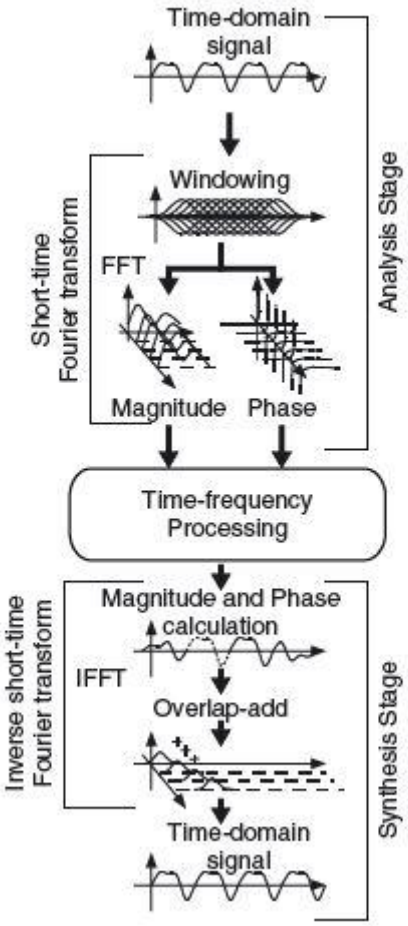
پروژه‌ی درس پردازش سیگنال‌های دیجیتال

بخش نخست: آشنایی با STFT و Phase vocoder

در این بخش توضیحی در مورد سازوکار STFT و به دنبال آن Phase vocoder می‌دهیم تا هدف پروژه و به دنبال آن درک پروژه روشن‌تر شود.

ایده‌ی اصلی STFT که مختصرشده‌ی Short Time Fourier Transform است، در ضرب‌کردن پنجره‌هایی با طول محدود در سیگنال ورودی، و سپس تبدیل فوریه گرفتن از این سیگنال‌ها نهفته است. احتمالا تا به حال با vocoder آشنا شده باشید؛ از عمده کاربردهای STFT در phase vocoder است که تعمیم ایده‌ی STFT برای تعداد زیادی پنجره در طول زمان است.



فرض کنید میخواهیم با سیگنال x و تک پنجره‌ی h این تبدیل را حساب‌کنیم:

$$X(n,k) = \sum_{m=-\infty}^{\infty} x(m)h(n-m)e^{-j\frac{2\pi mk}{N}} = |X(n,k)|e^{j\phi(n,k)}$$

البته با آگاهی نسبت به این نکته که طول پنجره محدود (N نقطه) است، محاسبه‌ی هرقسمت مانند محاسبه‌ی FFT سیگنال اصلی است که با ضرب در پنجره وزن‌دار شده‌است.

با دقت‌کردن به معادله‌ی بالا، می‌توانیم به تعبیر جالبی دست‌یابیم. اگر سیگنال h_k را به صورت زیر تعریف‌کنیم:

$$h_k(n) = h(n)e^{j\frac{2\pi nk}{N}}, k = 0,1,\dots,N-1 \rightarrow H_k\left(e^{j\omega}\right) = H\left(e^{j(\omega-\omega_k)}\right), \omega_k = \frac{2\pi k}{N}$$

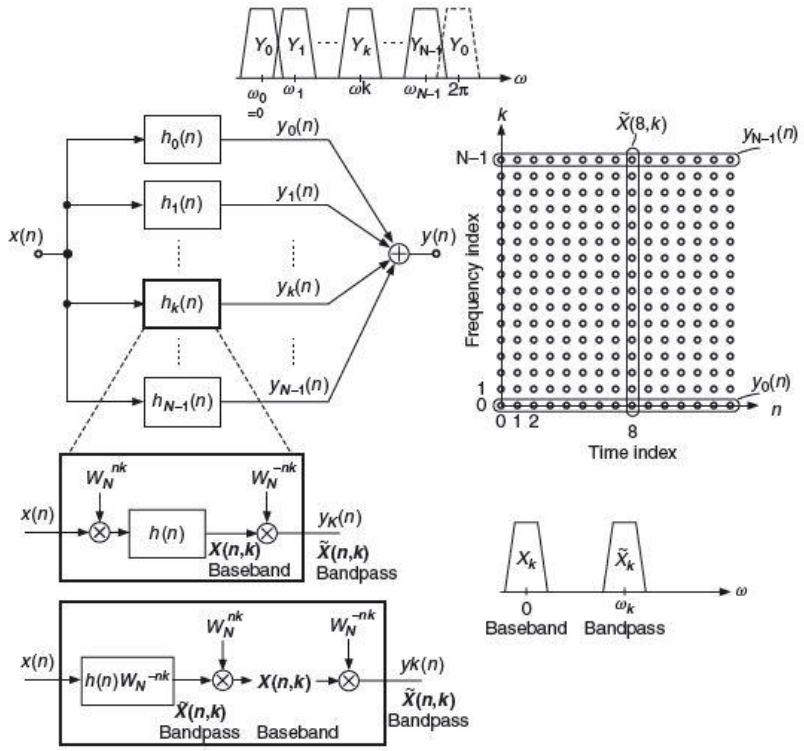
حالا می‌توانیم تعبیر STFT را به صورت عبور سیگنال اصلی از N فیلتر با مشخصات بالا تعریف کنیم. همچنین اگر به صورت فرکانسی این سیگنال دقت‌شود، مشاهده می‌کنیم که با انتخاب پنجره‌ی مناسب می‌توان با تقریب خوبی تمامی طیف فرکانسی سیگنال اصلی را پوشش‌داد. خروجی هرکدام از این bin های فرکانسی را با $y_k(n)$ نمایش می‌دهیم.

