناسایی شوند. چرا که به دلیل شباهت بسیار زیاد برخی از حروف در مرحله نخست قادر به تبعیض و شناسایی آن‌ها نمی‌باشیم. مانند حروف (م) و (ن). و همان طور که در آینده توضیح خواهیم داد برخی حروف نیز در سطح پردازش اسپکتروگرام شناسایی خواهند شد. و برخی دیگر نیز با استفاده از روش‌های مدرن طبقه بندی قابل شناسایی می‌باشند. هرچقدر بیشتر شناسایی کنیم، تشخیص از دقت بالاتری برخوردار خواهد بود. حتی اگر تعداد بسیار کمی از حروف شناسایی شوند باز هم قادریم تشخیص کلمه انجام دهیم ولی با خطاهای بسیار زیاد و دقت پایین.

تفکیک حروف صدا دار از حروف بی صدا

حروف صدا دار نسبت به حروف بی‌صدا بم‌تر (در محدوده فرکانس پایین) و از فرکانس‌های هارمونیک بسیار بیشتری برخوردارند ولی حروف بی صدا بسیار زیر(در محدوده فرکانس بالا) و اغلب در آن‌ها فرکانس‌های هارمونیک وجود ندارد. اگر در تصویر اسپکتروگرام سلام دقت کنید، حروف "آ"، "ل"، "ا" و "م" صدا دار می‌باشند و قسمت فرکانس‌های پایین پررنگ بوده ولی کلمه س بی صدا بوده و قسمت فرکانس‌های بالا پررنگ می‌باشد. ظاهراً شناسایی آنان بسیار آسان می‌باشد، قطعاً همین‌طور است!

پس از شناسایی حروف بی صدا و با صدا قادر خواهیم بود کلمات را تشخیص دهیم. فرض کنید بانک اطلاعاتی از کلمات فارسی داریم که برای هر کلمه حروف صدا دار و بی صدای آن را مشخص نمودیم. حرف صدا دار را با + و حرف بی صدا را با – مشخص می‌کنیم. جدول زیر را در نظر بگیرید :

ستون کلمه فارسی	ستون صداهای کلمه
سلام	– ++++
بنام خدا	– +++++ –+ +
خدا حافظ	– +– +– +– +–
در	– ++
بر	– ++
سر	– ++

همان‌طور که مشاهده می‌فرمایید با کم‌ترین دانش قادر به پیاده سازی این پروژه هستیم. ولی بهتر است عجله نکنیم، مسائلی وجود دارد که قبل از پیاده سازی بیشتر باید به آن‌ها بپردازیم و البته بهتر است بازه شناسایی حروفمان را بالاتر ببریم. ما در ادامه مشخصه‌های برخی از حروف مهم الفبا مانند انواع حروف صدا دار و برخی حروف مهم دیگر را شرح خواهیم داد.

شناسایی حروف

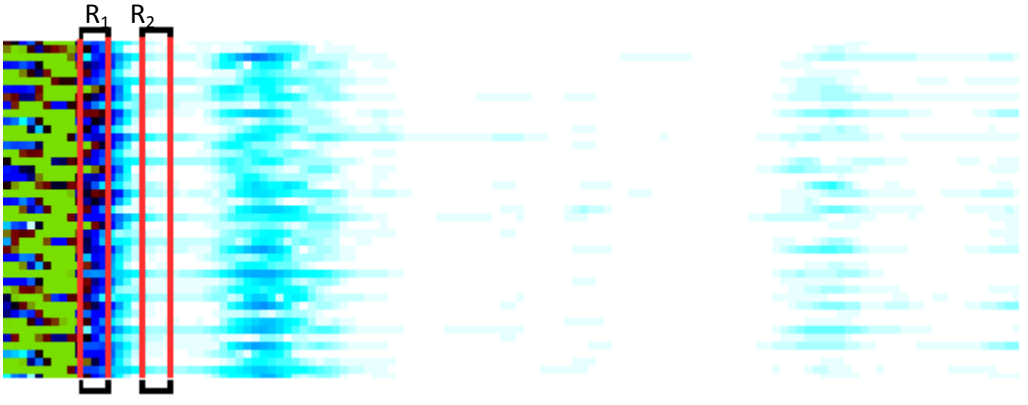
زمان خوبی است که از طریق چشم خود به دنبال ویژگی‌های منحصر به فرد حروف گشته و آن‌ها را استخراج کنیم. البته بهتر بود با یک الگوریتم هوش مصنوعی همراه با آموزش نمونه‌های مختلف به صورت نرم افزاری این ویژگی‌ها را استخراج کنیم. ولی برای این پایان نامه در همین حد چشمی کافی است. در این قسمت قصد داریم ویژگی‌های منحصر به فردی برای حروف آ، اَ، اِ، ای، س، ش ، ز، ژ، ج، چ پیدا کنیم. این ویژگی‌ها را از طریق نمودار اسپکتروگرام حروف یافت نموده و سعی می‌کنیم بر روی یک اسپکتروم به دنبال این ویژگی‌ها جهت شناسایی بگردیم.

کلیه ویژگی‌هایی که در این قسمت بیان خواهیم نمود ویژگی صدای مردان می‌باشد. البته جهت سازگاری صدای زنان اندک تغییراتی نیاز است و نیاز داریم تا زیر و بمی صدای گویند را مشخص کنیم که در بخش بعدی به تفصیل به انجام این عمل می‌پردازیم.

شناسایی حروف صدا دار

تشخیص حرف آ :

جهت شناسایی حروف آ سعی می‌کنیم با کمک میانگین بین مناطق مختلف وجه تمایزی برای آن‌ها در نظر بگیریم. بار دیگر به اسپکتروم حرف آ نگاه بیندازید.



شکل ۳.۱۴ اسپکتروگرام حرف آ می‌باشد که جهت استخراج ویژگی‌های آن، دو منطقه R_1 از ۸۴۰ تا ۱۲۰۰ هرتز و R_2 از ۱۴۰۰ تا ۱۶۰۰ هرتز در نظر گرفته شده.

ما نسبت میانگین مقادیر بین فرکانس‌های ۸۴۰ تا ۱۲۰۰ در منطقه R_1 را با میانگین مقادیر بین فرکانس‌های ۱۴۰۰ تا ۱۶۰۰ در منطقه R_2 بررسی می‌کنیم همان‌طور که در تصویر هم مشاهده می‌کنید، انتظار داریم قسمت اول حداقل چهار برابر بزرگ‌تر از قسمت دوم باشد. و این عمل با چهار عمل اصلی قابل پیاده سازی است. و در ریاضیات به صورت زیر قابل نمایش می‌باشد:

$$Rel1 = \frac{\sum_{f=840}^{1200} I_f}{1200-840} \Big/ \frac{\sum_{f=1600}^{1400} I_f}{1600-1400}$$

در عبارت فوق I_f مقدار بزرگی فرکانس f می‌باشد. و $Rel1$ نسبت بدست آمده می‌باشد که انتظار داریم **بیشتر از ۴** باشد.

و برنامه نویسی به زبان c# آن نیز به صورت زیر می‌باشد :

```
int Detect(double[] freq)
        {
            sum1 = sum2 = 0;

            // محاسبه جمع مقادیر منطقه ۱
            for (i = beg1; i < end1; i++)
                sum1 += freq[i];

            // محاسبه جمع مقادیر منطقه ۲
            for (i = beg2; i < end2; i++)
                sum2 += freq[i];

            // محاسبه میانگین ها
            avg1 = sum1 / (end1 - beg1);
            avg2 = sum2 / (end2 - beg2);

            // نسبت میانگین ها
            rel1 = avg1 / avg2;

            if (rel1 > 4)
                return 100;

            return 0;
        }
```

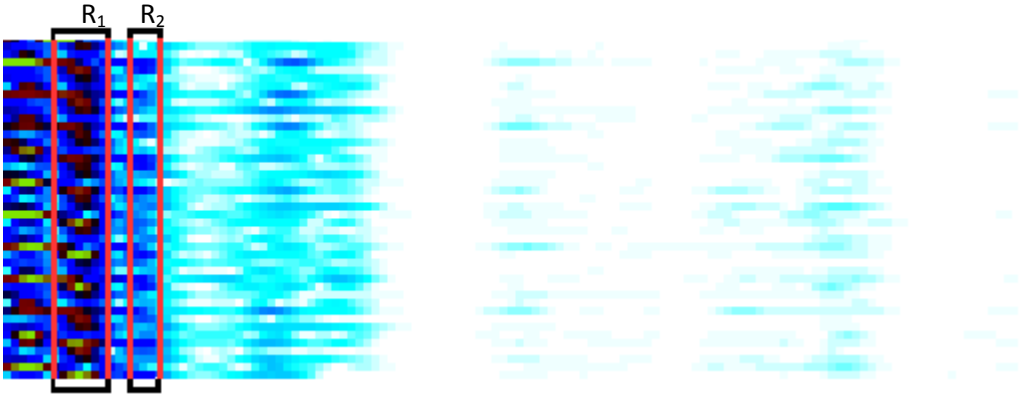
در کد فوق $beg1 = 840$ ، $end1 = 1200$، $beg2 = 1400$، $end2 = 1600$ می‌باشد. و `freq[i]` مقدار بلندی فرکانس می‌باشد. اگر حرف آ شناسایی شد ۱۰۰ و در غیر این صورت خروجی تابع صفر در نظر گرفته می‌شود.

در این قسمت از بیان تبدیل فرکانس به fft bin صرف نظر کردیم که در اصل بجای فرکانس مقادیر fft bin به این متغیرها اختصاص داده خواهند شد.

چیزی که باید شما بدانید آنکه باید اعتراف کنم دقت زیادی در شناسایی حروف انجام ندادم و شما با کمی دقت بیشتر قادرید با کیفیت بیشتری به شناسایی بپردازید. چه برسد که بخواهید این عمل را با روش‌های آماری مدرن و هوش مصنوعی انجام دهید.

جهت اطلاعات بیشتر در مورد کدهای این قسمت به بخش توضیح کدهای برنامه تشخیص حروف رجوع نمایید: `SoundAnalysis\Recognition\Phoneme\PhonemeDetector_AH.cs`

تشخیص حرف آ:



شکل ۳.۱۵ اسپکتروگرام حرف آ می‌باشد که جهت استخراج ویژگی‌های آن، دو منطقه R_1 از ۵۰۰ تا ۱۲۰۰ هرتز و R_2 از ۱۴۰۰ تا ۱۶۰۰ هرتز در نظر گرفته شده.

