به نام خدا

ملیکا احمدی رنجبر 97521036

فشرده سازی صوت

در بخش دوم این پروژه، برای پیاده سازی، از الگوریتم DCT استفاده شده است، که در ادامه آن را توضیح میدهیم و کد را نیز مرحله به مرحله بررسی میکنیم.

❖ DCT:

Discrete Cosine Transform is used in lossy audio compression because it has very strong energy compaction, i.e., its large amount of information is stored in very low frequency component of a signal and rest other frequency having very small data which can be stored by using very less number of bits.

❖ بررسی کد:

```
%% Read
[Voice, Frequency] = audioread('Sample.m4a');

% Chosing a block size
windowSize = 7000;

% Compression  percentages
samplesHalf = windowSize / 2;

% Initializing compressed matrice
HalfCompression = [];
```

در ابتدای کد، فایل صوتی را میخوانیم و آن را در قالب Vector ذخیره میکنیم، همچنین فرکانس آن را نیز داریم.
پس از آن با توجه به نحوه الگوریتمی که ستفاده میکنیم، باید اندازه قاب را مشخص کنیم (Window size).
در واقع این قابی که اندازه آن را مشخص میکنیم، به ترتیب و با اندازه این قاب روی Vector صوت ما حرکت میکند و DCT را روی آن قاب انجام میدهد. (در مراحل بعد میبینیم)
پس از تعیین اندازه قاب، مقداری که میخواهیم فایل فشرده شود را معلوم میکنیم، به نسبت فایل اویه. (همانطور که میبینید اندازه قاب را با توجه به مقدار تغییرات مد نظر ما برای فشرده سازی تغییر میدهد).
و در نهایت در این بخش،Vector فایل جدید را Initialize میکنیم.

```
%% Compression
for i= 1: windowSize:length(Voice) - windowSize
    windowDCT = dct(Voice(i: i + windowSize - 1));
    HalfCompression(i: i + windowSize - 1) = idct(windowDCT(1: samplesHalf), windowSize);
end
```

در این قسمت بخش اصلی را برای عمل فشرده سازی مشاهده میکنید، که البته از توابع آماده در خود نرم افزار متلب استفاده شده است، به این معنا که الگوریتم DCT نیازی به پیاده سازی کامل از طرف ما ندارد و کافی است با توجه به مقادیری که در ابتدا تعریف کردیم، این توابع را مورد استفاده قرار دهیم.

همانطور که مشاهده میشود یک حلق for داریم که با توجه به اندازه آرایه اولیه ما است که در آن فایل صوتی خود را ذخیره کردیم.

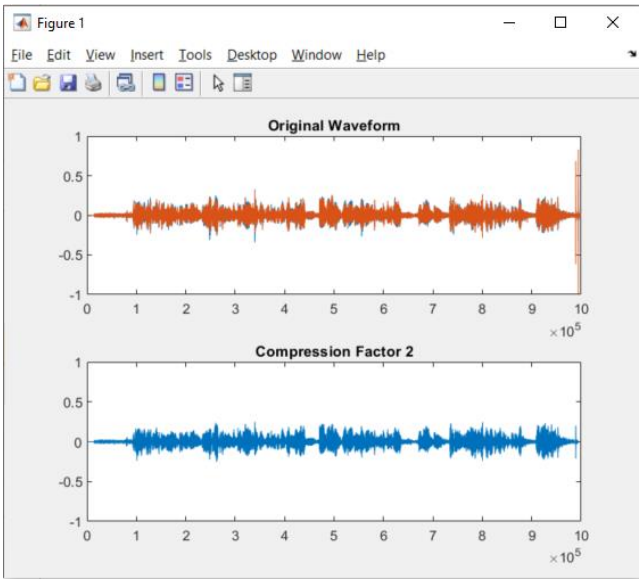
مقدار اضافه شدن این for به اندازه همان قابی است که در ابتدا تعریف نمودیم، قاب تا قاب تا! بر روی آرایه فایل صوتی ما حرکت مکنید.

Dct را صدا میزند ولی بعد از آن معکوس همان را برای آنکه فایل صوتی جدید را بسازد صدا میزند و با توجه به اندازه فشرده شدن فایل خواهد بود و همچنین نتیجه خود dct.

```
%% Plot
figure(1)
h1 = subplot(2, 1, 1);
plot(Voice), title('Original Waveform');
subplot(2, 1, 2)
plot(HalfCompression), title('Compression Factor 2'), axis(axis(h1));
```

پس از گذشتن از مراحل قبل کار اصلی فشرده سازی تمام میشود و در اینجا بیشتر به مقایسه دو فایل صوتی، اصلی و فشرده شده میپردازیم و دیاگرامهای آنها را رسم میکنیم. (در واقع هیستوگرام)

بتدا شکل موج فایل صوتی اصلی و سپس فایل فشرده شده، که به صورت زیر است:



```
%% Save
audiowrite('CompressionFactor2.m4a', HalfCompression,Frequency);
```

در این مرحله نوبت آن رسیده که فایل فشرده شده که آن را توسط الگوریتم dct تولید کردیم را با نام، در همان فولدر که Script را مینویسیم ذخیره کنیم.

```
%% Play
%playing files
disp('Original');
sound(Voice, Frequency);
FirstSize = dir('Sample.m4a');
display(FirstSize.bytes);
pause(20);
disp('Compression Factor 2');
sound(HalfCompression, Frequency);
FirstSize = dir('CompressionFactor2.m4a');
display(FirstSize.bytes);
```

این بخش نهایی است، برای بررسی کیفیت دو فایل صوتی، و مقایسه آنها از نظر صدا.
همچنین اندازه هر دو فایل را برحسب تعداد بایت تشکیل دهنده آنها خواهیم دید، همانند شکل زیر:

```
Original
      514152

Compression Factor 2
      339237
```

همانطور که پیداست، تقریبا فایل دوم تا حد خوبی کاهش حجم داشته است.(دقیقا نصف نشده است، اما کمی کمتر از نصف کاهش حجم فایل صوتی را داشتیم.)

ممنون از توجه شما،

امیدوارم این تمرین و مینی پروژه مورد قبول شما واقع شده باشد.