$$y_k(n) = \sum_{m=-\infty}^{\infty} x(m)h_k(n-m) = \sum_{m=-\infty}^{\infty} x(m)h(n-m)\,e^{j\frac{2\pi(n-m)k}{N}} = e^{j\frac{2\pi nk}{N}} \sum_{m=-\infty}^{\infty} x(m)h(n-m)\,e^{-j\frac{2\pi mk}{N}}$$

$$\rightarrow y_k(n) = e^{j\frac{2\pi nk}{N}}|X(n,k)|e^{j\phi(n,k)} \rightarrow |y_k(n)| = |X(n,k)| \hat{\phi}(n,k) = \frac{2\pi k}{N}n + \phi(n,k)$$

نکته‌ی مهم و کاربردی در مورد این ایده، امکان پیاده‌سازی آن به صورت بانک فیلتر است. ولی باید به این نکته توجّه داشت که میان این فاز و فازاصلی حاصل از STFT تفاوت وجوددارد.

با شکل زیر می‌توان بهتر با کارکرد این ایده آشناشد:



$$y(n) = \sum_{k=0}^{N-1} y_k(n) = \sum_{k=0}^{N-1} X(n,k)e^{j\frac{2\pi nk}{N}}$$

با توجه به آن‌که میدانیم سیگنال ورودی سیگنال حقیقی صوت است و همچنین سیگنال پنجره هم حقیقی است، میتوانیم رابطه‌ی زیر را بنویسیم:

$$y_k(n) = X(n,k)e^{j\frac{2\pi nk}{N}} = X^*(n,k)\left(e^{j\frac{2\pi n(N-k)}{N}}\right)^* = y^*_{N-k}(n)$$

$$\rightarrow \hat{y}_k(n) := y_k(n) + y_{N-k}(n) = y_k(n) + y^*_k(n) = |X(n,k)|\left(e^{j\hat{\phi}(n,k)} + e^{-j\hat{\phi}(n,k)}\right) = 2|X(n,k)|\cos\left(\hat{\phi}(n,k)\right)$$

$$\rightarrow \hat{y}_k(n) = 2|X(n,k)|.\cos(\omega_k n + \phi(n,k)) \rightarrow y(n) = \sum_{k=0}^{N/2} \hat{y}_k(n) = \sum_{k=0}^{N/2} A(n,k).\cos\left(\hat{\phi}_k(n,k)\right)$$

$\breve{A(n,k)}$ $\breve{\hat{\phi}(n,k)}$

دو راهکاری که در قسمت‌های بالا توضیح‌داده شدند، راهکارهای اصلی استفاده‌شده در بخش‌های اول تا سوم پروژه هستند. هرچند پروژه به گونه‌ای طرّاحی شده‌است که شما بدون آگاهی از این سازوکار هم می‌توانید پاسخ‌های مورد نظر پروژه را به دست بیاورید؛ امّا آگاهی از مبنای علمی آن می‌تواند هدف پروژه را بهتر برآورده کند. در صورت نیاز به راهنمایی بیشتر در مورد مبنای علمی و همچنین کارکرد کدها، می‌توانید به مسئولان حل‌تمرین پروژه‌ی درس مراجعه‌نمایید.

قبل از شروع باید چند نکته یادآوری شود:

۱- فایل نهایی شما شامل توابعی که در آن‌ها تغییری داده‌شده و فایل‌های صوتی حاصل (که به ترتیب در فولدرهای Functions و Results قراردارند) است. همچنین ارائه‌ی گزارش «تایپ‌شده» مخصوص بخش‌های تحلیلی که در آن نمودارهای حاصل و تحلیل‌های شما قرار دارد اجباری‌ست و عدم وجود آن به معنای از دست دادن کل نمره‌ی پروژه است. به منظور کاهش حجم فایل پروژه نهایی، برای بعضی بخش‌ها نیاز نیست فایل‌های صوتی نهایی را در گزارش خود داشته‌باشید. (در زمان ارسال گزارش آن فایل‌ها را از فولدر Results حذف‌کنید.) دقت کنید که در نهایت چیزی که باید آپلود کنید، فایلی زیپ‌شده با نام شماره دانشجویی خودتان است.

۲- پیشنهاد ما به شما این است که توابع را با صدای خودتان هم چک‌کنید. جالب خواهد بود! اما دقت کنید که فایل‌های نهایی که برای ما ارسال می‌کنید باید تنها شامل نتایج حاصل از نمونه‌های خود پروژه باشند. همچنین رفرنس اصلی که مبنای آشنایی ما با این تکنیک‌های صوتی بود در انتهای ترم و پس از تحویل پروژه در سایت درس اعلام خواهدشد تا علاقمندان با تکنیک‌های جذّاب این حوزه آشنا شوند.

۳- سعی‌شده تا جای ممکن پروژه روتین و سرراست باشد، و قسمت‌هایی که باید آن‌ها را تکمیل‌کنید واقعا ساده طراحی شده‌اند. به همین دلیل و به منظور تقویت توان خودتان، سعی‌کنید از تقلَب (و عواقب آن) بپرهیزید.

۴- سه جا نمره‌ی امتیازی برای شما در نظرگرفته شده‌است. ۵٪ برای پاسخ به سوال امتیازی، ۱۰٪ مرتّب بودن و تمیزی گزارش و ۱۰٪ تمیزی و رعایت قواعد کدزدن.

۵- در قسمت‌هایی که باید آن‌ها را تغییر دهید، کدهایی به صورت default وجود دارند. هدف از این کدها ارور ندادن خروجی در اجرای اولیه (قبل از اعمال تغییرات) است و باید آن‌ها را تغییر دهید.