ما نسبت میانگین مقادیر بین فرکانس‌های ۵۰۰ تا ۱۲۰۰ در منطقه R_1 را با میانگین مقادیر بین فرکانس‌های ۱۴۰۰ تا ۱۶۰۰ در منطقه R_2 بررسی می‌کنیم و انتظار داریم قسمت اول حد اکثر از ۶ برابر بزرگ‌تر از ناحیه دوم نباشد!. و این عمل با چهار عمل اصلی قابل پیاده سازی است. و در ریاضیات به صورت زیر قابل نمایش می‌باشد:

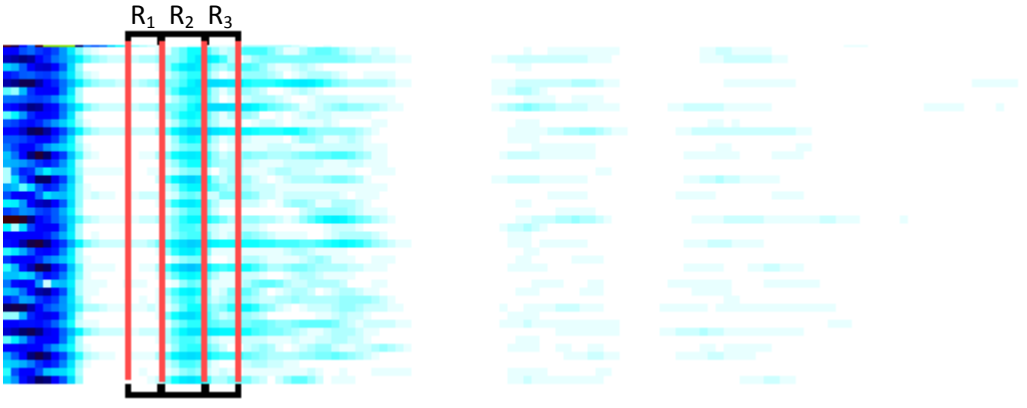
$$Rel1 = \frac{\sum_{f=500}^{1200} I_f}{1200-840} \Big/ \frac{\sum_{f=1600}^{1400} I_f}{1600-1400}$$

در عبارت فوق I_f مقدار بزرگی فرکانس f می‌باشد. و $Rel1$ نسبت بدست آمده می‌باشد که انتظار داریم **کمتر از ۶** باشد.

هرچقدر آیتم‌ها جهت تشخیص بیشتر شوند مشکلات این روش شناسایی بیشتر خواهد شد چرا که احتمال یکسان بودن ویژگی‌ها بیشتر و بیشتر می‌شود. بدون داشتن حد اقل دانش آمار و احتمالات کمی کارمان مشکل‌تر و با خطای بیشتری روبرو خواهد بود.ولی ما سعی می‌کنیم در ادامه با کمی سماجت برخی دیگر از حروف را شناسایی کنیم.

جهت اطلاعات بیشتر در مورد کدهای این قسمت به بخش توضیح کدهای برنامه تشخیص حروف رجوع نمایید: SoundAnalysis\Recognition\Phoneme\PhonemeDetector_AA.cs

تشخیص حرف اِ :

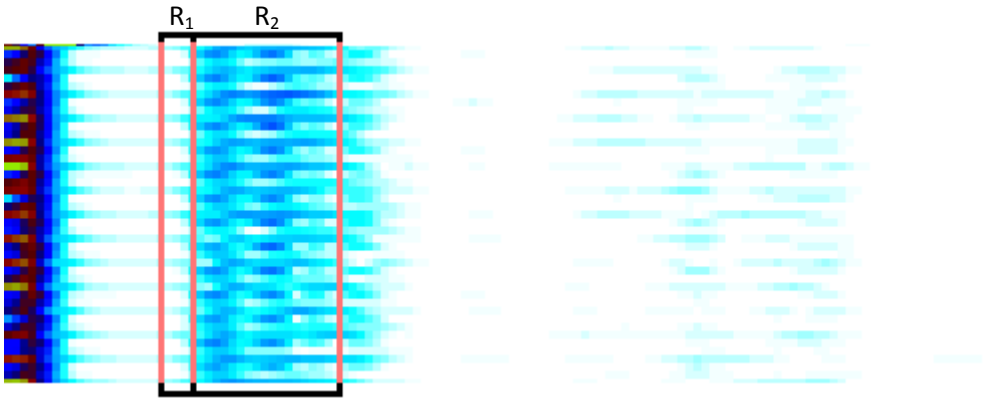


شکل ۳.۱۶ اسپکتروگرام حرف اِ می‌باشد که جهت استخراج ویژگی‌های آن، سه منطقه R_1 از ۵۰۰ تا ۱۲۰۰ هرتز، R_2 از ۱۲۰۰ تا ۱۴۰۰ هرتز و R_3 از ۱۴۰۰ تا ۱۶۰۰ هرتز در نظر گرفته شده

شرط شناسایی حرف اِ را این‌گونه تعریف می‌نماییم که میانگین مقادیر R2 بزرگ‌تر از R1 و R3 کوچک‌تر از R2 باشد. البته شرط دوم برای محکم کاری است و می‌تواند حذف شود. البته در برخی نقاط مغایرت دارد که در شناسایی کلمات رای با اکثریت می‌باشد.

جهت اطلاعات بیشتر در مورد کدهای این قسمت به بخش توضیح کدهای برنامه تشخیص حروف رجوع نمایید: SoundAnalysis\Recognition\Phoneme\PhonemeDetector_EH.cs

تشخیص حرف ای :



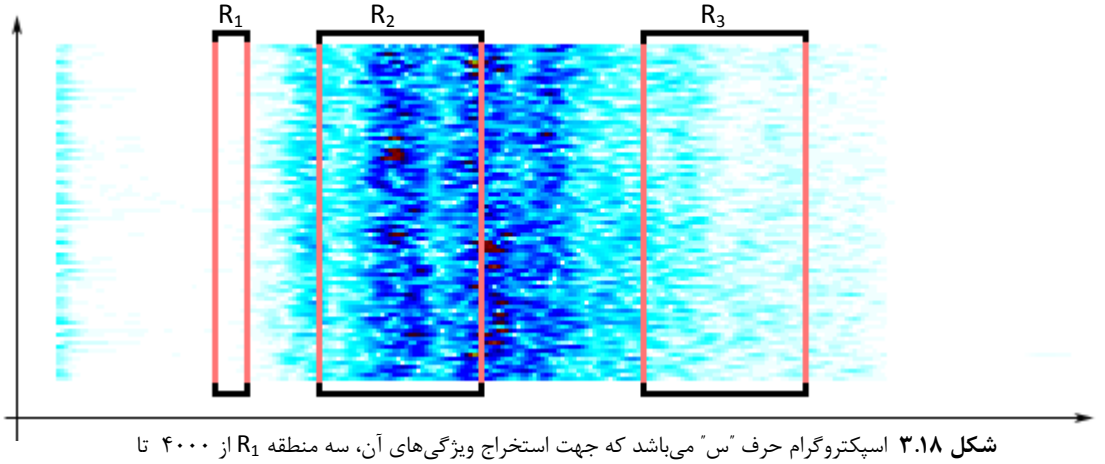
شکل ۳.۱۷ اسپکتروگرام حرف ای می‌باشد که جهت استخراج ویژگی‌های آن، دو منطقه R_1 از ۱۶۰۰ تا ۲۰۰۰ هرتز، R_2 از ۲۰۰۰ تا ۴۰۰۰ هرتز در نظر گرفته شده

شرط شناسایی حرف ای بزرگ‌تر بودن میانگین مقادیر منطقه دوم نسبت به منطقه اول است.

تشخیص حروف بی صدا و نیم صدا

حروف بی صدا در آن ها قسمت هارمونیک دیده نمی شود و تنها در قسمت فرکانسهای بالاتر آنها مقدار زیادی نویز دیده می شود. که شامل حروف (س، ش، چ،ح، خ،ت، ف،ک) می باشند. برخی دیگر حروف وجود دارند که علاوه بر وجود نویز در قسمت فرکانسهای بالا فرکانسهای هارمونیک هم در انها وجود دارد کا ما آنها را نیم صدا می نامیم و شامل حروف (ر، ز، ژ، ق،ب،ج، د،گ) می باشند.

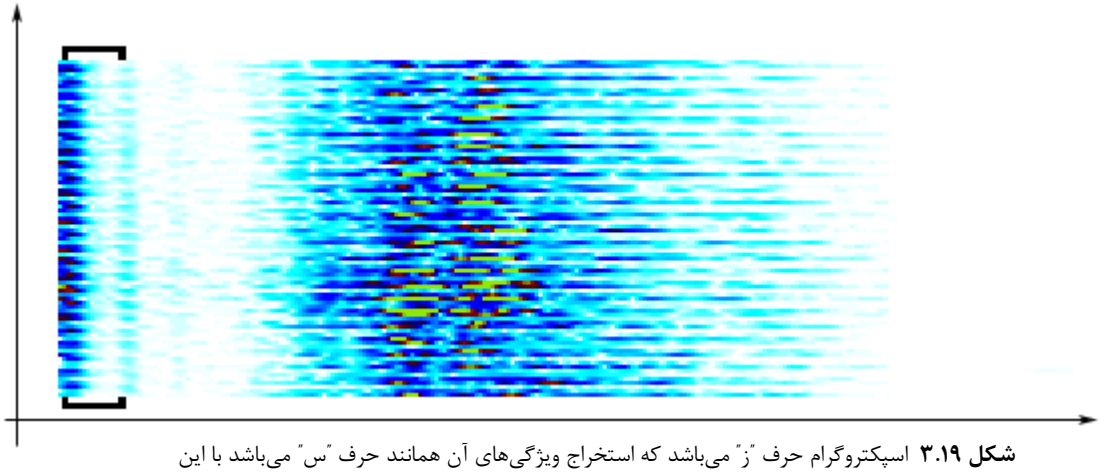
تشخیص حرف س :



شکل ۳.۱۸ اسپکتروگرام حرف "س" می‌باشد که جهت استخراج ویژگی‌های آن، سه منطقه R_1 از ۴۰۰۰ تا ۶۰۰۰ هرتز، R_2 از ۶۰۰۰ تا ۱۰۰۰۰ هرتز و R_3 از ۱۴۰۰۰ تا ۱۶۰۰۰۰ هرتز در نظر گرفته شده

تلفظ حرف ″ س ″ فرکانس‌های غیر هارمونیک در ناحیه ۶۰۰۰ تا ۱۴۰۰۰ تولید می‌کند. و ما از طریق سه مقایسه منطقه‌ای که در تصویر مشخص شده می‌توانیم حرف ″س ″ را تشخیص دهیم. بدین صورت که میانگین منطقه دوم باید از میانگین منطقه یک و همچنین از میانگین منطقه سه بزرگ‌تر باشد.

