



دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

بررسی الگوریتم‌های فشرده‌سازی و کاربردهای آنها

نگارنده:

محمد مهدی عرفانیان

درس:

ارائه مطالب علمی و فنی

مدرس:

دکتر همت‌یار

۲۰ دی ۱۳۹۸

چکیده

الگوریتم‌های فشرده‌سازی در طی عمر علوم کامپیوتر یکی از مهم‌ترین موضوعات مورد بحث آن بوده اند. در ابتدا با توجه به کندی و ضعف سخت‌افزار و نیاز به انتقال حداقلی داده این الگوریتم‌ها مورد توجه قرار گرفتند؛ سپس با پیشرفت اینترنت و نیاز به انتقال سریع داده، این الگوریتم‌ها دوباره به یکی از مهم‌ترین زمینه‌های تحقیق در علوم کامپیوتر تبدیل شدند، در این زمان بسیاری از الگوریتم‌های مهم فشرده‌سازی داده کشف شدند. فرمت‌هایی هم‌چون mp3 یا jpeg - که از شناخته‌ترین فرمت‌های فشرده‌سازی حال حاضر هستند - ماحصل تحقیقات همین دوران اند. الگوریتم‌های فشرده‌سازی نه تنها راه خود را به دنیای مدیا باز کردند بلکه در پردازش سیگنال نیز تاثیر عمده‌ای داشتند، تمامی ارتباطات رادیویی با استفاده از این الگوریتم‌ها سعی در بالا بردن امنیت و کاهش هزینه انتقال داده دارند، در رمزنگاری نیز این الگوریتم‌ها راه‌گشایی می‌کنند و بسیاری از الگوریتم‌های رمزنگاری و درهم‌سازی خود نوعی الگوریتم فشرده‌سازی محسوب می‌شوند.

در این مستند سعی شده است تا مقدمه‌ای برای ورود به دنیای شگفت‌انگیز الگوریتم‌های فشرده‌سازی نوشته شود و خوانندگان را با تعاریف بنیادین و کاربردهای اصلی این الگوریتم‌ها آشنا کند، این مستند مقدمتاً درباره تعاریف اصلی و انواع الگوریتم‌های فشرده‌سازی و کاربردهای اصلی آن‌ها در دنیای امروز بحث می‌کند و پس از آن به بررسی جزئی‌تر دو فرمت اصلی فشرده‌سازی (mp3 و jpeg) می‌پردازد. امید است تا مطالب این مستند مورد توجه خوانندگان عزیز قرار گیرد.

کلمات کلیدی: الگوریتم‌های فشرده‌سازی، فشرده‌سازی تصویر، فشرده‌سازی صوت، mp3،

JPEG

فهرست مطالب

۱	مقدمه	۱
۲	تعریف	۱.۱
۲	انواع الگوریتم‌های فشرده‌سازی	۲.۱
۲	الگوریتم‌های بدون هدررفت داده	۱.۲.۱
۴	الگوریتم‌های با هدررفت داده	۲.۲.۱
۴	کاربردهای دیگر فشرده‌سازی	۳.۱
۶	بررسی نحوه فشرده‌سازی در فرمت JPEG	۲
۷	تبدیل کسینوسی گسسته	۱.۲
۷	تعریف	۱.۱.۲
۷	تبدیل دنباله‌ای از پیکسل‌ها به تابع	۲.۲
۷	نحوه کار JPEG	۳.۲
۹	یک مثال از فشرده‌سازی JPEG	۴.۲
۱۰	جدول کوانتیزه کردن	۱.۴.۲
۱۲	فشرده‌سازی هافمن	۵.۲
۱۲	تعریف	۱.۵.۲
۱۵	بررسی نحوه فشرده‌سازی در فرمت ۳MP	۳
۱۶	مقدمه	۱.۳
۱۶	نرخ بیتی	۲.۳
۱۶	نحوه کار ۳mp	۳.۳
۱۷	بررسی دستگاه شنوایی انسان	۴.۳
۱۷	ضعف شنوایی بزرگسالان	۱.۴.۳
۱۷	درک کمتر جزئیات صداها	۲.۴.۳
۱۷	آستانه شنوایی انسان	۳.۴.۳
۱۷	اثر پوشش‌دهی صداها	۴.۴.۳
۱۹	فشرده‌سازی mp3 در عمل	۵.۳
۲۱	مراجع	۴