حالا به سراغ خواسته‌های پروژه می‌رویم. قبل از شروع پروژه باید این توضیح را اضافه‌کنم که قسمت‌هایی از کد که باید تغییر داده‌شوند به صورت دقیق در فایل‌ها توضیح داده‌شده‌اند و پیشنهاد ما این است که در قسمت‌های دیگر کد تغییری ایجاد نکنید. اما ممکن است بعضی ورژن‌های نرم‌افزار متلب در کار با دستورات addpath دچار مشکل باشند که راه‌حل آن اضافه‌کردن دستی مسیر فایل محل ذخیره‌ی پروژه به تک‌تک آدرس‌های load یا save است.

قسمت اول:

از پوشه‌ی Functions، تابع phasewrapperرا انتخاب کنید. این تابع به‌این منظور تعریف شده‌است که با گرفتن آرایه‌ای از فازها، مقدار principle argument آن‌ها (یعنی مقدار اصلی فاز در بازه‌ی $[-\pi, \pi]$) را به عنوان خروجی گزارش‌دهد.

۱) این تابع را به گونه‌ای تعریف‌کنید که خروجی دلخواه را پیاده‌سازی کند.

از پوشه‌ی Functions، تابع voice_effects را انتخاب کنید. این تابع به‌این منظور طراحی شده‌است تا با دریافت فایل صوتی مورد نظر، دو افکت «خش‌دار کردن» و همچنین «صدای روباتیک» را پیاده‌سازی کند.

برای خش‌دار کردن صدا به این صورت عمل می‌کنیم که پس از اعمال STFTو محاسبه‌ی تبدیل فوریه‌ی پنجره‌ی مورد نظر، مقدار فاز رندمی بسته به مقدار ضریب hoarsening_effect به آن اضافه می‌کنیم. $\phi_r = C_{hoarsening}.Unif(-\pi, \pi)$

سپس مقادیر فاز جدید را با تابع phasewrapper به محدوده‌ی $[-\pi, \pi]$ باز می‌گردانیم.

۲) در بخش hoarsening effect، تبدیل فوریه‌ی segment را حساب‌کرده، مولفه‌های فاز و اندازه‌ی تبدیل فوریه‌ی حاصل را در دو بردار a و phi قرار دهید . سپس فاز رندمی که باید به هربخش اضافه شود را تعریف‌کنید.

به منظور اضافه‌کردن اثر روباتیک به صدا، پس از اعمال STFT، مقدار فاز تمامی مولفه‌های فرکانسی را برابر صفر قرار می‌دهیم. سپس عکس‌تبدیل فوریه را صورت می‌دهیم و دوباره فایل‌های حاصل را جمع می‌کنیم.

۳) در بخش robotizing effect، تبدیل فوریه‌ی segment را محاسبه نموده، پس از صفر قرار دادن فاز مولفه‌های آن تبدیل فوریه‌ی معکوس آن بخش را حساب‌کنید. دقت کنید که چون سیگنال صوت نمی‌تواند شامل مولفه‌های موهومی باشد، از مولفه‌های حقیقی این تبدیل فوریه‌ی معکوس استفاده شده‌است.

۴) (بخش تحلیلی) حالا که توابع مورد نیاز را ساختید، به فایل main برگردیم. در این فایل یک سری selector وجود دارند که مشخص می‌کنند چه بخش‌هایی از کد و چه توابعی اجرایی شوند. به منظور کار با قسمت نخست، section_selector را برابر ۱ قرار دهید.

۰.۴) ابتدا نمودار مربوط به تابع phasewrapper را مشاهده خواهیدکرد؛ آن را در گزارش خود ضمیمه کنید.

۱.۴) متغیر sub_section را برابر ۱ قرار دهید تا صدای خش‌دار تحویل بگیرید. ضریب hoarsening_coeff را از صفر تا یک برای چند مقدار مختلف تغییر دهید و فایل‌های خروجی را مقایسه‌کنید. (راهنمایی: برای مقادیر ۰ تا ۵.۰ تغییر زیادی در خروجی مشاهده نخواهیدکرد.) مطابق آموزه‌های استاد در کلاس، آنچه در سیگنال صوت مهم‌است دامنه، و آنچه در سیگنال تصویر مهم است فاز است. با توجه به خروجی‌های این بخش این گزاره را بررسی کنید. (تنها دو تا از فایل‌های صوتی که تفاوت معناداری دارند را در فایل نهایی ارسالی پروژه قرار دهید.)

۲.۴) متغیر sub_section را برابر ۲ قراردهید تا افکت صدای روباتیک را تحویل بگیرید. طول پنجره در این قسمت به صورت پیش‌فرض برابر ۱۰۲۴ نمونه درنظر گرفته‌شده. با تغییر متغیر multiplier-win از ۱ تا ۳۲ به صورت توان‌های ۲، خروجی‌های مختلف را مقایسه‌کنید. با توجه به این‌که این افکت و افکت قبلی هر دو افکت‌های غیرخطی هستند، انتظار تحلیل دقیق از شما نمی‌رود؛ اما پیشنهاد ما این است که در مورد علت این پدیده‌ها فکرکنید. (تنها دو تا از فایل‌های صوتی که تفاوت معناداری دارند را در فایل نهایی ارسالی پروژه قرار دهید.)