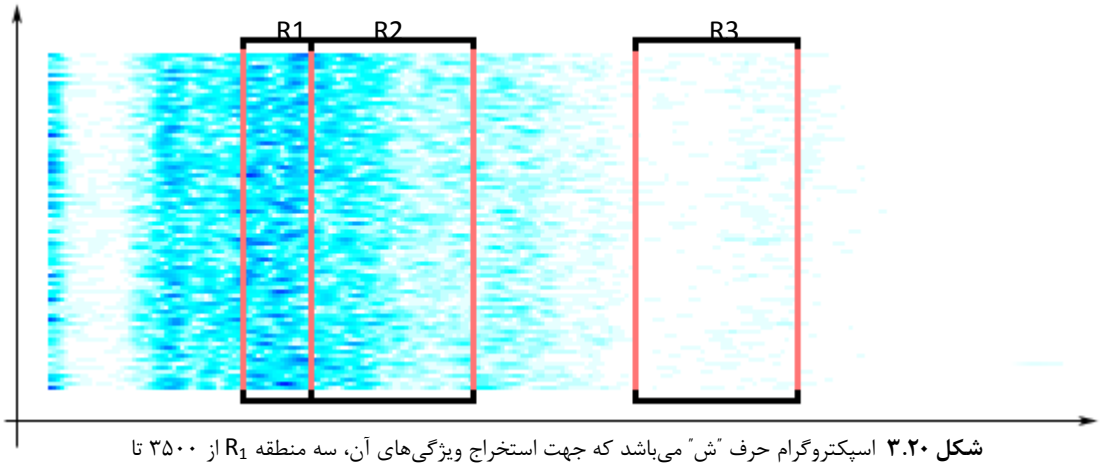
تشخیص حرف ز :



شکل ۳.۱۹ اسپکتروگرام حرف ″ز″ می‌باشد که استخراج ویژگی‌های آن همانند حرف ″س″ می‌باشد با این تفاوت که در فرکانس‌های پایین آن فرکانس‌های سازنده موجود می‌باشد

تشخیص حرف ″ ز ″ دقیقاً مانند تشخیص حرف ″ س ″ می‌باشد و نیازی به کد نویسی نمی‌باشد. و تمایز بین آن‌ها از فرکانس‌های هارمونیک آن‌هاست. چون اگر دقت کنید برای تلفظ حرف ″ ز ″ از حنجره صدای هارمونیک نیز خارج می‌کنیم و برای پیدا کردن اینکه در قسمت هارمونیک صدا وجود دارد یا خیر می‌توانیم یک فیلتر تحلیلگر نویز بین فرکانس‌های پایین قرار دهیم.

تشخیص حرف ش :

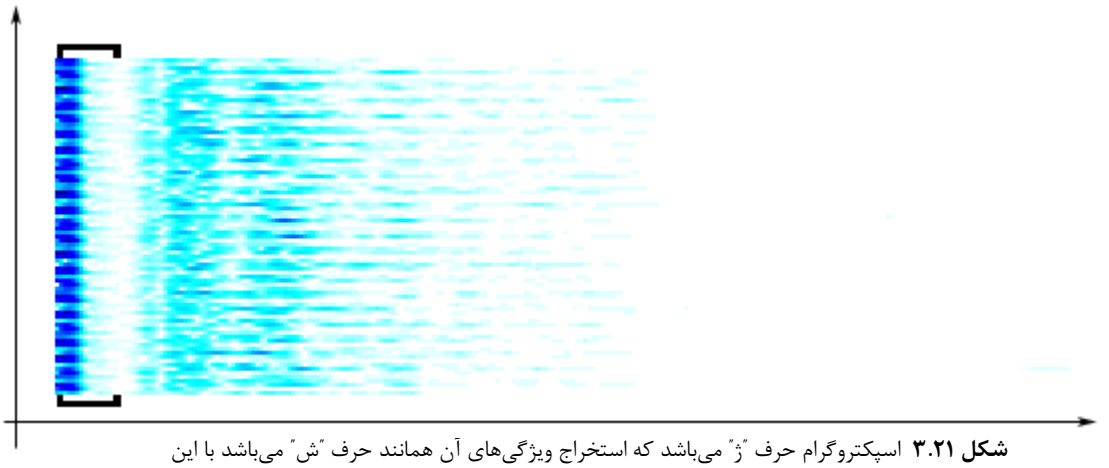


شکل ۳.۲۰ اسپکتروگرام حرف "ش" می‌باشد که جهت استخراج ویژگی‌های آن، سه منطقه R_1 از ۳۵۰۰ تا ۴۰۰۰ هرتز، R_2 از ۶۰۰۰ تا ۱۰۰۰۰ هرتز و R_3 از ۱۴۰۰۰ تا ۱۶۰۰۰۰ هرتز در نظر گرفته شده

تلفظ حرف " ش " فرکانس‌های غیر هارمونیک در ناحیه ۳۰۰۰ تا ۱۰۰۰۰ تولید می‌کند. و ما از طریق سه مقایسه منطقه‌ای که در تصویر مشخص شده می‌توانیم حرف "ش " را تشخیص دهیم. بدین صورت که میانگین منطقه اول باید از میانگین منطقه‌ی دوم و همچنین از میانگین منطقه سوم بزرگ‌تر باشد.

تشخیص حرف ژ :

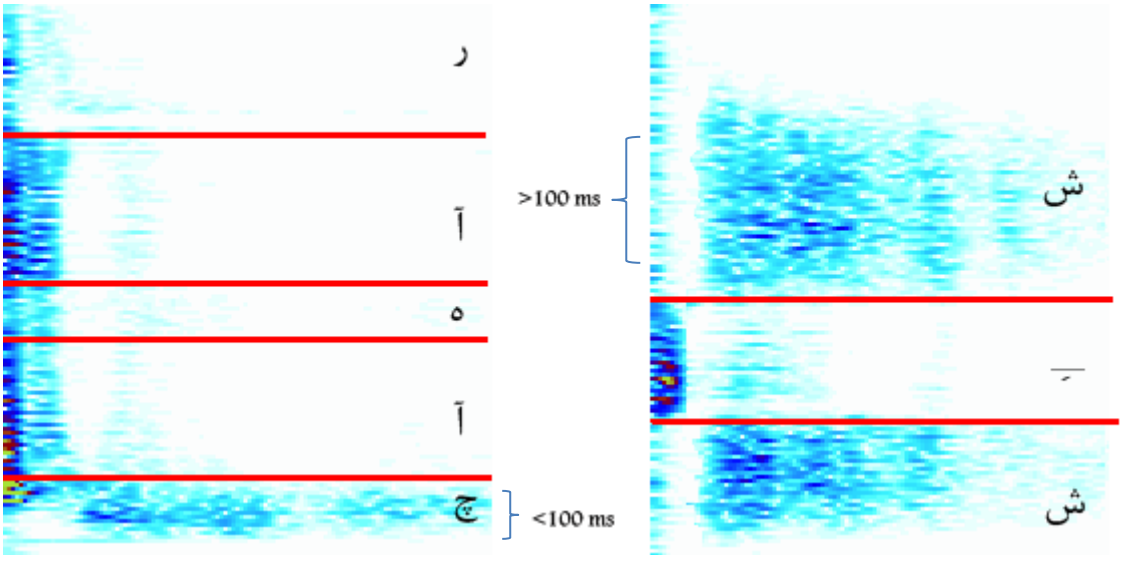
تشخیص حرف " ژ " دقیقاً مانند تشخیص حرف " ش " می‌باشد و نیازی به کد نویسی نمی‌باشد. و تمایز بین آن‌ها از فرکانس‌های هارمونیک آن‌هاست. چون اگر دقت کنید برای تلفظ حرف " ژ " از حنجره صدای هارمونیک نیز خارج می‌کنیم و برای پیدا کردن اینکه در قسمت هارمونیک صدا وجود دارد یا خیر می‌توانیم یک فیلتر تحلیلگر نویز بین فرکانس‌های پایین قرار دهیم.



شکل ۳.۲۱ اسپکتروگرام حرف "ژ" می‌باشد که استخراج ویژگی‌های آن همانند حرف "ش" می‌باشد با این تفاوت که در فرکانس‌های پایین آن فرکانس‌های سازنده موجود می‌باشد

تشخیص حرف چ :

تشخیص حرف " چ " دقیقاً مانند تشخیص حرف " ش " می‌باشد و نیازی به کد نویسی نمی‌باشد. تمایز بین آن‌ها از طریق مدت زمان وقوع آن‌ها در نمودار اسپکتروم صورت می‌پذیرد. اگر دقت کنید حرف "چ" نسبت به حرف "ش" بسیار کوتاه‌تر بیان می‌شود.



شکل ۳.۲۲ اسپکتروگرام تلفظ اعداد "شِش" در شمت راست و "چهار" در سمت چپ می‌باشد. اسپکتروگرام حرف "چ" مانند حرف "ش" می‌باشد با این تفاوت که حرف "ش" زمان طولانی‌تری نسبت به حرف "چ" تلفظ می‌شود.

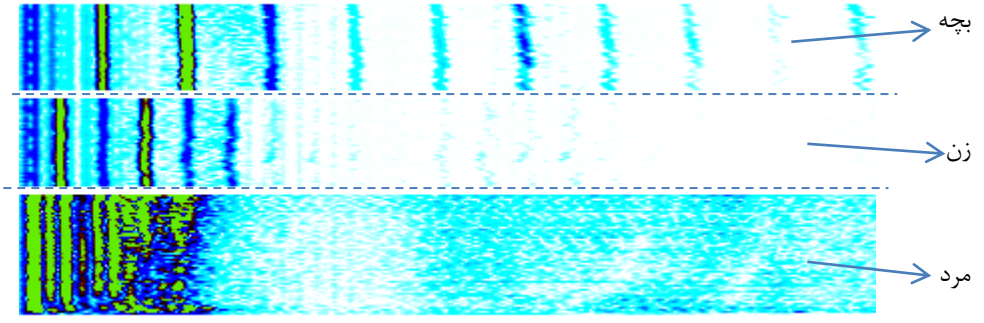
تشخیص حرف ج :

تشخیص حرف ″ ج ″ دقیقاً مانند تشخیص حرف ″ ژ ″ می‌باشد و نیازی به کد نویسی نمی‌باشد. تمایز بین آن‌ها از طریق مدت زمان وقوع آن‌ها در نمودار اسپکتروگرام صورت می‌پذیرد. اگر دقت کنید حرف ج نسبت به حرف ژ بسیار کوتاه‌تر بیان می‌شود و مانند حرف ″ ژ ″ دارای فرکانس‌های هارمونیک می‌باشد. البته برای شناسایی آن کمی دقت بیشتری نیاز است چون قسمت هارمونیک در اکثر موارد کمی زودتر نمایان می‌شود.