فهرست تصاویر

۳ شکل نمونه برای تست میزان کمپرس در فرمت PNG	۱.۱
۸ توصیف فرکانسی توابع $\cos(x), \cos(2x)$	۱.۲
۸ توصیف فرکانسی تابع $\cos(x) + \cos(2x)/2$	۲.۲
۹ نمونه تصویر برای فشرده‌سازی	۳.۲
۱۱ ماتریس فرکانس‌های استاندارد	۴.۲
۱۴ مراحل الگوریتم Huffman [۱۱]	۵.۲
۱۸ آستانه شنوایی انسان برای فرکانس‌های مختلف	۱.۳
۱۸ اثر پوشش‌دهی	۲.۳
۲۰ ساختار فایل ۳MP	۳.۳

فهرست جداول

۳	حجم فایل نمونه در فرمت PNG	۱.۱
۴	حجم فایل نمونه در فرمت JPG	۲.۱
۱۲	نوعی نظیرسازی حروف با طول ثابت برای هر حرف	۱.۲
۱۳	نوعی نظیرسازی حروف با طول متغیر برای هر حرف	۲.۲
۱۴	جدول تکرار حروف در یک رشته نمونه	۳.۲

فصل ۱

مقدمه

توضیحی اولیه مبنی بر تعریف کلی فشرده‌سازی، انواع الگوریتم‌ها و کاربردها

۱.۱ تعریف

الگوریتم‌های فشرده‌سازی، الگوریتم‌هایی هستند که می‌توان با استفاده از آن‌ها داده‌ها را طوری رمزنگاری کرد که در تعداد کمتری بیت نسبت به آرایش اولیه قابل ارائه باشند [۱]. برای مثال می‌دانیم که برای ذخیره هر بیت اسکی^۱ هشت بیت فضا لازم است، می‌توان با استفاده از نگاشتی متشکل از حروف استفاده‌شده در یک متن تعداد بیت‌های مورد نیاز برای نشان دادن هر حرف استفاده‌شده در متن را کاهش داد. با استفاده از این تکنیک در حقیقت متن را در قالب جدیدی فشرده کرده‌ایم.

۲.۱ انواع الگوریتم‌های فشرده‌سازی

در یک دسته‌بندی الگوریتم‌های فشرده‌سازی را به دو نوع زیر افراز می‌کنند.

- بدون هدررفت داده^۲
- همراه هدررفت داده^۳

۱.۲.۱ الگوریتم‌های بدون هدررفت داده

در این سری الگوریتم‌ها داده ورودی بدون هیچ‌گونه هدررفتی از داده خروجی قابل بازیابی است، الگوریتم‌های این دسته با استفاده از افزودن آمارهای تلاش می‌کنند تا نحوه نمایش داده را در نگاشتی به نحوه نمایش دیگری که به فضای کمتری نیاز دارد تبدیل کنند. این الگوریتم‌ها در مواقعی که ثابت ماندن داده در طی فشرده‌سازی الزامی است استفاده می‌شوند، همچنین معمولاً برای بازیابی اطلاعات فشرده‌شده نیاز به داده‌هایی خارجی است که با کمک آن عمل بازیابی انجام می‌گیرد، از این رو می‌توان از این نوع الگوریتم‌ها در رمزنگاری نیز استفاده کرد. یکی از کاربردهای اصلی این الگوریتم‌ها در فشرده‌سازی متون است که اشتباه شدن حتی یک حرف می‌تواند باعث بدخوانی و بدفهمی متن اصلی گردد. الگوریتم‌های مشهور فشرده‌سازی بدون هدررفت داده به شرح زیر اند [۲].