قسمت دوم:

از پوشه‌ی Functions، تابع voice_timevar را انتخاب‌کنید. این تابع به این منظور طراحی شده تا با دریافت سیگنال صدای مورد نظر، با کمترین آسیب به خصوصیات فرکانسی صدا، آن‌را در حوزه‌ی زمان منبسط (stretching) یا منقبض (compressing) کند. از کاربردهای اصلی این ایده می‌توان به نرم‌افزارهای مخصوص podcast یا audiobook اشاره‌کرد که در آن‌ها شنونده می‌خواهد سرعت پخش را بدون تغییر ماهیت فرکانسی صدای گوینده تغییر دهد. دو راهکاری که ما در این پروژه به کارگرفته‌ایم بر مبنای STFT هستند.

در قسمت اول این کد، به سه صورت و با بی‌دقتی سعی می‌کنیم انبساط و انقباض در حوزه‌ی زمان را پیاده‌سازی کنیم. دو خروجی down و upاز dwonsamling و upsampling سیگنال ورودی به دست آمده‌اند. این دو خروجی کنار سیگنال up2 خروجی اصلی این بخش را تشکیل می‌دهند.

۵) سیگنال up2 را به صورت زیر تعریف کنید: (L طول سیگنال ورودی است.)

$$\forall i \in \{1,2,\dots,L\}: up2(2i) = up2(2i-1) = input(i)$$

در قسمت دوم این کد، با استفاده از ایده‌ی جمع کسینوس‌های مولفه‌های فرکانسی (که در ابتدای این فایل توضیح داده‌شده) و همچنین ایده‌ی فرکانس لحظه‌ای (simultaneous frequency) انبساط و انقباض در حوزه‌ی فرکانس صورت داده‌شده. به زبان خیلی ساده، این ایده به این صورت کار می‌کند که از گام iام تا گام i+1ام، قرار است بردار اندازه‌ها ($\vec{a}$) و بردار فازها ($\vec{\Phi}$) به اندازه‌ی $\Delta\vec{a}$ و $\Delta\vec{\Phi}$ تغییرکند. حالا اگر بدانیم فاصله از این گام تا گام بعدی برابر s2 نمونه‌است، sample به sample خروجی باید چه مقدار تغییرکند. در صورت علاقه و برای اطلاعات بیشتر برای قسمت دوم و سوم این سوال می‌توانید به این مقاله مراجعه نمایید.

در قسمت سوم این کد، **که قرار است شما آن را تکمیل کنید**، با استفاده از ایده‌ای که در مقاله‌ی مذکور وجود دارد، سعی می‌کنیم پنجره‌هایی که در نتیجه‌ی STFT به دست آوردیم را، در مکان‌های متفاوت قرار دهیم. به این معنا که اگر در ورودی فاصله‌ی مرکز دو پنجره‌ی دارای همپوشانی s1 بود، در خروجی این فاصله برابر s2(i) باشد. چون این پنجره‌ها دارای همپوشانی هستند، اگر ضرایب انبساط را خیلی زیاد انتخاب نکنیم، زیاد نگران از بین‌رفتن سیگنال دلخواه در خروجی نخواهیم بود.

۶) در این قسمت، می‌خواهیم با داشتن بردار اندازه‌های اصلی ($\vec{a}$) از داده‌های اصلی و بردار فازهای اصلاح‌شده ($\overrightarrow{\Phi}$)، بردار مختلط جدیدی بسازیم که اندازه‌های اصلی و فازهای جدید داشته‌باشد، تا در قسمت بعد از آن تبدیل فوریه‌ی معکوس بگیریم. بردار $\overrightarrow{ft}$ را به گونه‌ای تعریف کنید که اندازه‌اش را از بردار $\vec{a}$ و فازش را از بردار $\overrightarrow{\Phi}$ گرفته باشد. یعنی:

$$\left|\overrightarrow{ft}\right| = \vec{a} \;\&\; \angle\overrightarrow{ft} = \overrightarrow{\Phi}$$

۷) (بخش تحلیلی) به فایل main بازمی‌گردیم. برای کار با این قسمت، section_selector را برابر ۲ قرار دهید.

۱.۷) متغیر sub_section را برابر ۱ قرار دهید و به فایل‌های خروجی گوش کنید. سپس sub_section را برابر ۳ قرار دهید. حالا هر دو متغیر rmin و rmax را یکبار برابر ۲/۱ (انقباض) و بار دیگر برابر ۲ (انبساط) قرار دهید و به فایل‌های خروجی گوش‌دهید. چه تفاوتی با فایل قسمت اول دارد؟ (فایل‌های صوت حاصل از این قسمت را در فایل‌های نهایی ارسالی قرار ندهید.)

۲.۷) متغیر rmin را برابر ۱/۳ و متغیر rmax را برابر ۳/۲ قرار داده و sub_section را برابر ۳ قرار دهید. به فایل‌های حاصل گوش کنید و فایل حاصل را در فایل‌های نهایی ارسالی قراردهید.

۳.۷) متغیر drawplot را برابر ۱ قراردهید. ضمن الصاق نمودارهای بدست آمده به گزارش پروژه‌ی خود، تفاوت میان راهکارهای اصلی و راهکارهای ساده‌انگارانه را بیان کنید.