البته ویژگی‌های بهتری برای تشخیص وجود دارد که در زمان کوتاه مجبور به انتخاب این نواحی شدم. ضمناً برای انتخاب ناحیه‌ها باید نمونه‌های مختلفی از طریق انواع مختلف میکروفون، محیط‌های متفاوت و صداهای افراد مختلف بررسی شوند.

تشخیص زیر و بمی صدای گوینده

جهت تحلیل زیر و بمی صدا نیاز به اطلاعات بیشتری داریم. فرکانس‌های نمونه‌های ۱۰۲۴ تایی بسیار محدود می‌باشند و اطلاعات فرکانسی کافی جهت انجام این عمل به ما نمی‌دهند. به همین دلیل مجبوریم از اسپکتروم باند باریک استفاده کنیم و برای این منظور از دسته نمونه‌های ۸۱۹۲ تایی استفاده می‌کنیم. در زیر نمونه صدای تلفظ حرف آ از یک مرد، زن و بچه را در قالب نمودار اسپکتروگرام باند باریک مشاهده نمایید.



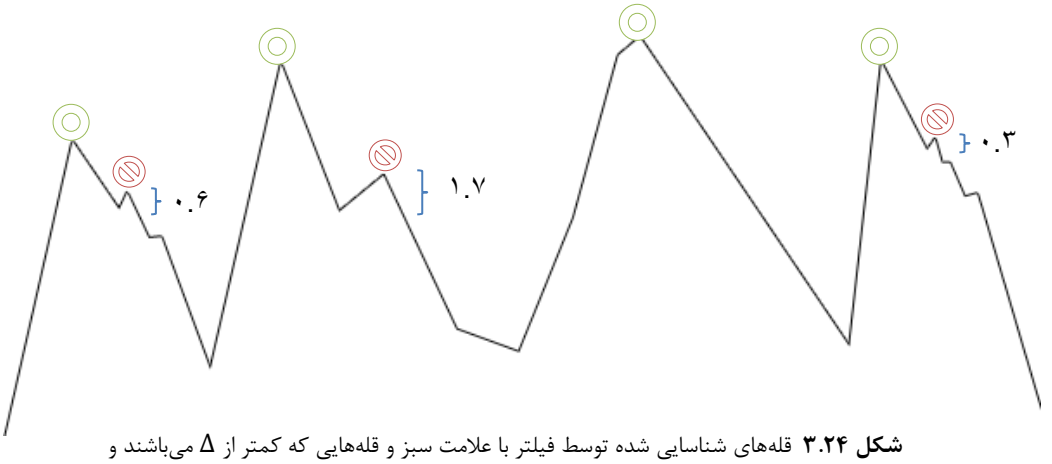
شکل ۳.۲۳ اسپکتروگرام تلفظ حرف آ توسط یک زن، مرد و بچه

احتمالاً شما نیز با مشاهده نمودار فوق راه حلی به ذهنتان خطور کرده. بله فاصله بین تمام خطوط هارمونیک موجود در نمودار دقیقاً برابر است! اگر بتوانیم فاصله بین خطوط را بدست آوریم می‌توانیم زیر و

بمی صدای شخص صحبت کننده را بفهمیم و جهت شناسایی بهتر کلمات برای این سه دسته پردازش‌های متفاوت انجام دهیم.

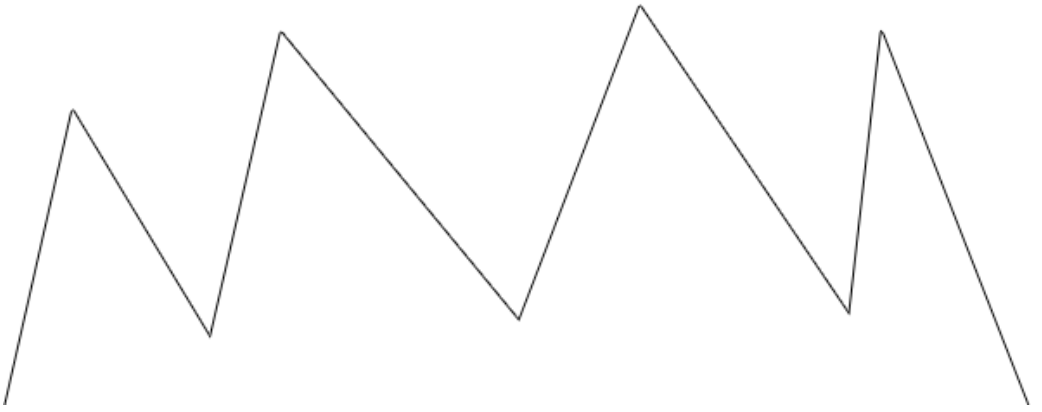
برای این منظور فیلتر نرمال سازی جهت نرمال نمودن خطوط هارمونیک طراحی نمودم که خروجی آن برای خودم نیز بسیار جالب بود. که در ادامه به شرح این فیلتر می‌پردازیم.

هدف از این فیلتر حذف نقاط اضافه روی نمودار اسپکتروم می‌باشد. به طوری که اگر تغییرات بلندی فرکانس کوچک‌تر از دلتا بود آن فرکانس را حذف می‌کند. و همچنین خطوط پهن را تنها به یک نقطه تبدیل می‌کند. به این صورت که قله‌ها را تشخیص داده و دامنه‌های قله را کاملاً حذف می‌کند(صفر می‌کند). و اگر قله از دلتا کوچک‌تر بود خود قله نیز حذف می‌شود. نمودار اسپکتروم زیر را در نظر بگیرید : $\Delta = ۲$



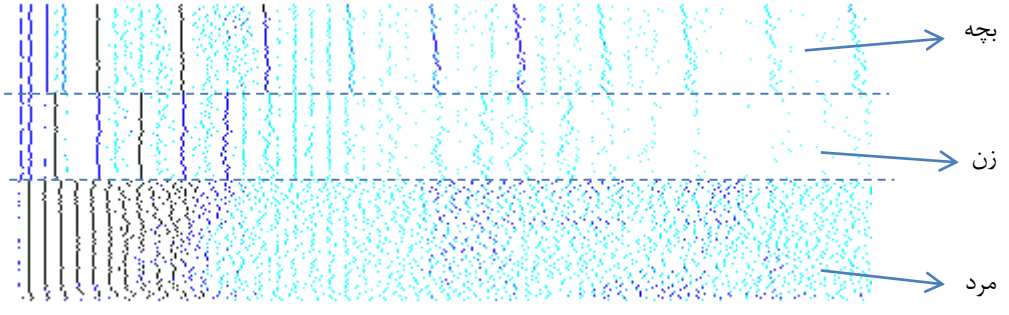
شکل ۳.۲۴ قله‌های شناسایی شده توسط فیلتر با علامت سبز و قله‌هایی که کمتر از Δ می‌باشند و باید حذف شوند با رنگ قرمز به نمایش در آمده است

خروجی این فیلتر :



شکل ۳.۲۵ خروجی فیلتر نرمال کننده اسپکتروم

در زیر نمونه صدای تلفظ حرف آ از یک مرد، زن و بچه را در قالب نمودار اسپکتروگرام باند باریک نرمال شده مشاهده نمایید.



شکل ۳.۲۶ اسپکتروگرام باند باریک نرمال شده تلفظ حرف آ توسط یک زن، مرد و بچه

جهت اطلاعات بیشتر در مورد کدهای این فیلتر به بخش توضیح کدهای برنامه رجوع نمایید:

`SoundAnalysis\ Filters\ SpectrumNormalizerFilter.cs`

حالا جهت تشخیص فاصله خطوط هارمونیک کارمان بسیار راحت شده است. یک الگوریتم ساده می‌تواند به این صورت باشد که بزرگ‌ترین مقدار هر ردیف را بدست آوریم، سپس از ابتدا شروع به پیمایش کنیم و در صورتی که نقطه هارمونیک با بیشترین مقدار کمتر از سه برابر پایین‌تر بود یعنی به نقطه هارمونیک بعدی رسیدیم. و بیشترین مقدار را مقدار نقطه هارمونیک جدید در نظر گرفته و این کار را ادامه می‌دهیم تا به حد

اکثر ۵ نقطه هارمونیک یافت کنیم. نقاط بسیار کم‌رنگ که مد نظر مان نمی‌باشد بسیار بیشتر از ۳ برابر کمتر می‌باشند. البته انشا الله در بیشتر موارد جواب خواهد داد. ولی احتمال اشتباه نیز هست که با رای اکثریت حل خواهد شد. برای نتیجه بهتر، خوب است که ضریب چند برابر بودن را بر حسب بزرگ‌ترین مقدار متفاوت در نظر بگیریم. چون اگر عدد بسیار بزرگی بدست آوریم احتمال اینکه نقطه هارمونیک بعدی بیش از ۳ برابر باشد بسیار زیاد است.

جهت اطلاعات بیشتر در مورد کدهای این فیلتر به بخش توضیح کدهای برنامه رجوع نمایید:
`SoundAnalysis\ Filters\ TempoAnalyzerFilter.cs`

برنامه آزمایشگاه صوت

علاوه بر پروژه تشخیص پایان نامه شامل یک پروژه آزمایشگاه صوت می‌باشد که از جمله امکانات این پروژه از قرار زیر می‌باشد :

- رویت اسپکتروگرام به صورت زنده با طیف رنگ‌های مختلف
- قابلیت ذخیره سازی و بازیابی اسپکتروگرام
- رویت فرکانس‌ها و بلندی با حرکت ماوس بر روی صفحه
- قابلیت رویت کل داده‌های اسپکتروم با کلیک بر روی هر ردیف
- تنظیم کنتراست و بزرگنمایی
- تعیین تعداد نمونه‌های صوتی ورودی اسپکتروگرام
- نمایش نرمال شده اسپکتروگرام
- نمایش اسپکتروم زنده

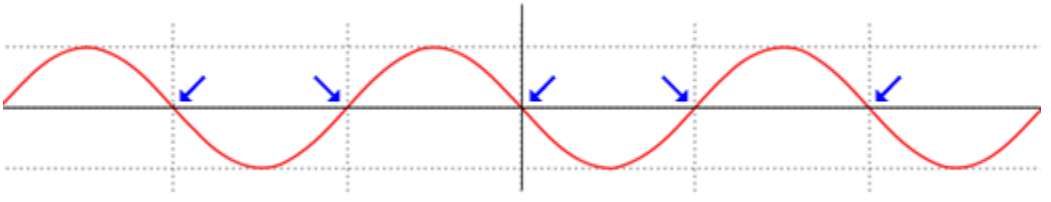
البته احتمالاً در این پروژه به باگ‌های متعددی برخورد خواهید کرد. کدهای این نرم افزار به طور کامل در اختیارتان قرار خواهد گرفت و شرح آن نیز در قسمت پروژه `SpectogramLabApp` به تفصیل بیان خواهد شد.