- فشرده‌سازی طول متن^۴
- فشرده‌سازی LZ^۵
- کدگذاری هافمن^۶
- تبدیل باروس-ویلر^۷

البته لازم به ذکر است که در عمل از مجموعه‌ای از الگوریتم‌های فوق برای رسیدن به درصد مطلوب فشرده‌سازی استفاده می‌شود.

^۱ ASCII

^۲ Lossless

^۳ Lossy

^۴ Run-Length Encoding (RLE)

^۵ Lempel-Ziv (LZ)

^۶ Huffman Encoding

^۷ Burrows Wheeler Transform

۸	۱	۸	۵	۱	۰	۰	۱	۱	۸
۸	۴	۴	۳	۹	۱	۷	۳	۰	۵
۸	۹	۷	۰	۵	۲	۸	۹	۲	۰
۴	۱	۶	۰	۱	۰	۲	۸	۶	۳
۱	۴	۷	۲	۱	۵	۴	۷	۵	۶
۷	۱	۲	۳	۶	۷	۵	۵	۹	۰
۴	۸	۲	۱	۲	۲	۱	۶	۲	۶
۶	۰	۵	۹	۵	۴	۹	۹	۶	۱
۲	۳	۹	۹	۵	۴	۵	۴	۰	۹
۴	۴	۴	۸	۶	۶	۰	۹	۹	۴

شکل ۱.۱: شکل نمونه برای تست میزان کمپرس در فرمت PNG [۳]

جدول ۱.۱: حجم فایل نمونه در فرمت PNG

Format	Size
BMP	۷.۷ مگابایت
PNG	۹۸ کیلوبایت

نمونه‌های الگوریتم‌های بدون هدررفت داده

در عمل از الگوریتم‌های بدون هدررفت داده در مواقعی که پایداری داده‌های ذخیره‌شده حیاتی‌ست یا این که فایل در آینده به تعداد زیادی بار فشرده و گسترده می‌شود و از دست دادن قسمتی از داده در هربار فشرده‌سازی منجر به اختلاف فاحش نهایی خواهد شد استفاده می‌شود. احتمالاً می‌توان مشهورترین فرمت فشرده‌سازی بدون هدررفت داده را فرمت ZIP نامید، این فرمت که برای فشرده‌سازی فایل‌های مختلف استفاده می‌شود به علت گستردگی استفاده در درون خود از الگوریتم‌های بدون هدررفت داده استفاده می‌کند، به جز فرمت ZIP فرمت‌های دیگری که استفاده‌های تخصصی‌تری دارند نیز از این الگوریتم‌ها استفاده می‌کنند که در ادامه به دو مورد مهم ایشان اشاره می‌کنیم.

• PNG

در طراحی فرمت PNG برای فشرده‌سازی تصاویر از الگوریتم Lempel-Ziv-Welch (LZW) که الگوریتمی Lossless است استفاده شده. در شکل ۱.۱ و جدول ۱.۱ یک نمونه عکس در حالت فشرده‌نشده و فشرده‌شده با فرمت PNG و مقدار حجم آن در حالت‌های مختلف آورده شده است.

• FLAC [4]

کدک صوتی بدون هدررفت داده آزاد^۸ یا FLAC الگوریتمی که با استفاده از اطلاعات ذاتی داده‌های صوتی به فشرده‌سازی آن‌ها می‌پردازد، نرخ فشرده‌سازی این الگوریتم با توجه به سطح فشرده‌سازی سازی آن معمولاً بین ۴۰ تا ۶۰ درصد می‌باشد اما در حالت بیشینه ممکن است تا ۸۰ درصد هم برسد.