۴.۷) (امتیازی) در فایل drawplots در پوشه‌ی Functions، بخش مربوط به time stretching را مشاهده‌کنید. ابتدا سیگنال up2 را اینجا هم تعریف‌کنید. سپس با استفاده از ایده‌ی مربوط به بخش ۵ نمودار سومی شامل دو زیرنمودار رسم‌کنید: یکی تبدیل فوریه‌ی up و دیگری up2 باشد. حالا این دو را با هم مقایسه‌کنید. چه تفاوتی وجود دارد؟ چرا؟

بخش دوم)

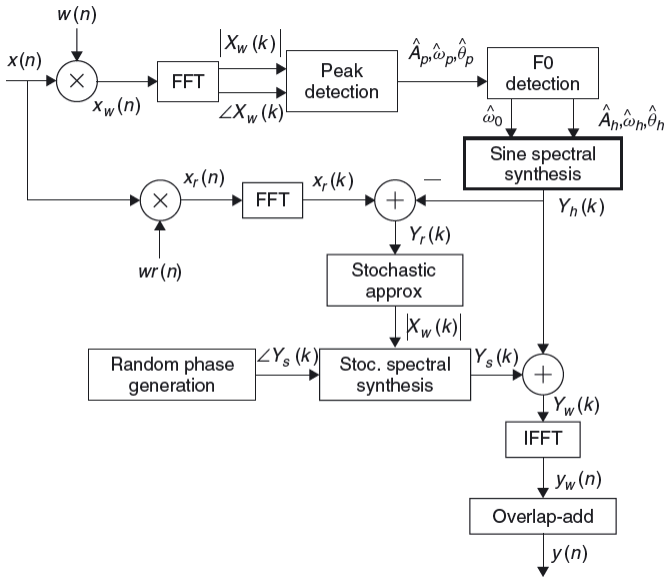
در بخش‌های قبل با مدل STFT آشنا شدید. این نوع نمایش سیگنال در فضای فرکانسی برای اعمال افکت‌های پیچیده در صدا مناسب نمی‌باشد. به همین دلیل به سراغ مدلی می‌رویم که از لحاظ محاسباتی پیچیده‌تر ولی انعطاف‌پذیری آن نسبت به مدل STFT بیش‌تر است. مدلی که در این بخش بررسی می‌کنیم مدل سینوسی به همراه باقی‌مانده‌ی تصادفی (Stochastic Residual) نام دارد. سیگنال صدای $s(t)$ به صورت زیر نمایش داده می‌شود:

$$s(t) = \sum_{r=1}^{R} A_r(t)cos[\theta_r(t)] + e(t)$$

که در آن $A_r(t)$ و $\theta_r(t)$ به ترتیب دامنه و فاز لحظه‌ای r امین مولفه‌ی سینوسی و R تعداد این مولفه‌هاست. $e(t)$ نیز مولفه‌ی باقی‌مانده است که در این جا فرض می‌شود که یک سیگنال تصادفی (Stochastic) است که می‌توان آن را به صورت یک نویز سفید فیلتر شده در نظر گرفت. محدودیتی که برای سادگی در این پروژه در نظر گرفته می‌شود فرض هارمونیک بودن مولفه‌های سینوسی است که لازم می‌دارد صدای ورودی هارمونیک و مونوفونیک باشد. یعنی فرکانس هر مولفه‌ی سینوسی ضریبی صحیح از فرکانس اولین مولفه‌ی سینوسی است.

اعمال افکت روی یک سیگنال صدا معمولا از سه مرحله تشکیل شده است. آنالیز، اعمال تغییر و سنتز. در این پروژه ورودی یک سیگنال صداست که هر بار چند نمونه از آن در محدوده‌ی یک فیلتر پنجره‌ای انتخاب می‌شود (فریم) و پس از اعمال STFT بر روی آن در حوزه‌ی فرکانس پارامترهای مربوط به مدل بالا استخراج شده (آنالیز) و سپس تغییرات موردنظر بر روی طیف سیگنال صدا اعمال می‌شود (اعمال تغییر). در انتها با تبدیل فریم‌ها به حوزه‌ی زمان و کنار هم قرار دادن فریم‌های حاصل به شیوه‌ای مناسب صدای خروجی به دست می‌آید (سنتز).

بلوک دیاگرام مراحل لازم برای آنالیز و سنتز در شکل زیر آورده شده است:



پس از اعمال STFT و به دست آوردن نمایش فریم در حوزه‌ی فرکانس به این نکته توجه می‌کنیم که با به کار بردن فیلتر پنجره‌ای مناسب در هنگام ساختن فریم و با فرض پایدار بودن دامنه و فرکانس مولفه‌های سینوسی می‌توان در حوزه‌ی فرکانس مکان‌های مربوط به این مولفه‌ها را شناسایی کرد. در این مکان‌ها طیف فریم شبیه طیف فیلتر پنجره‌ای است که در هنگام ساختن فریم استفاده کرده‌ایم. با توجه به این نکته می‌توان فرکانس مولفه‌های سینوسی (Partials) را پیدا کرد. فیلتری که به کار می‌بریم این مشخصه را دارد که بیش‌تر انرژی آن حول فرکانس صفر جمع شده است و در فرکانس صفر Peak دارد. پس کافی است در طیف حاصل از STFT ابتدا بیشینه‌های محلی را پیدا کنیم و سپس آن‌هایی را که از یک مقدار آستانه بیش‌تر هستند به عنوان Peak در نظر می‌گیریم.

