پروژه درس سیگنال سیستم دکتر امینی

محمدحسین استادی

تیر ۱۴۰۱

فهرست مطالب

۱ توضیحات توابع ۳
۱.۱ audio import ۳
۲.۱ FFT ۳
۳.۱ STFT ۳

۲ **create database** ۳

۳ ساخت لیست در **search database** ۴

۴ چرا **hash map**؟ ۴

۵ **test musics** ۴

۶ تولید نویز با **snr** ۵

۷ احتمال با نویز سفید گاوسی ۵

۸ استودیو نویزدار! ۵

۹ ترکیب دو آهنگ ۷

۱ توضیحات توابع

۱.۱ audio import

ابتدا توسط تابع audioread سیگنال صوتی و نرخ نمونه برداری آن بدست می آید. سپس بین دو کانال میانگین گرفته و mono audio بدست می آید. در نهایت توسط تابع resample سیگنال را دون سمپل میکنیم.

۲.۱ FFT

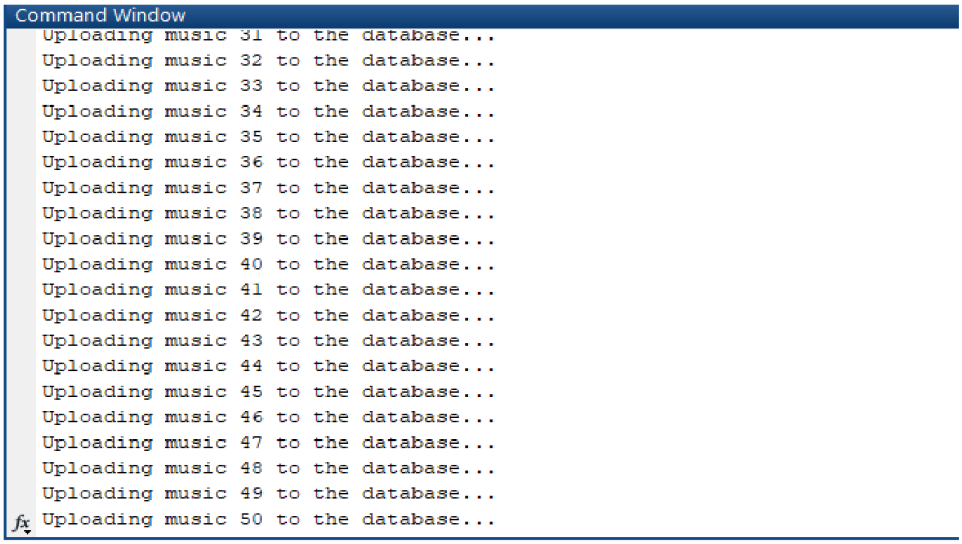
مراحل انجام شده در تابع مانند سوال است. اندازه تبدیل فوریه را تقسیم بر طول بردار ورودی کرده ، سپس این بردار را به توان دو میرسانیم. حال باید فرکانس های منفی را صفر کرده و مثبت ها را در دو ضرب کنیم. فرکانس های مثبت در واقع نیمه اول بردار توان به دست آمده است و فرکانس های منفی نیمه دوم آن هستند. پس با یک حلقه نیمه اول را در دو ضرب کرده و در خروجی تابع قرار میدهیم.

۳.۱ STFT

بخش قابل توجهی از این تابع به صورت آماده بوده است و تنها کد حلقه تکمیل شده است. ابتدا پنجره متناظر با شمارنده حلقه را پیدا کرده سپس از آن FFT گرفته و در ستون i میگذاریم.

۲ create database

ابتدا مسیر چهار پوشه به برنامه داده میشود و یک مپ خالی به نام دیتابیس که نوع داده های آن مشخص است ساخته میشود. بعد از این باید اثرانگشت آهنگ ها در دیتابیس ذخیره شود. پس در یک حلقه که به اندازه طول آهنگ ها است موزیک ها را به همراه فرکانس جدیدشان ترتیب توسط تابع import audio گرفته و به کمک تابع STFT پنجره STFT را میسازیم.
بخش بعدی نیز مربوط به ترسیم پنجره گفته شده است. حال باید anchor point ها را پیدا کنیم! مقادیر پارامترهای dt و df را وارد کرده و با تابع find anchor points دیتاها را در متغیر anchor points میگذاریم. چندخط بعدی مربوط به ترسیم anchor points است.
حال hash tagها باید ساخته شوند. مقادیر df hash و dt hash داده شده سپس با تابع مربوط hash value و hash key مقداردهی میشوند. اکنون باید این اطلاعات را در دیتابیس ذخیره کنیم. در حلقه بعدی این کار مطابق با فرمت گفته شده انجام میشود.(با درنظر گرفتن hash key تکراری).
در نهایت نیز دیتابیس ذخیره میشود. با اجرا کردن این کد دیتابیس تکمیل میشود.