- ■ توضیحات این قسمت هنوز تکمیل نشده

قطعه بندی کلمات (Segmentation)

بخش بندی کلمات تلفظ شده و مشخص نمودن محدوده حروف در یک کلمه، به قطعه بندی معروف است. قطعه بندی از اهمیت بسیار ویژه‌ای در شناسایی کلمات برخوردار می‌باشد. هر چقدر قطعه بندی دقیق‌تری انجام دهیم شناسایی کلمات دقیق‌تر خواهد بود. به همین دلیل اصولاً برای نتیجه بهتر قطعه بندی از چندین تکنیک به صورت ترکیبی استفاده می‌کنند. مهم‌ترین تکنیک در قطعه بندی، محاسبه نرخ گذر از صفر (Zero-crossing rate) یا به اختصار ZCR می‌باشد که به وفور در قطعه بندی از آن استفاده می‌شود.

محاسبه نرخ گذر از صفر مستقیماً بر روی نمونه‌های صوتی قابل انجام می‌باشد. بدین‌صورت که هنگام رفت و بازگشت یک موج وقتی از صفر عبور می‌کند آن را ثبت می‌کنیم.



شکل ۳.۲۷ گذر از صفر یا Zero-crossing با فلش آبی مشخص شده است

برای محاسبه نرخ گذر از صفر ابتدا نمونه های ورودی را به اندازه‌های کوچک‌تر تقسیم بندی می‌کنیم. مثلاً ۱۰۲۴ (در این پروژه حدود ۱۰ میلی ثانیه می‌باشد). تعداد گذر از صفر را محاسبه می‌کنیم و با تعداد گذر از صفر پنجره بعدی (۱۰۲۴ نمونه بعدی) مقایسه می‌کنیم. اگر تفاوت از حد مشخص شده تجاوز کرد احتمالاً شروع کلمه دیگر می‌باشد.

همان طور که قبلاً گفتم ZCR به تنهایی نتیجه عالی به ما نمی‌دهد. حتماً باید با الگوریتم‌های دیگر به کار گرفته شود تا به نتیجه مورد قبول برسیم.

تشخیص کلمه

- توضیحات این قسمت هنوز تکمیل نشده

۴ راهنمای استفاده از کدهای برنامه

پروژه کتابخانه‌ای دریافت اصوات – SoundCapture

جهت دریافت نمونه‌های صوتی روش‌های گوناگونی وجود دارد که ما در پروژه‌های پایان نامه از DirectX استفاده نمودیم. این پروژه یک کتابخانه (DLL) می‌باشد که جهت سهولت در استفاده از DirectSound در برنامه‌هایمان از آن استفاده می‌کنیم.

هر چند این برنامه از کدهای اندکی تشکیل شده، به دلیل اینکه هدف اصلی این پروژه تحلیل صوت است و نه ارتباط با کارت صدا نیازی به تشریح بسیار دقیق این پروژه نمی‌باشد. به هر حال در ادامه به شرح مختصری از این پروژه می‌پردازیم.

تمام هدفمان از ایجاد این پروژه در یافت نمونه‌های صوتی می‌باشد. از طریق کلاس CaptureBuffer موجود در فضای کاری Microsoft.DirectX.DirectSound به انتظارِ رسیدن نمونه‌های صوتی می‌نشینیم تا نمونه‌های صوتی جدید را دریافت و به ما تحویل دهد. ممکن است در رایانه‌مان چندین کارت صدا وجود داشته باشد که ما کارت صدای پیش فرض را برای انجام این عمل انتخاب نمودیم.
تمام کدها جهت دریافت نمونه‌های صوتی در قسمت SoundCapture/SoundCaptureDevice.cs موجود در کلاس ابسترکت SoundCaptureBase می‌باشد. کلیه فرایند دریافت را در متد Start قرار دادیم. و پس از دریافت نمونه‌های صوتی متد ProcessData موجود در همین کلاس را فراخوانی خواهیم نمود. البته این متد وجود خارجی ندارد و به صورت ابسترکت می‌باشد. جهت استفاده از این پروژه کافی است کلاس جدیدی درست نماییم و از امکانات کلاس SoundCaptureBase ارث بری نماییم و تابع ProcessData را مجدداً تعریف نماییم. به هر حال برای برنامه نویسان مبتدی که با برنامه نویسی شی گرا آشنایی چندانی ندارند نیازی به درک دقیق این فرایند نمی‌باشد.

با وجود این پروژه، دیگر جهت دریافت نمونه‌های صوتی نیازی به کد نویسی فرایندهای پیچیده نداریم. در ادامه برنامه‌ای ساده می‌نویسیم که نمونه‌های صوتی را دریافت می‌کند.

۱- جهت انجام این کار پروژه‌ای ایجاد می‌نماییم که از DLL پروژه SoundCapture استفاده می‌نماید. (این DLL را از فهرست مربوطه به پروژه اضافه کنید)

۲- کلاس جدید می‌سازیم که از SoundCaptureBase ارث بری کند و تابع ProcessData را نیز برای این کلاس پیاده سازی می‌کنیم.

```
public class SoundInput : SoundCaptureBase
    {

        protected override void ProcessData(short[] data)
        {
            // در این قسمت نمونه‌ها را دریافت می‌کنیم

        }

    }
```

۳- جهت استفاده از برنامه یک شی از کلاس SoundInput تعریف می‌کنیم.

```
SoundInput input = new SoundInput();
```

۴- با فراخوانی متد input.Start() فرایند دریافت نمونه‌ها آغاز گشته و داده‌ها از طریق متود ProcessData در اختیارمان خواهد گرفت. و ما می‌توانیم نمونه‌ها را از طریق پارامتر data دریافت نماییم. و در انتهای برنامه جهت بازگرداندن منابع استفاده شده به سیستم عامل حتماً متود Close را فراخوانی نمایید.

آموزش ویدئویی نحوه ایجاد یک پروژه ساده جهت دریافت نمونه‌های صوتی در فهرست زیر همراه با سی دی پروژه موجود می‌باشد.

/Video/Projects/SimpleAudioProject.wmv

پروژه کتابخانه‌ای تحلیل – SoundAnalysis

این پروژه قلب پردازش‌ها و تحلیل‌های صوتی است که شامل سه قسمت اصلی فیلتر ها، عملیات تبدیل فوریه و تشخیص تقسیم می‌شود. در ادامه به تشریح هر یک می‌پردازیم.

فیلتر ها

در این پروژه فیلترها کلاس‌هایی می‌باشند که روی نمونه‌های صوتی و یا داده‌های فرکانس مربوط به نمونه‌ها پردازش انجام می‌دهند. برخی از فیلترها داده‌های فرکانس یا نمونه‌ها را تغییر داده و برخی دیگر این داده‌ها را تحلیل می‌نمایند و می‌توانیم نتیجه تحلیل را از طریق مشخصه‌های کلاس فیلتر بدست آوریم.
تمام کلاس‌های فیلتر از رابط `IFreqFilter` تبعیت می‌کنند که کلاس فیلتر را مجبور می‌کند تابع `ProcessData` را با قالب پارامترهای زیر پیاده سازی کند.

```
void ProcessData( double[] specData, double[] samples )
```

پارامتر `samples` نمونه‌های صوتی می‌باشند و `specData` نیز فرکانس‌های مربوط به نمونه‌ها است (اسپکتروم) و طول هر دو آرایه یکسان می‌باشد.
برخی فیلترها به هردو آیتم نیاز دارند و برخی دیگر به یکی از آن‌ها. ولی همچنان هر دو پارامتر را دریافت خواهند نمود. خواه از آن استفاده کنند یا خیر. به عنوان مثال فیلتر پنجره فقط به داده‌های نمونه‌ها نیاز دارد و روی آن‌ها تغییرات ایجاد می‌کند. و هیچ اطلاعات تحلیلی به ما نمی‌دهد ولی فیلتر **تحلیل نویز** اطلاعات سطح نویز محیط را در اختیارمان قرار خواهد داد ولی داده‌های نمونه را تغییر نمی‌دهد.

فیلترها در فهرست `Filters` قرار دارند و در ادامه به معرفی بیشتر فیلترها می‌پردازیم.

فیلتر تحلیل شدت صوت

نام کلاس فیلتر : `IntensityAnalyzerFilter`
فضای کاری : `SoundAnalysis.Filters`
مسیر : `SoundAnalysis\Filters\IntensityAnalyzerFilter.cs`

شرح فیلتر

این فیلتر با تحلیل بر روی نمونه‌های صوتی سطح نویز محیط را به ما باز می‌گرداند. به این صورت که از کل نمونه‌ها میانگین گرفته و در طول زمان کم‌ترین میانگین را همیشه در خود ذخیره می‌کند. به

دلیل اینکه احتمال بیشتر شدن سطح نویز در زمان وجود دارد، حدود هر ۵ ثانیه مجدداً به محاسبه حد اقل نویز محیط می‌پردازد. وقتی نمونه‌های جدید تحویل این فیلتر داده می‌شود ابتدا میانگین آن را محاسبه نموده و بررسی می‌نماید که آیا از آخرین سطح نویز بدست آمده کمتر است یا خیر، در صورت کمتر بودن آن را به عنوان حد اقل در نظر می‌گیرد. جهت محکم کاری انتظار دارد سطح نویز چندین بار از حد اقل کمتر شود و بعد به عنوان حد اقل جدید شناسایی شود.