چون سیگنال صدای ورودی را هارمونیک فرض کرده‌ایم می‌توانیم با یافتن فرکانس هارمونیک اصلی مکان بقیه‌ی هارمونیک‌ها را نیز تخمین بزنیم. پس در این مرحله باید فرکانس هارمونیک اصلی را پیدا کنیم. برای این کار از الگوریتم Yin بهره می‌بریم. این الگوریتم در حوزه‌ی زمان عمل می‌کند. الگوریتم به این صورت است که سعی می‌کند دوره تناوب اصلی سیگنال را بیابد. با داشتن دوره تناوب اصلی می‌توان فرکانس هارمونیک اصلی را یافت. توجه به این نکته که یک سیگنال متناوب با شیفت یافته‌اش در حوزه‌ی زمان همبستگی بالایی دارد می‌تواند در پیدا کردن دوره تناوب ما را یاری کند. برای این منظور توابعی به شکل زیر تعریف می‌کنیم:

$$d(\tau) = \sum_{j=t+1}^{t+w} (x(j) - x(j+\tau))^2$$

$$\mathrm{d}'(\tau) = \begin{cases} 1 & if\ \tau = 0 \\ \frac{\tau . d(\tau)}{\sum_{j=1}^{\tau} d(j)} & O.W. \end{cases}$$

در توابع بالا x سیگنال ورودی، τ مقداری که سیگنال را شیفت داده‌ایم و W اندازه‌ی پنجره‌ای است که روی آن جمع بالا را انجام می‌دهیم. سپس در تابع $d'(\tau)$ به دنبال کمینه‌های محلی می‌گردیم که از یک مقدار آستانه کم‌تر باشند. این مقادیر را با استفاده از Parabolic Interpolation بهبود می‌بخشیم و اولین τ ای که در آن کمینه رخ می‌دهد را به عنوان دوره تناوب معرفی می‌کنیم و در انتها فرکانس هارمونیک اصلی مشخص می‌شود.

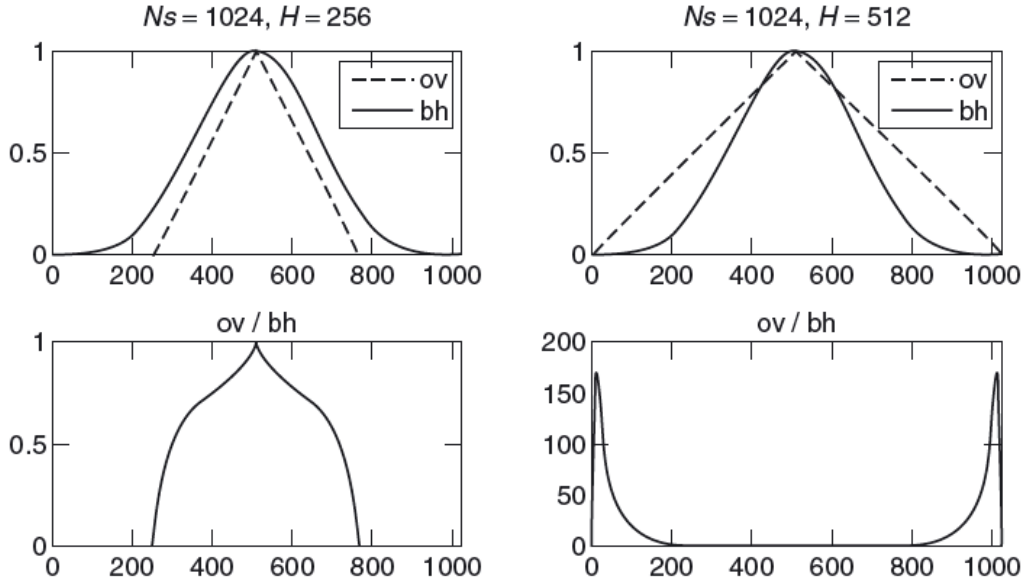
پس از یافتن Peakها و فرکانس هارمونیک اصلی، آرایه‌ای از فرکانس‌های هارمونیکی را ساخته و مشخصات Peakای را که فرکانس آن به هارمونیک nام نزدیک است به عنوان مشخصات هارمونیک nام دخیره می‌کنیم.

حال می‌توانیم طیف فرکانسی بخش سینوسی یک فریم را که در یک فیلتر پنجره‌ای هم ضرب شده است بسازیم. اما ابتدا به برخی نکات در رابطه با پنجره‌ی سنتز توجه می‌کنیم. گفتیم که برای یافتن طیف مولفه‌های سینوسی از یک فیلتر پنجره‌ای استفاده می‌کنیم که بیش‌تر انرژی آن حول فرکانس صفر جمع شده است. فیلتر مورد نظر در این پروژه فیلتر Blackman-Harris 92 dB است. برای جلوگیری از تغییرات ناگهانی میان فریم‌های متوالی، فریم‌ها با یکدیگر overlap دارند. در این حالت فیلتر پنجره‌ای باید این شرط را برآورده کند تا سیگنال اصلی تغییر نکند:

$$\sum_{n=-\infty}^{\infty} w(m - nH) \approx constant$$

که در آن w فیلتر پنجره‌ای و H فاصله‌ی بین وسط فریم‌های متوالی (Hop Size) است.

برای فیلتر Blackman-Harris 92 dB باید H مقدار کمی باشد تا شرط فوق برآورده شود ولی این از لحاظ محاسباتی به صرفه نیست. راه حلی که برای مقابله با این مشکل وجود دارد این است که پس از اعمال فیلتر، اثر فیلترینگ بر روی فریم را با تقسیم بر فیلتر در حوزه‌ی زمان و قبل از Overlap خنثی کنیم و سپس از فیلتر مثلثی برای overlapping استفاده کنیم. مقدار هم‌پوشانی بین دو فریم در هنگام سنتز نیز باید بیش‌تر از ۵۰٪ باشد تا مقدار گین در لبه‌های فیلتر خیلی بزرگ نباشد. برای فهم بهتر نمودارهای زیر را ملاحظه بفرمایید .