شکل ۱: ذخیره کردن اثرانگشت آهنگ ها

۳ ساخت لیست در search database

در بخش های ابتدایی کد مراحلی که در قسمت قبل گفته شد طی میشود و دو متغیر hash key و hash value بدست می آید. سپس تمام hash key ها را بررسی میکنیم. به ازای هریک از key tagها (همان hash key است که متمرکز و تبدیل به رشته شده است) چک میکنیم که در دیتابیس موجود است یا خیر. متغیر temp1 hash valueهایی هستند که از دیتابیس به دست آمده و با کاراکتر + از هم جدا شده اند. حال در حلقه بعدی به ازای هر hash value یک پارامتر temp2 داریم که شامل شماره آهنگ و زمان anchor point مرجع است.
لیست با یک متغیر جدید با این اطلاعات آپدیت میشود؛ شماره آهنگ دیتابیس، زمان درایه anchor point مرجع در آهنگ دیتابیس، زمان درایه anchor point مرجع در آهنگ جستجوشده.

۴ چرا hash map؟

الگوریتم را از دو جهت میتوان بررسی کرد؛ یکی اردر میانگین زمانی و دیگری نیز بدترین حالت. الگوریتم جستجو در هش مپ متلب در بدترین حالت $O(n)$ است اما میانگین زمانی آن $O(1)$ است؛ یعنی به اندازه داده ها وابستگی ندارد. درحالی که اگر hash tag را جستجو میکردیم در بدترین حالت $O(n^2)$ و در میانگین اردر $O(n)$ را داشتیم. پس استفاده از hash map بهینه تر است.

۵ test musics

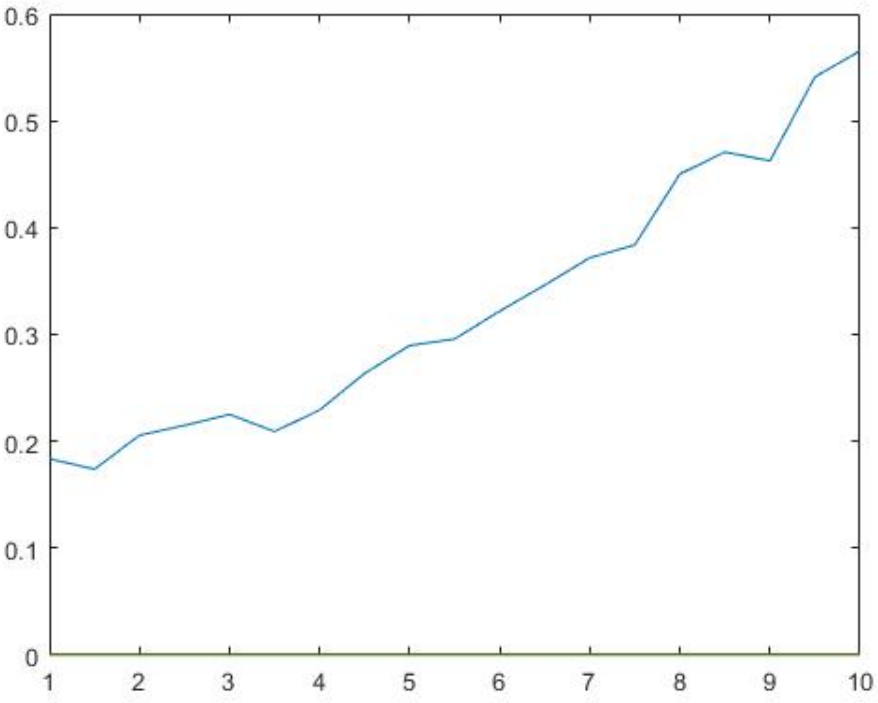
تست اول متعلق به آهنگ پنجم است. ابتدا از ستون اول ردیف هایی که شماره پنج دارند را پیدا کرده و این را در ستون دوم ضرب میکنیم. سپس میانه اعضای غیرصفر این بردار را پیدا میکنیم زیرا اعضای صفر برای بقیه آهنگ ها هستند و نباید در محاسبات وارد شود. سپس این عدد را بر 20 تقسیم کرده تا زمان واقعی به دست می آید. تصاویر زیر زمان های شروع به دست آمده را نشان میدهد.