^۸ Free Lossless Audio Codec

۲.۲.۱ الگوریتم‌های با هدررفت داده

معمولا در فشرده‌سازی با هدررفت داده از تبدیل فضایی^۹ استفاده می‌شود که داده‌ها را از فضای حقیقی به فضایی دیگر (معمولا فرکانس) می‌برد و در آنجا از قسمت‌هایی از داده که تاثیرگذاری و حس‌پذیری کمتری نسبت به دیگران دارند صرف نظر می‌شود، سپس وارون تبدیل اجرا شده و داده‌های کوچک‌شده جدید این بار با الگوریتم‌های بدون هدررفت داده فشرده می‌شوند. یکی از مشهورترین تبدیل‌های فضایی تبدیل کسینوسی گسسته^{۱۰} است که در فصل دوم این مستند بیشتر با آن آشنا خواهیم شد [۶].

نمونه‌های الگوریتم‌های با هدررفت داده

الگوریتم‌های با هدررفت داده با همه‌گیر شدن اینترنت و اشتراک‌گذاری بیشتر مدیا در فضای اینترنت بسیار گسترده شدند، از این رو اکثر این الگوریتم‌ها برای فشرده‌سازی فایل‌های صوتی-تصویری یا به اصطلاح مدیا^{۱۱} استفاده می‌شوند. فرمت‌های زیر همگی در بطن خود از الگوریتم‌های با هدررفت داده برای فشرده‌سازی مدیا استفاده می‌کنند.

• JPEG

• MP3

• MP4

• H.26x

به عنوان نمونه برای الگوریتم با هدررفت داده حالت فشرده‌شده عکس ۱۰۱ با فرمت JPEG با کیفیت‌های مختلف در جدول ۲.۱ آورده شده است.

جدول ۲.۱: حجم فایل نمونه در فرمت JPG

کیفیت (%)	اندازه	فرمت
۱۰۰	۷.۷ مگابایت	BMP
۱۰۰	۹۸ کیلوبایت	PNG
۹۰	۸.۹۶ کیلوبایت	JPG
۵۰	۹.۶۲ کیلوبایت	JPG
۲۰	۴۹ کیلوبایت	JPG

۳.۱ کاربردهای دیگر فشرده‌سازی

الگوریتم‌های فشرده‌سازی داده علاوه بر این که در کاربرد اصلی خود یعنی فشرده کردن فایل‌ها بسیار استفاده می‌شوند اما کاربردهایی دیگری هم در دنیای کامپیوتر دارند، به عنوان مثال یکی از حیاتی‌ترین

^۹ Transformation

^{۱۰} Discrete Cosine Transform (DCT)

^{۱۱} Media

عناصر پردازش سیگنال^{۱۲} را فشرده‌سازی سیگنال تشکیل می‌دهد. در ارتباطات رادیویی برای بالا بردن امنیت و کم کردن هزینه انتقال از فشرده‌سازی داده استفاده می‌شود. از دیگر کاربردهای جدید فشرده‌سازی داده ژنتیک است، رشته‌های دی‌ان‌ای^{۱۳} و آران‌ای^{۱۴} را معمولاً می‌توان به صورت یک رشته متنی نشان داد و با الگوریتم‌های فعلی فشرده‌سازی متن فشرده کرد اما به دلیل این که این رشته‌ها طول و تعداد زیادی دارند بهتر است از روش‌های دیگری برای بازیابی و فشرده‌سازی سریع آن‌ها استفاده کنیم، با رشد سریع ژنتیک و بایوانفورماتیک در جهان و نیاز مبرم به ذخیره‌سازی رشته‌های دی‌ان‌ای و آران‌ای الگوریتم‌های مختلفی مختص فشرده‌سازی این رشته‌ها معرفی شده‌اند [۱۲].