حال برای محاسبه‌ی طیف فرکانسی بخش سینوسی یک فریم کافی است مقادیر دامنه و فاز را حول فرکانس‌های محاسبه شده برای Peak هایی که معرف هارمونیک-ها هستند و با توجه به مقادیر فیلتر Blackman-Harris 92 dB محاسبه کنیم. در این پروژه 4 مقدار پیش و پس هر Peak در نظر گرفته شده است. فیلتر Blackman-Harris 92 dB و zero-centered شده در حوزه‌ی زمان به صورت زیر است:

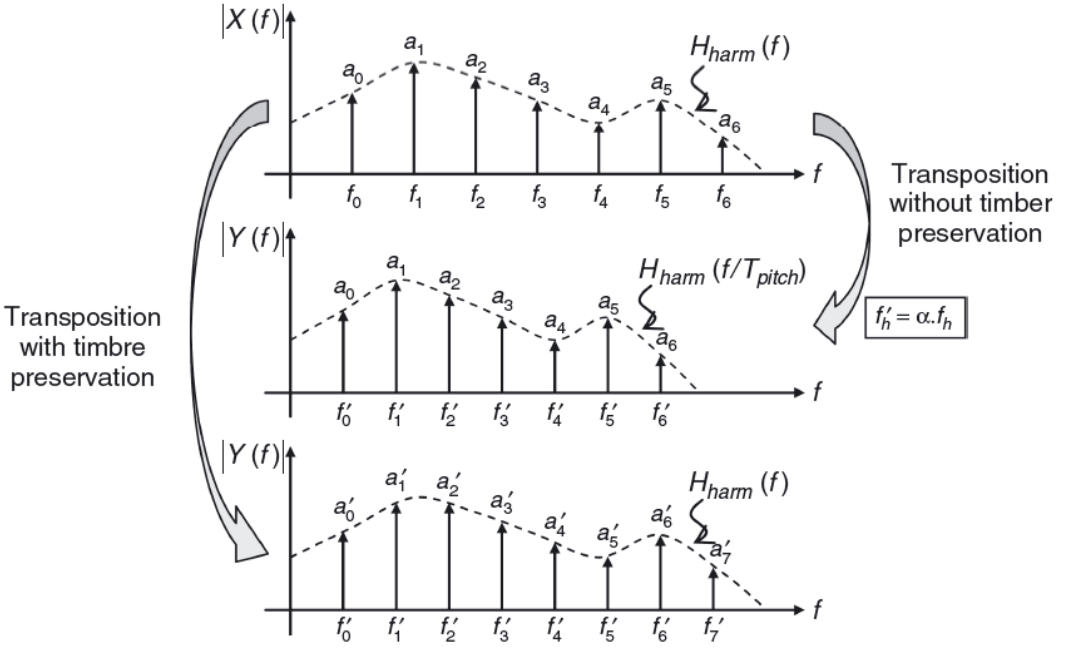
$$w(n) = 0.35875 + 0.48829cos\frac{2\pi n}{N} + 0.14128cos\frac{4\pi n}{N} + 0.01168cos\frac{6\pi n}{N}$$

پس از محاسبه‌ی طیف فرکانسی بخش سینوسی سیگنال، به محاسبه‌ی طیف فرکانسی بخش باقی‌مانده (Residual) که فرض کردیم Stochastic است می‌پردازیم. با این فرض، لازم به محاسبه‌ی فاز لحظه‌ای یا اطلاعات دقیق شکل طیف نیست. چون در محاسبه‌ی بخش سینوسی یک سیگنال فریم‌ها را از فیلتر Blackman-

Harris عبور دادیم، ابتدا فریم اصلی را دوباره از این فیلتر عبور داده و در فضای فرکانسی طیف بخش سینوسی را از طیف فریم اصلی فیلتر شده کم می‌کنیم تا طیف بخش باقی‌مانده به دست آید. سپس دامنه‌ی این طیف را Downsample می‌کنیم تا تخمینی از پوش این سیگنال داشته باشیم و با استفاده از فازهای تصادفی در هر فریم می‌توانیم طیف بخش باقی‌مانده را تخمین بزنیم.

افکت‌هایی که قصد بررسی آن‌ها را داریم عبارتند از: تبدیل صدای مرد به زن و بالعکس و نیز تبدیل صدای مرد به بچه.

می‌دانیم که فرکانس‌های هارمونیک صدای زن معمولا از صدای مرد مقادیر بالاتری دارند. پس باید برای تبدیل صدای مرد به زن فرکانس هارمونیک‌ها را در عددی بزرگ‌تر از یک ضرب کنیم و برای تبدیل عکس باید فرکانس هارمونیک‌ها را در یک عدد کوچک‌تر از یک ضرب کنیم. تغییر دیگری که باید در صدای اصلی ایجاد شود scale کردن کنترل شده‌ی timbre است. وقتی فرکانس هارمونیک‌ها را Scale کردیم اگر دامنه‌ی آن‌ها را اصلاح نکنیم timbre هم ناخواسته scale می‌شود. یعنی محدوده‌ی طیف فرکانسی سیگنال دچار تغییر زیر می‌شود که این مشکل‌ساز است. به همین دلیل باید روی این اثر کنترل داشته باشیم.



برای رفع این مشکل ابتدا با درون‌یابی خطی دامنه‌های طیف سیگنال، envelope آن را به دست می‌آوریم و سپس دامنه‌ی فرکانس‌های Scale شده را از envelope به دست آمده محاسبه می‌کنیم. برای بخش باقی‌مانده‌ی طیف نیز این کار را انجام می‌دهیم با این تفاوت که این بار envelope این بخش را با استفاده از دامنه‌های طیف بخش residual به دست می‌آوریم و سپس دامنه‌های جدید را برای طیف باقی‌مانده محاسبه می‌کنیم.