۶ تولید نویز با snr

برای تولید موزیک نویزی یک فایل جدید به نام create noisy music زده شده است. موزیک شماره ده از پوشه موزیک ها انتخاب شده سپس بیست ثانیه از آن جدا میشود. سپس نویز تولید شده و با استفاده از snr توان آنرا تنظیم میکنیم. در نهایت موزیک را با استفاده از بردار نهایی تولید میکنیم. این آهنگ های تولید شده را در پوشه noisyMusics قرار میدهیم. نرخ snr آنها به ترتیب برابر با پنج، منفی پنج، منفی پانزده و منفی بیست به ازای آهنگ های یازده تا چهارده است. این آهنگ ها را در پوشه تست قرار داده و با فایل سرچ آنها را تست میکنیم. حدودا در نرخ snr منفی بیست دیگر موزیک پیدا نمیشود.

۷ احتمال با نویز سفید گاوسی

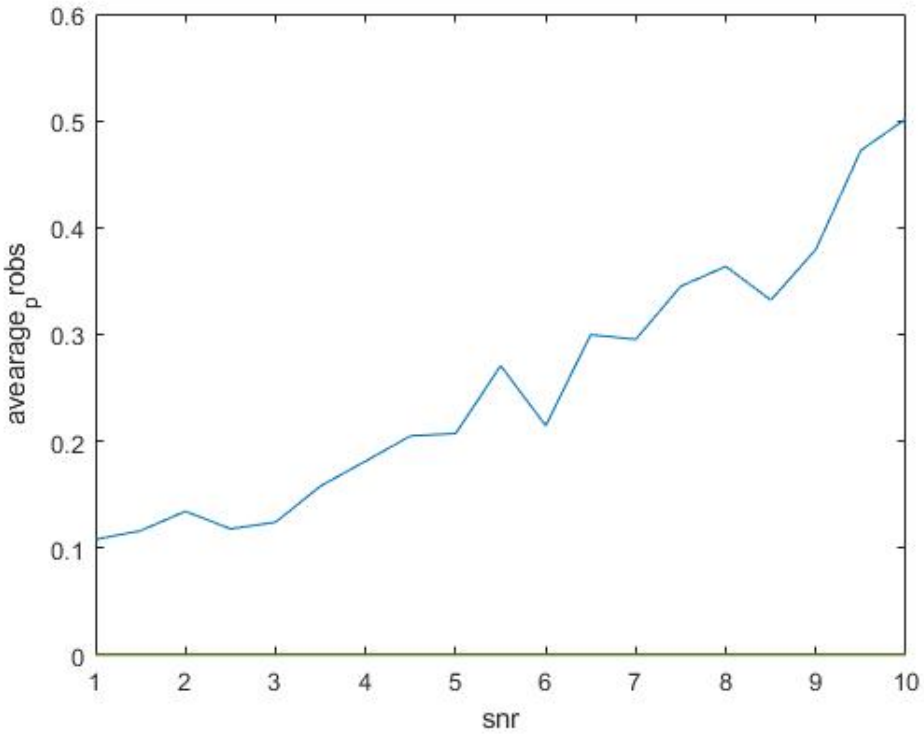
برای این بخش کد probability snr search database قرار داده شده است. بخش زیادی از این کد کپی شده از search database است برای کاهش زمان اجرای کد گام های snr به نیم تبدیل شده و تعداد آهنگ های رندوم نیز به ۵۰ کاهش پیدا کرده است. با اجرا این برنامه بر روی آهنگ ۱۵ تصویر زیر بدست می آید. این کار برای آهنگ های ۱۶ و ۱۷ نیز انجا شده و تصویر آنها در گزارش موجود است.