¹² Signal Processing¹³ DNA¹⁴ RNA

فصل ۲

بررسی نحوه فشرده‌سازی در فرمت JPEG

توضیح نحوه کار فشرده‌سازی در فرمت JPEG، مقدمه‌ای بر تبدیلی کسینوسی گسسته و توضیح کلی کدگذاری هافمن

۱۰.۲ تبدیل کسینوسی گسسته

۱۰.۱.۲ تعریف

تبدیل کسینوسی گسسته^۱ یا مختصراً DCT، دنباله ای محدود از اعداد (داده ها) را به صورت مجموع توابع کسینوسی با فرکانس های متفاوت نمایش می دهد

تبدیل کسینوسی گسسته، شباهت بسیاری به تبدیل فوریه گسسته (DFT) دارد، با این تفاوت که حاصل تبدیل فقط مقادیر حقیقی دارد (بر خلاف تبدیل فوریه که منجر به مقادیر مختلط می شود).

به صورت علمی تر می توان نوشت که DCT تابعی معکوس پذیر و خطی از R^N به R^N است.

فرمول کلی DCT برای فضای یک بعدی به شکل زیر است [۶].

$$X_k = \sum_{n=0}^{N-1} x_n \cos[\pi/n(n+1/2)k], k = 0, \dots, N-1 \quad (1.2)$$

۲.۲ تبدیل دنباله ای از پیکسل ها به تابع

هر تصویر در حقیقت به صورت چهار ماتریس دوبعدی ذخیره می شود که هر ماتریس مقدار رنگی پیکسل را برای آن کانال رنگی (RBG و α) نشان می دهد، برای فشرده کردن یک عکس ماتریس های کانال های رنگی مختلف را جداگانه فشرده می کنیم، کانال رنگی R را در نظر بگیرید، در این حالت یک ماتریس $n*m$ داریم که هر خانه آن عددی بین ۰ تا ۲۵۵ را نشان می دهد، برای سادگی یک سطر از این ماتریس را در نظر بگیرید، این سطر معادل با یک آرایه از اعداد می باشد، در صورتی که تمامی اعداد را از دامنه ۰ تا ۲۵۵ به دامنه ۱۲۸- تا ۱۲۷ ببریم می توانیم این دنباله را با مجموعه ای از توابع کسینوسی بازتولید کنیم، این ضرایب همان ضرایبی ست که با استفاده از DCT می خواهیم به دست بیاوریم.

شکل ۱.۲ نمونه ای از توصیف عکس با توابع کسینوسی را نشان می دهد.

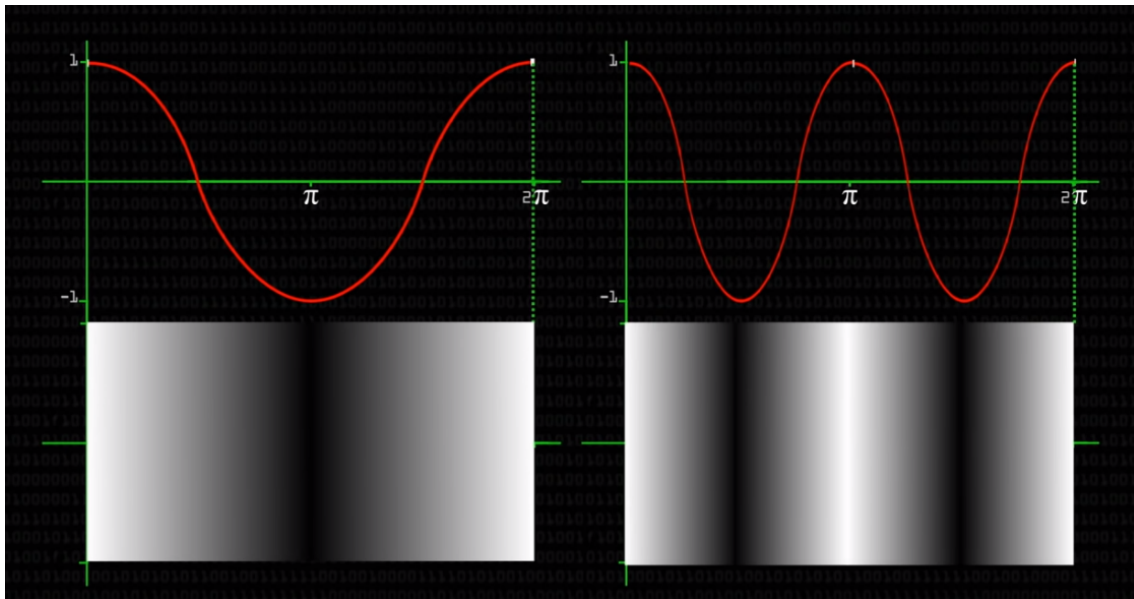
حال اگر دو تابع نشان داده شده در شکل ۱.۲ را با هم جمع کرده و میانگین بگیریم، به نمایش فرکانس دیگری می رسم که در شکل ۲.۲ نشان داده شده است. در صورتی که به شکل های متفاوت و با ضرایب متفاوت توابع مختلف کسینوسی را با هم جمع کنیم می توانیم هر فرکانسی را بسازیم. این کاری ست که در عمل الگوریتم DCT برای ما انجام می دهد.