تارهای صوتی زن معمولاً نسبت به تارهای صوتی مرد کوتاه‌تر هستند که منجر به این می‌شود که صدای زن فرکانس‌های رزونانس بالاتری داشته باشد. برای تبدیل صدای مرد به زن، timbre صدا باید گسترده‌تر شود؛ یعنی پس از scale فرکانس‌های هارمونیک، طیف سینوسی و طیف بخش باقی‌مانده باید باند گسترده‌تری داشته باشند. در این پروژه بخش فرکانس پایین صدای مرد از ۰ تا ۴۰۰۰ هرتز را به ۰ تا ۵۰۰۰ هرتز گسترش می‌دهیم تا به صدای زن تغییر یابد. برای تبدیل صدای زن به مرد هم محدوده‌ی فرکانسی بین ۰ تا ۵۰۰۰ هرتز صدای زن به ۰ تا ۴۰۰۰ هرتز فشرده می‌شود. تبدیل صدای مرد به بچه هم با گسترش محدوده‌ی فرکانسی ۰ تا ۳۶۰۰ هرتز به ۰ تا ۵۰۰۰ هرتز انجام می‌شود. در گسترش یا فشرده‌سازی طیف توجه می‌کنیم که طبق توضیحات بالا دامنه‌ها را اصلاح کنیم.

قسمت سوم)

۱) تابع bh92transform را پیاده‌سازی کنید. این تابع تبدیل فوریه‌ی پنجره‌ی Blackman-Harris و Zero-Centered شده را در binهای فرکانسی موردنظر حساب می‌کند. اطمینان حاصل کنید که بخش اعظم انرژی این فیلتر حول مبدا است.

۲) پس از تکمیل تابع bh92transform می‌توانیم با استفاده از آن و با داشتن مکان‌ها و مشخصات دامنه و فاز Peakها طیف فرکانسی بخش سینوسی سیگنال را بسازیم. تابع sinestransform با گرفتن مکان و دامنه و فاز Peakها به عنوان ورودی طیف بخش سینوسی را می‌سازد. این تابع را تکمیل کنید. طیف دامنه‌ی یک فریم نمونه در خروجی نشان داده می‌شود.

۳) تابع yindetectf0 فرکانس هارمونیک اصلی سیگنال را مشخص می‌کند. این تابع را بر مبنای الگوریتم Yin تکمیل کنید. یک فریم نمونه به عنوان ورودی به تابع داده می‌شود تا فرکانس هارمونیک اصلی در خروجی چاپ شود.

۴) (بخش تحلیلی) در فایل main متغیر section_selector را برابر ۳ قرار دهید.

الف) متغیر sub_section را برابر ۱ قرار دهید. متغیر آرایه‌ای input_bins را ابتدا برابر [20:20-] قرار دهید و دامنه‌ی طیف خروجی فیلتر Blackman-Harris را مشاهده کنید. حال با تغییر این متغیر حدودا تعیین کنید که بیش‌تر انرژی این فیلتر حول چه فرکانس‌هایی است؟

ب) متغیر sub_section را برابر ۲ قرار دهید. متغیر bn را برابر ۱، ۴ و ۱۰ قرار دهید و دامنه‌ی طیف بخش سینوسی را مشاهده کنید. آیا تفاوت محسوسی بین حالات bn = 1 و bn = 4 می‌بینید؟ در مورد حالات bn = 4 و bn = 10 چه طور؟ به طول زمان اجرای کد و کیفیت طیف خروجی توجه کنید. برای اجرای بهینه‌ی کد کدام حالت را انتخاب می‌کنید؟ پاسخ خود را توجیه کنید.

پ) متغیر sub_section را برابر ۳ قرار دهید. متغیر yin_window_length را که طول پنجره‌ی Integration در الگوریتم Yin را نشان می‌دهد برابر ۰٫۰۱۲۵، ۰٫۰۱۰۰، ۰٫۰۰۵۰ و ۰٫۰۰۳۰ قرار دهید و فرکانس‌های هارمونیک اصلی به دست آمده را مقایسه کنید. اثر طول پنجره‌ی Integration را در این الگوریتم توضیح دهید.

ت) متغیر sub_section را برابر ۳ ، متغیر bn را برابر ۴ و متغیر yin_window_length را برابر ۰٫۰۱۲۵ قرار دهید. سپس با قرار دادن متغیرهای effect_number برابر ۱ و fscale برابر ۲ و timbremapping برابر [0 4000 fs/2;0 5000 fs/2] با اجرای کد، تبدیل صدای مرد به زن انجام می‌شود. متغیر fscale را کم‌تر و بیش‌تر کنید. در هر مورد چه تاثیری در خروجی می‌بینید. با بزرگ و کوچک کردن محدوده‌ی mapping فرکانس ورودی در متغیر timbremapping چه تغییری در خروجی ایجاد می‌شود؟ (این بخش را با نمونه صدای basket.wav یا ACS.wav انجام دهید.)

ث) مراحل بالا را برای effect_number برابر ۲ و fscale برابر ۰٫۵ و timbremapping برابر [0 5000 fs/2;0 400 fs/2] تکرار کنید. با اجرای کد، تبدیل صدای زن به مرد انجام می‌شود. (این بخش را با نمونه صدای meetingt.wav یا Metamorphosis.wav انجام دهید.)

ج) مراحل بالا را برای effect_number برابر ۳ و fscale برابر ۲ و timbremapping برابر [0 3600 fs/2;0 5000 fs/2] تکرار کنید. با اجرای کد، تبدیل صدای مرد به بچه انجام می‌شود. (این بخش را با نمونه صدای basket.wav یا ACS.wav انجام دهید.)