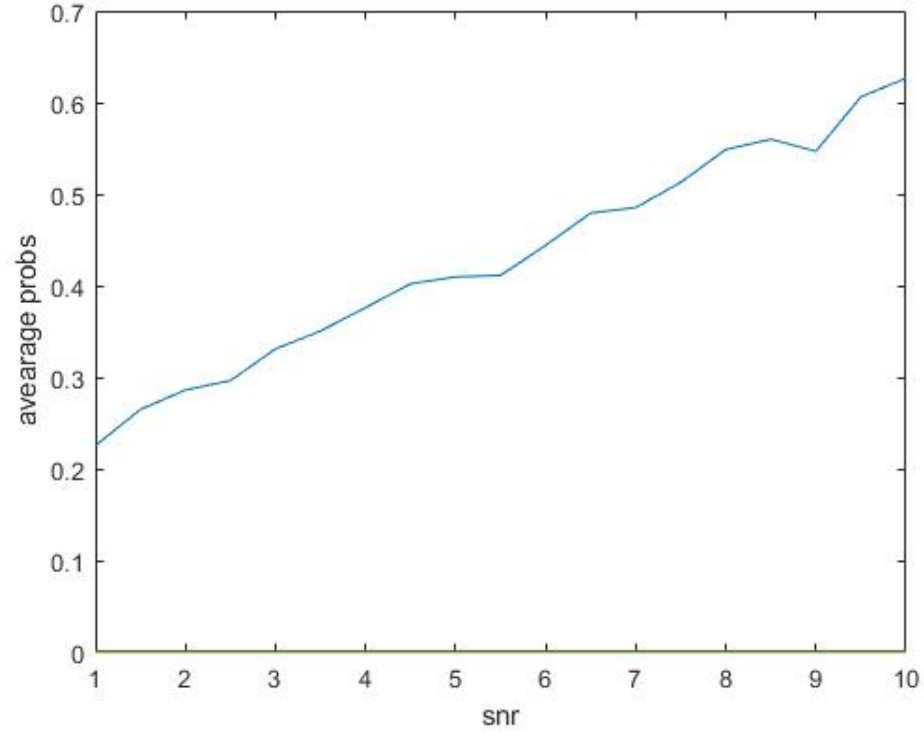
شکل ۲: میانگین احتمال بر حسب snr

۸ استودیو نویزدار!

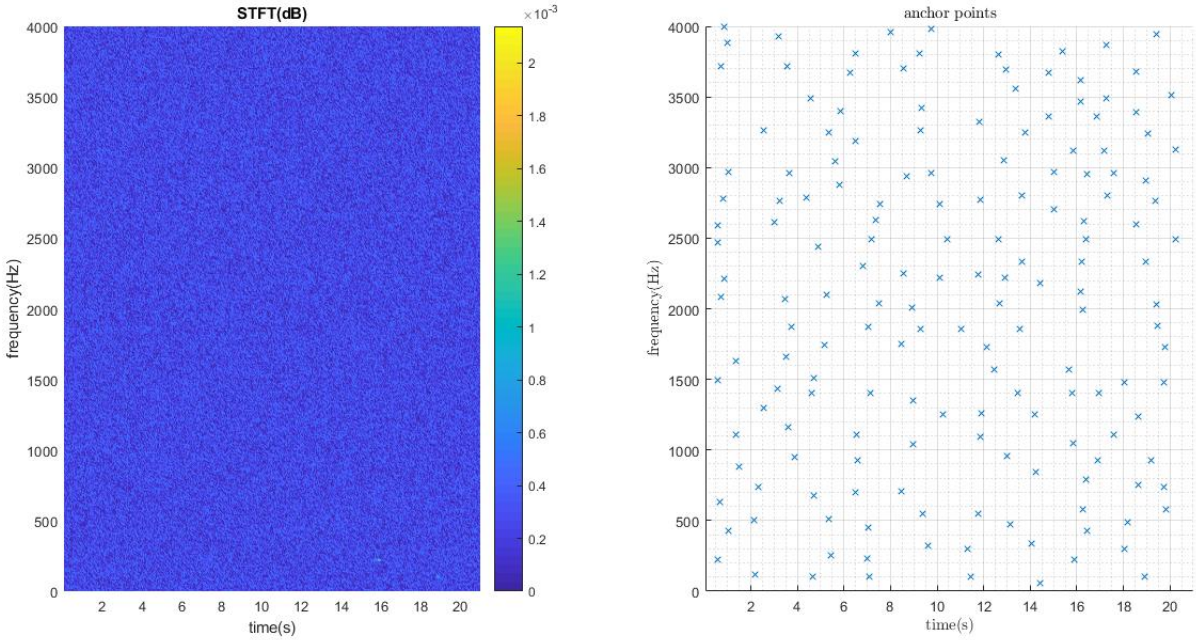
با استفاده از گوشی در شرایط مختلف از نظر بلندی صدا و بلندی نویز و ... سه آهنگ به دست آمده که در پوشه noisy studio قرار دارند. با قرا دادن اینها در پوشه تست و جستجو آنها نتیجه شد که آهنم ۱۵ تقریبا هیچگاه پیدا نمیشود. شماره ۱۶ با احتمال کمی در صدر لیست قرار میگیرد و گاهی نیز پیدا نمیشود و در آخر نیز فایل ۱۷ تقریبا همیشه جواب صحیح را میدهد.(پسوند فایل باید به m4a تغییر کند) نمودار های خواسته شده برای آهنگ ۱۷ رسم شده است.