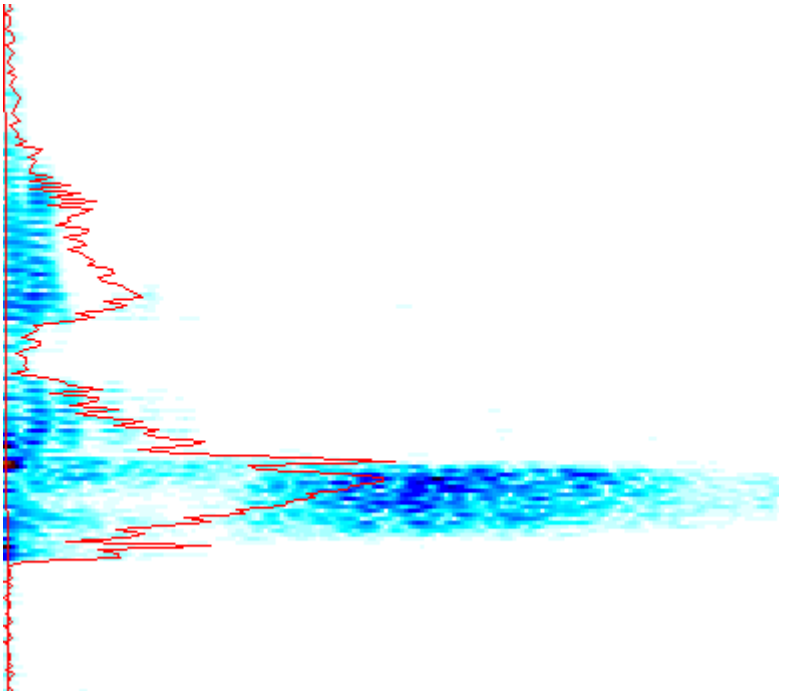
یک مشخصه مهم این فیلتر `IsSpeaking` می‌باشد که اگر کاربر در حال مکالمه باشد مقدار صحیح و در غیر این صورت مقدار غلط را باز می‌گرداند. و بدین‌صورت عمل می‌نماید که در صورتی که اگر مقدار صدای جدید به نسبت خاصی بیشتر شود یعنی مکالمه صورت گرفته. حساسیت تحریک سنجش صحبت را می‌توان از طریق سازنده کلاس مشخص نمود که عدد ۲.۵ به عنوان حساسیت پیش‌فرض مشخص شده است. سطح نویز را نیز از طریق مشخصه `NoiseLevel` می‌توان یدست آورد. ضمناً با آرشیو نمودن مشخصه `Avg` می‌توانید انرژی بکار رفته شده توسط صحبت کننده را جهت پردازش جدا سازی کلمه بکار برید. جهت بدست آوردن میانگین از قدر مطلق مقادیر استفاده نمودیم تا مقدار موج منفی را به مثبت تبدیل کند.

کد نمونه

ابتدا فیلتر را در قسمت فیلدهای کلاسی که می‌خواهید از ان استفاده کنید تعریف کنید چون نیاز دارد اطلاعات تحلیلی را در خود نگه داشته باشد. سپس برای اجرا ، متود `ProcessData` را همراه با آرگومان نمونه های جدید دریافت شده فراخوانی می کنیم.

```
intensityFilter.ProcessData(null, samplesData);
if (intensityFilter.IsSpeaking)
       lblSpeacking.Text = "در حال صحبت";
```

در تصویر ۴.۱ خروجی این فیلتر را از طریق برنامه آزمایشگاه مشاهده نمایید.



شکل ۴.۱ در این تصویر خطوط آبی توسط فیلتر تحلیلگر شدت صوت محاسبه شده است.

فیلتر تحلیل نویز محیط

نام کلاس فیلتر : `NoiseAnalyzerFilter`

فضای کاری : `SoundAnalysis.Filters`

مسیر : `SoundAnalysis\Filters\NoiseAnalyzerFilter.cs`

شرح فیلتر

نحوه عملکرد و کاربرد این فیلتر نیز مانند فیلتر تحلیل شدت صوت می‌باشد با این تفاوت که در این فیلتر سطح نویز با تحلیل بر روی داده‌های فرکانس به دست می‌آید. یعنی جمع بزرگی فرکانس را جهت انجام میانگین بکار می‌بریم. دلیل طراحی این فیلتر این بود که در برخی موارد نیاز داشتیم بفهمیم آیا کاربر صدای هارمونیک از خود تلفظ می‌کند یا خیر. برای این منظور در سازنده این کلاس می‌توانیم مشخص کنیم این تحلیل مربوط به چه رنجی از فرکانس‌ها باشد.

کد نمونه

تعریف شی کلاس باید در مکان فیلدها قرار بگیرد چون نیاز به ماندگاری اطلاعات داخلی دارد. در زیر بازه فرکانس از ۱۰۰ تا ۴۰۰ تعریف شده است. (البته مقدار به صورت bin می‌باشد و باید فرکانس مورد نظرمان را به بین تبدیل کنیم)

```
NoiseAnalyserFilter filter = new NoiseAnalyserFilter(100, 400);
```

استفاده از شی

```
filter.ProcessData(specData, null);
if (filter.IsSpeaking)
	lblSpeacking.Text = "حرف صدا دار";
```

فیلتر کاهنده نویز

نام کلاس فیلتر : NoiseReductionFilter

فضای کاری : SoundAnalysis.Filters

مسیر : SoundAnalysis\Filters\NoiseReductionFilter.cs

شرح فیلتر

این فیلتر یک فیلتر بسیار مبتدی برای کاهش صداهای مزاحم محیط می‌باشد و بر روی داده‌های اسپکتروم تغییرات ایجاد می‌نماید. به این صورت که در صورتی مکالمه انجام نشود از هر فرکانس میانگینی تهیه نموده و آن را صدای مزاحم فرض می‌کند و در هنگام مکالمه این مقدار نویز هر فرکانس را از فرکانس‌های هنگام مکالمه کم می‌کند. ظاهراً عمل می‌کند ولی ایرادهایی دارد که باید حل شود. به طور مثال هنگام مکالمه برخی از نویزها به دلیل انرژی بیشتر مکالمه حذف خواهند شد. و خیلی از مسائل دیگر که در این فیلتر رعایت نشده. و به دلیل برخی مشکلاتی که با استفاده از آن پدیدار شده بود دیگر از آن استفاده ننمودم و بهتر است به طور ویژه اصلاح شود و در پروژه بکار گرفته شود.چون وجود یک فیلتر کاهنده نویز خوب در تشخیص صدا حیاطی است.

فیلتر نرمال کننده اسپکتروم

نام کلاس فیلتر : SpectrumNormalizerFilter

فضای کاری : SoundAnalysis.Filters

مسیر : SoundAnalysis\Filters\SpectrumNormalizerFilter.cs

شرح فیلتر

نرمال سازی اسپکتروم نیز یکی از مباحث مهم در تشخیص صدا می‌باشد. البته این فیلتر فقط در اسپکتروم‌های باند باریک قابل استفاده می‌باشد. چون جزئیات زیادی از فرکانس در آن قرار گرفته و ما نیاز داریم تا اطلاعات مزاحم را حذف کنیم. اصولاً جهت نرمال سازی اسپکتروم از فرایند پیچیده آماری استفاده می‌شود ولی ما برای این منظور الگوریتم بسیار ساده‌ای طراحی نمودیم که می‌توانید شرح عملکرد آن را در بخش تشخیص صوت مشاهده نمایید. به طور خلاصه این فیلتر در نمودار اسپکتروم به دنبال خطوط هارمونیک گشته و اطلاعات نسبتاً کم ارزش‌تر را حذف می‌کند. جهت این عمل به دنبال قله‌های بزرگ می‌گردد و سر راه در صورتی که به قله‌های به طول کوچک‌تر از دلتا برخورد نماید آن‌ها را حذف نموده و نوک قله‌ها را به عنوان خروجی در نظر می‌گیرد.

ما در تشخیص زیری و بمی از این فیلتر کمک گرفته‌ایم.

کد نمونه

```
SpectrumNormalizerFilter filter = new SpectrumNormalizerFilter();
filter.ProcessData(specData, null);
```

فیلتر تحلیلگر زیری و بمی صدای گوینده

نام کلاس فیلتر : TempoAnalyzerFilter

فضای کاری : SoundAnalysis.Filters

مسیر : SoundAnalysis\Filters\TempoAnalyzerFilter.cs

شرح فیلتر

قبل از استفاده این فیلتر ابتدا داده‌ها را با استفاده از فیلتر نرمال کننده نرمال نموده و سپس تحویل فیلتر تحلیل گر زیر و بمی می‌دهیم تا درجه زیری و بمی را تشخیص دهد. درجه زیری و بمی در

مشخصه شمارشی Character قرار می‌گیرید که می‌تواند یکی از مقادیر Male، Female و Child باشد. برای این منظور از فاصله نقاط هارمونیک استفاده نمودیم. هرچه فاصله‌های به اندازه مساوی زیادتر باشد صدا زیرتر و هرچه کوتاه‌تر صدا بم‌تر می‌باشد. حد فاصله ۶ تا ۲۱ را مرد در نظر گرفتیم، ۲۲ تا ۳۰ را زنان و بیش از ۲۰ صدای بچه در نظر گرفته خواهد شد. برای شرح بیشتر به بخش تشخیص صدا و تشخیص زیری و بمی رجوع نمایید.

کد نمونه

```
SpectrumNormalizerFilter normalizerfilter =
      new SpectrumNormalizerFilter();
TempoAnalyserFilter tempoFilter =
      new TempoAnalyserFilter();

normalizerfilter.ProcessData(specData, null);
tempoFilter.ProcessData(specData, null);
```

تشخیص

ما برای تشخیص هر حرف یک کلاس ساده در نظر گرفتیم که فعلاً کاری بیش از عمل میانگین و مقایسه منطقه‌های مختلف اسپکتروم انجام نمی‌دهند. در این کلاس‌ها متود Detect با دریافت داده‌های اسپکتروم یک عدد به عنوان خروجی باز می‌گرداند که بیانگر احتمال وجود آن حرف می‌باشد. البته در حال حاضر احتمال وجود یا صفر و یا ۱ می‌باشد. اگر بخواهیم در آینده روش تشخیص را به الگوریتم کارا تری تغییر دهیم باید عددی اعشاری مابین ۰ و ۱ بازگرداند که ۰ بیانگر عدم وجود و ۱ بیانگر وجود حتمی و اعداد مابین بیانگر تخمین وجود می‌باشد.

این کلاس‌ها در فضای کاری SoundAnalysis.Recognition.Phoneme تعریف شده‌اند. که در ادامه به شرح عملکرد کلاس تشخیص حرف " آ " می‌پردازیم و به دلیل اینکه مابقی حروف نیز عملکردی مشابه دارند از تشریح آنان اجتناب نمودم. جهت آشنایی با نحوه شناسایی کلیه حروف به بخش تشخیص رجوع نمایید.