شکل ۳: نمودار احتمال آهنگ ۱۶



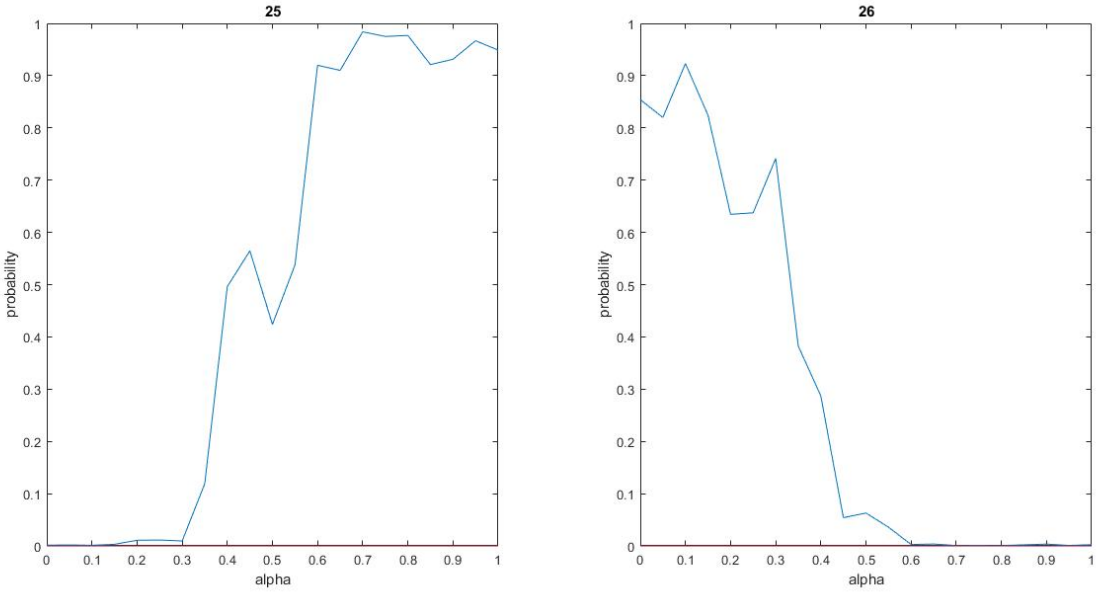
شکل ۴: نمودار احتمال آهنگ ۱۷



شکل ۵: STFTو anchorPoints موزیک۱۷

۹ ترکیب دو آهنگ

برای این بخش اسکریپت create mix music ایجاد شده است. ابتدا دو موزیک را(در اینجا ۲۵ و ۲۶ انتخاب شده) import کرده و با ضرایب گفته شده در سوال آنها را پس از کات کردن بیست ثانیه ای ترکیب میکنیم. گام های آلفا برای دقت مناسب 0.05 انتخاب شده است. بخش بعدی در حلقه از فایل search database کپی شده است. در نهایت پس از امتیازدهی احتمال دو فایل را پیدا کرده و در $p1$ و $p2$ وارد میکنیم. تصویر به شکل زیر است.



شکل ۶: احتمال ترکیب دو آهنگ