تشریح کدهای کلاس شناسایی حرف " آ " :

نام کلاس: PhonemeDetector_AH

فضای کاری : SoundAnalysis.Recognition.Phoneme

مسیر : SoundAnalysis\Recognition\Phoneme\PhonemeDetector.cs

کدهای کلاس

```
public class PhonemeDetector_AH : PhonemeDetectorBase
 {
     // سازنده کلاس
     public PhonemeDetector_AH(PhonemeDetector detector)
         : base(detector)
     {
         // آغاز و پایان منطقه اول
         beg1 = (int)(842 * FrequencyScale);
         end1 = (int)(1200 * FrequencyScale);

         // آغاز و پایان منطقه دوم
         beg2 = (int)(1400 * FrequencyScale);
         end2 = (int)(1600 * FrequencyScale);

     }

     // متود تشخیص
     public override double Detect(double[] fftSamples)
     {
         sum1 = sum2 = 0;

         // مجموع منطقه اول
         for (i = beg1; i < end1; i++)
             sum1 += fftSamples[i];

         // مجموع منطقه دوم
         for (i = beg2; i < end2; i++)
             sum2 += fftSamples[i];

         // محاسبه میانگین مناطق
```

```
            avg1 = sum1 / (end1 - beg1);
            avg2 = sum2 / (end2 - beg2);

            // نسبت دو منطقه
            rel1 = sum1 / sum2;


            if (rel1 > 4)
                return 1;
            else
                return 0;
        }
    }
```

کلاس ابسترکت PhonemeDetectorBase یک کلاس پایه مخصوص کلاس‌های تشخیص حروف می‌باشد که شامل یک سری متغیرهای عمومی جهت تسهیل و خوانایی برنامه می‌باشد. کلیه کلاس‌های تشخیص حروف از این کلاس پایه ارث می‌برند. کلاس پایه شامل متغیرهایی جهت ذخیره سازی رنج فرکانس، عملیات جمع و میانگین و ذخیره سازی نسبت میانگین می‌باشد. همچنین کلاس پایه دارای یک متد سازنده می‌باشد که پارامتری از نوع PhonemeDetector با نام detector دریافت نموده و در فیلد _detector ذخیره می‌کند. کلاس PhonemeDetector نیز اطلاعاتی نظیر سطح نویز، زیر و بمی، ضریب فرکانس جهت تبدیل به فرکانس به bin در خود دارد و کلیه کلاس‌های تشخیص به این کلاس جهت دریافت این اطلاعات نیاز دارند.

در کدهای فوق ابتدا در سازنده شروع و پایان فرکانس مناطق مختلف را در فیلدهای begN و endN ذخیره می‌کنیم. در تشخیص حرف " آ " از دو منطقه که منطقه نخست از فرکانس ۸۴۲ تا ۱۲۰۰ و منطقه دوم از فرکانس ۱۴۰۰ تا فرکانس ۱۶۰۰ می‌باشد استفاده شده. جهت کار با داده‌های اسپکتروم بجای فرکانس باید از اندیس آرایه یا همان fft bin استفاده کنیم بنابراین در سازنده قبل از اختصاص به فیلدها در ضریب نسبت فرکانس به fft bin ضرب می‌کنیم.

متد Detect شامل یک پارامتر می‌باشد که از طریق آن داده‌ای اسپکتروم را دریافت نموده و عددی اعشاری بین ۰ تا ۱ نمایانگر احتمال تلفظ حرف " آ " را به عنوان خروجی مشخص می‌نماید. که ما در این کلاس برای سادگی کار فقط ۰ به عنوان عدم وجود و ۱ به عنوان وجود احتمال در نظر گرفته شده.

جهت تخمین ابتدا مجموع مناطق را در متغیر sumN قرار داده و سپس میانگین آنان را درون avgN و نسبت بین مناطق در متغیر relN و در نهایت با بررسی نسبت احتمال وجود یا عدم احتمال وجود حرف " آ " مشخص می‌شود.

PhonemeDetector مسئولیت تشخیص حروف در یک سطح بالاتر را دارا می‌باشد. بدین صورت که درون خود از هر نوع کلاس تشخیص موجود در سطح پایین‌تر، شی دارد و توسط متود Detect عمل تشخیص را از طریق این اشیاء انجام می‌دهد. متود Detect از بیرون فراخوانی می‌شود و فراخوانی کننده موظف است مقادیر پارامترهای سطح نویز و سطح نویز در قسمت هارمونیک، سرعت صدا و داده‌های اسپکتروم را برای این متود مهیا سازد.

```
public virtual int Detect(double[] fftSamples, double tempo,
                                double noise,  double noiseVowel)
        {
            _noise = noise;           // سطح نویز
            _tempo = tempo;             // زیر و بمی
            _noiseVowel = noiseVowel;  // سطح نویز قسمت هارمونیک


             Inf.pVowel = tempo != 0? 1:0 ;
             Inf.pAH = _detector_AH.Detect(fftSamples);
             Inf.pEH = _detector_AA.Detect(fftSamples);
             Inf.pAA = _detector_EH.Detect(fftSamples);
             Inf.pEE = _detector_E.Detect(fftSamples);
             Inf.pS = _detector_S.Detect(fftSamples);
             Inf.pSH = _detector_SH.Detect(fftSamples);

             DetectedPhoneme = checker.Method1(Inf);

             return 0;
        }
```

همان‌طور که کدهای PhonemeDetector را مشاهده می‌نمایید نحوه عملکرد این کلاس بسیار آسان می‌باشد بدین‌صورت که اشیاء تشخیص را فراخوانی نموده و نتایج را در ساختار Inf ذخیره می‌نماید.

pVowel نیز احتمال صدا دار بودن حروف می‌باشد که ما برای سادگی کار از پارامتر سرعت استفاده نمودیم به طوری که اگر سرعت مخالف صفر بود یعنی حروف صدا دار است.

حال ما چندین متغیر احتمال وجود از حروف درون ساختار Inf داریم و یک روش جهت تشخیص نهایی این است که باید تشخیص دهیم احتمال رخداد کدام حرف بیشتر است. و روش بعدی می‌تواند این باشد که احتمال رخداد مجموعه‌ای از حروف احتمال برخی وجود برخی دیگر را بالا ببرد که ما فعلاً چون رنج مقادیر احتمالمان فقط بلی و خیر است از روش اول استفاده می‌کنیم. در نهایت باید حروف تشخیص داده شده را در متغیر شمارشی DetectedPhoneme قرار دهیم. کلاس کمکی PhonemeChecker با گرفتن این

احتمال‌ها به وسیله چند شرط ساده مشخص می‌کند که احتمال رخداد کدام حرف نزدیک‌تر است. Method1 در کلاس PhonemeChecker با گرفتن Inf این عمل را برایمان انجام می‌دهد.

پروژه برنامه تشخیص صدا – RecognizerApp

این پروژه ، برنامه‌ای تحت ویندوز می‌باشد حروف تلفظ شده و همچنین درجه زیری و بمی گوینده را به نمایش در می‌آورد. برای نوشتن این برنامه از دو پروژه‌ی کتابخانه‌ای که در بخش قبل شرح دادیم استفاده کردیم. یعنی پروژه‌های SoundCapture جهت دریافت نمونه‌های صوتی از کارت صدا و پروژه‌ی SoundAnalysis جهت انجام اعمال تحلیل بر روی سیگنال‌ها و فرکانس‌های صوتی.

آماده سازی پروژه جهت تشخیص صدا

پس از ایجاد پروژه طبق فرایند تشریح شده در نحوه عملکرد پروژه SoundCapture، کلاسی به نام SoundInput به صورت زیر جهت دریافت نمونه‌های صوتی ایجاد می‌کنیم.

```
using SoundCapture;
namespace RecognizerApp
{
    public delegate void SampleDataReceiverDelegate(short[] data);

    public class SoundInput : SoundCaptureBase
    {
        public event SampleDataReceiverDelegate FrequencyDetected = null;

        public SoundInput(SampleDataReceiverDelegate receiver)
        {
            SampleRate = 192000;
            FrequencyDetected += new SampleDataReceiverDelegate(receiver);
        }

        protected override void ProcessData(short[] data)
        {
            if (FrequencyDetected != null)
                FrequencyDetected( data);
        }
    }
}
```

ما قصد داریم سیگنال‌های صوتی را در قسمت فرم اصلی دریافت نماییم. برای ارتباط با این قسمت یک delegate از نوع SampleDataReceiverDelegate در این کلاس با نام FrequencyDetected را

در قالب رویداد تعریف نمودیم تا در فرم اصلی تابع مورد نظر را جهت دریافت نمونه‌ها به آن ربط دهیم. و هنگام دریافت از طریق SoundInput تابع موجود در فرم اصلی را از طریق FrequencyDetected به اجراء در آوریم.

حال تابع مورد نظر را سازگار با SampleDataReceiverDelegate (یک پارامتر و بدون خروجی) در فرم اصلی به صورت زیر تعریف می‌نماییم.

```
// جهت دریافت نمونه‌های صوتی از کارت صدا
private void NewInputSamplesArrivedEvent(short[] data)
{

}
```

جهت آغاز به کار مکانیزم دریافت، ابتدا یک شی از نوع SoundInput در قسمت فیلدهای فرم اصلی ایجاد نموده و در رویداد MainForm_Load تابع دریافت را به صورت زیر به آن نسبت می‌دهیم و با متد input.Start فرایند دریافت آغاز می‌شود.

:

```
private void MainForm_Load(object sender, EventArgs e)
    {
        // جهت دریافت نمونه‌های صوتی از کارت صدا
        input = new SoundInput(NewInputSamplesArrivedEvent);
        // آغاز به کار شنود و دریافت نمونه‌های صوتی
        input.Start();
```

ما پس از دریافت نمونه‌های صوتی نیاز داریم آن را به دو صورت به داده‌های اسپکتروم (دامنه فرکانس) تبدیل نماییم. نخست به صورت اسپکتروم باند باریک (نمونه‌های ۸۱۹۲ تایی) جهت تشخیص زیری و بمی صدا و دوم به صورت اسپکتروم باند پهن (نمونه‌های ۱۰۲۴ تایی) جهت تشخیص صدا.

نمودار فعالیت برنامه را در زیر مشاهده نمایید.

192000
نمونه در ثانیه دریافت میکنیم

رویداد دریافت نمونه های جدید

جهت تبدیل فوریه
بصورت باند باریک

تقسیم بندی به 2318

جهت خذف مشکل پخش
شدگی در تبدیل فوریه

اِعمال تابع پنجره

تبدیل فوریه

پس از تبدیل رویدادی جهت اتمام تبدیل
برای هر دسته 2918 تایی شامل اطلاعات
مربوط به نمونه ها و مربوط به فرکانسها

ارسال رویداد تبدیل 2918 تایی

این تقسیم بندی از بین نمونه های 2918
تایی فوق انجام می شود

تقسیم بندی 4201 تایی

اعمال تابع پنجره

تبدیل فوریه

ارسال رویداد تبدیل 4201 تایی

شکل ۵.۱ نمودار فعالیت برنامه پایه

ما عملیات تبدیل فوریه را جهت خوانایی بالاتر در کلاس استاتیک FrequencyHelper انجام می‌دهیم. این کلاس نمونه‌های دریافت شده از کارت صدا را گرفته، سپس به تعداد ۸۱۹۲ تایی تقسیم نموده و پس از اتمام عملیات تبدیل فوریه برای هر قسمت ۸۱۹۲ تایی رویدادی را جهت اطلاع به فرم اصلی ارسال می‌نماید. پس از ارسال هر رویداد داده‌های نمونه ۸۱۹۲ تایی را به ۸ تکه ۱۰۲۴ تایی تقسیم نموده و مجدداً پس از اتمام عملیات تبدیل فوریه بر روی هر یک، رویداد مربوط به آن را جهت اطلاع به فرم اصلی ارسال می‌نماید.

جهت دریافت این رویدادها دو تابع دیگر ایجاد می‌نماییم :

```
// دریافت فرکانس‌های بدست آمده از نمونه‌های ۱۰۲۴ تایی
// جهت تشخیص گفتار
private void New1024FourierFrequencyArrived(double[] data, double[]
samples)
{ . . . }

// دریافت فرکانس‌های بدست آمده از نمونه‌های ۸۱۹۲ تایی
// جهت زیر و بمی
private void New8192FourierFrequencyArrived(double[] data, double[]
samples)
{ . . . }
```

و در رویداد دریافت نمونه‌های صوتی کارت صدا، جهت انجام تبدیل فوریه و ارسال نتایج به دو تابع فوق به صورت زیر اقدام می‌نماییم :

```
// جهت دریافت نمونه‌های صوتی از کارت صدا
private void NewInputSamplesArrivedEvent(short[] data)
{
    // تبدیل نمونه‌های صوتی به فرکانس
    FrequencyHelper.ProcessForierTransfer(data,
        New1024FourierFrequencyArrived,
        New8192FourierFrequencyArrived);

}
```

تابع FrequencyHelper.ProcessForierTransfer با دریافت نمونه‌های صوتی آن را به قسمت‌های ۸۱۹۲ تایی و قسمت‌های ۸۱۹۲ تایی را به قسمت‌های ۱۰۲۴ تایی تقسیم می‌نماید و توابعی که به عنوان آرگومانهای دوم و سوم معرفی نمودیم را به اجرا در می‌آورد و نمونه‌ها و فرکانس‌های مربوط به آن نمونه‌ها را به دو پارامتر آنان ارسال می‌نماید.

در ادامه صرفاً با این دو تابع سروکار خواهیم داشت.

تشخیص صحبت یا سکوت

ابتدا قصد داریم صحبت گوینده را تشخیص دهیم. همان‌طور که در بخش‌های گذشته توضیح دادیم به راحتی می‌توانیم از فیلتر IntensityAnalyserFilter یا NoiseAnalyserFilter نماییم. فقط کافی است که این فیلتر را داخل تابع New8192FourierFrequencyArrived بکار ببریم. برای این منظور یک فیلتر در قسمت فیلدهای فرم به صورت زیر تعریف می‌کنیم :

```
IntensityAnalyserFilter noiseFilterSpeaking =
                            new  IntensityAnalyserFilter();

bool speaking=false;
```

و در قسمت رویدادها به راحتی از آن استفاده می‌کنیم :

```
private void New8192FourierFrequencyArrived(
              double[] data, double[] samples)
{
            noiseFilterSpeaking.ProcessData(data, samples);
            speaking = noiseFilterSpeaking.IsSpeaking;
}
```

تشخیص زیری و بمی

تشخیص زیری و بمی صدا با تلفظ حروف صدا دار مشخص می‌شود. پس باید بفهمیم که آیا حروف گوینده صدادار است یا خیر. برای این کار فقط کافی است اندازه صدا در قسمت فرکانس پایین تغییر چشمگیری داشته باشد که برای تشخیص آن می‌توانیم از یک فیلتر NoiseAnalyserFilter استفاده نماییم. بدین صورت که در قسمت فیلدها یک فیلتر تعریف نموده و مقادیر فرکانس را از ۱۰۰ هرتز تا ۱۴۰۰ هرتز در نظر می‌گیریم. البته در زیر این مقادیر را به fft bin برای نمونه‌های ورودی ۸۱۹۲ تبدیل نمودیم که همان ۶ تا ۴۳ می‌شود. و ضریب را ۱.۳ در نظر گرفتیم که اگر ۳۰ درصد افزایش صدا داشتیم تحریک شود.

```
NoiseAnalyserFilter noiseFilterVowelChecker =
                            new NoiseAnalyserFilter(6,44,1.3)
```

در مرحله بعد باید ببینیم اگر در این منطقه صحبت وجود داشت، اسپکتروم را نرمال کنیم و فاصله نقاط هارمونیک را در آوریم و این فاصله همان زیری و بمی صدا می‌باشد (برای اطلاعات دقیق‌تر به بخش تشخیص صدا رجوع نمایید و بخش توضیح کدهای فیلترها رجوع نمایید).

برای این منظور از دو فیلتر SpectrumNormalizerFilter و TempoAnalyserFilter استفاده می‌نماییم. این فیلترها را در قسمت فیلدهای فرم اصلی به صورت زیر تعریف می‌کنیم

```
SpectrumNormalizerFilter normalizerFilter =
                    new SpectrumNormalizerFilter();

TempoAnalyserFilter tempoFilter = new  TempoAnalyserFilter();
```

همچنین tempoFilter جهت تشخیص زیری و بمی در صورت سکوت نیاز دارد تا ریست شود. کدهای تشخیص زیری و بمی را در قسمت زیر مشاهده نمایید

```
private void New8192FourierFrequencyArrived(double[] data,
                          double[] samples)
        {
            noiseFilterSpeaking.ProcessData(data, samples);
            noiseFilterVowelChecker.ProcessData(data, samples);

            // اگر حرف صدادار تلفظ شده بود
            if (noiseFilterVowelChecker.IsSpeaking)
            {
                // اسپکتروم باید نرمال شود
                normalizerFilter.ProcessData(data, samples);
                // تشخیص زیری و بمی با استفاده از اسپکتروم نرمال شده
                tempoFilter.ProcessData(data, samples);
            }
            else
                // حرف، صدادار نیست و باید ریست شود
                tempoFilter.Reset();
        }
```

زیری و بمی در قالب یک شماره که در بخش‌های گذشته شرح دادیم درون tempoFilter.Tempo قرار می‌گیرد.

حال نوبت استفاده از توابع کتابخانه‌ای تشخیصمان می‌باشد. برای این و منظور از نیاز به اسپکتروم باند پهن داریم بنابراین از تابع New1024FourierFrequencyArrived استفاده می‌کنیم. همان‌طور که میدانید در این قسمت جهت تشخیص قصد داریم تا از کلاس PhonemeDetector استفاده نماییم. پس یک شی در قسمت فیلدها این نوع این کلاس به صورت زیر تعریف می‌نماییم :

```
PhonemeDetector phonemeDetector = new PhonemeDetector(1024, 192000);
```

این کلاس دو پارامتر برای سازنده‌اش نیاز دارد که با اولین پارامتر میگوییم که داده‌های اسپکتروم ۱۰۲۴ تایی می‌باشد و دومی نرخ نمونه‌های دریافت شده از کارت صدا در ثانیه می‌باشد. سپس به راحتی از طریق متد Detect می‌توانیم حرف تلفظ شده را شناسایی نماییم. در زیر می‌توانید کدهای مربوط به این قسمت را مشاهده نمایید :

```
private void New1024FourierFrequencyArrived(double[] data,
                        double[] samples)
        {

            if (noiseFilterSpeaking.IsSpeaking)
                phonemeDetector.Detect(
                        data,
                        tempoFilter.Tempo,
                        noiseFilterSpeaking.NoiseLevel,
                        noiseFilterVowelChecker.NoiseLevels);

        }
```

حال حرف تلفظ شده را می‌توانیم از طریق عضو شمارشی phonemeDetector.DetectedPhoneme دریافت نماییم.

ضمیمه ها

ضمیمه ۱ : پیش‌نیازهای نرم افزاری و سیستم عامل

ضمیمه ۲ : پیش‌نیازهای برنامه نویسی

ضمیمه ۳ : استفاده از برنامه آزمایشگاه SpectogramLabApp

دانشجویان عزیز در صورت داشتن هرگونه ابهام، اصلاحات، پیشنهادات و یا در صورت مقدور به اشتراک گذاری دانش و دست آورد های جدید می‌توانند با پست الکترونیکی بنده به نشانی زیر ارتباط برقرار نمایند :

https://github.com/ghominejad/VoiceRecognition

ghominejad@gmail.com

منابع

[1] Richard M. Warren, "Auditory Perception An Analysis and Synthesis", 2008

[2] Steven W. Smith, "Digital Signal Processing" 1999

[4] Josh Beggs and Dylan Thede "Designing Web Audio", 2001

[5] آزمایشگاه آنلاین فیزیک، موج ها

http://physics-animations.com/Physics/English/waves.htm

http://www.physicsclassroom.com/class/waves/

[6] Wikipedia : Window function, Formant, Fundamental frequency, Vocal tract, Spectrogram, Fft, Speech segmentation, Speech recognition, Zero crossing rate

به نام خدا

فرم دفاع دانشجویان کارشناسی ناپیوسته کامپیوتر – دانشگاه آزاد اسلامی واحد گرگان

پروژه دانشجو ..... دانشجوی رشته مهندسی تکنولوژی نرم افزار (کارشناسی ناپیوسته)

تحت عنوان .....

در تاریخ ..... دفاع گردید و با نمره ..... با موفقیت به اتمام رسید.

اسامی داوران [به ترتیب نام خانوادگی]

۱)

۲)

۳)

نام استاد راهنما :

- مهندس علی برومند
- مهندس محمد تقی خیر آبادی
- مهندس علیرضا مهینی
- مهندس شیده سرائیان
- مهندس حسن ابراهیمی
- مهندس امین بزازی
- مهندس حمیدرضا مهینی
- مهندس عسگری پریچهره