۳.۲ نحوه کار JPEG

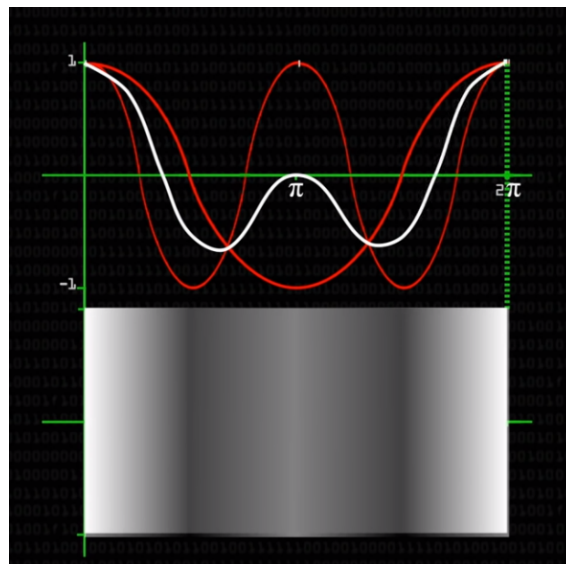
برای فشرده سازی یک تصویر در JPEG مراحل زیر انجام می شود.

- تبدیل تصویر به بلوک های کوچک
- تبدیل هر بلوک به مجموعه ای از توابع کسینوسی استاندارد با DCT
- تنظیم ضرایب متناسب برای بلوک های DCT
- فشرده کردن ضرایب با استفاده از کدگذاری هافمن

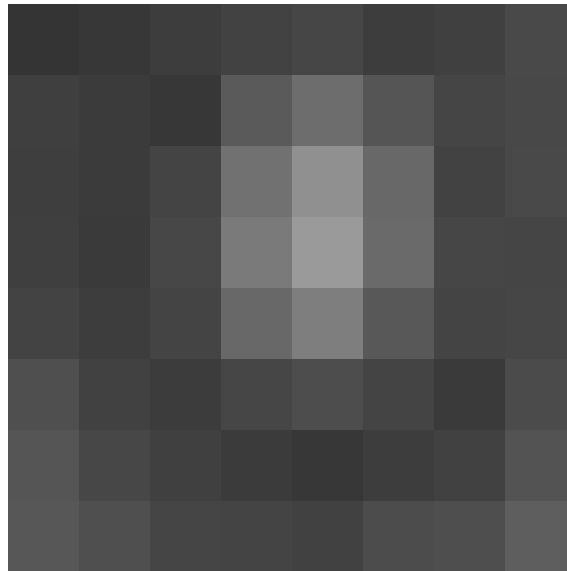
¹ Discrete cosine transform



شکل ۱.۲: توصیف فرکانسی توابع $\cos(x)$, $\cos(2x)$ [۸]



شکل ۲.۲: توصیف فرکانسی تابع $\cos(x) + \cos(2x)/2$ [۸]



شکل ۳.۲: نمونه تصویر برای فشرده‌سازی [۷]

۴.۲ یک مثال از فشرده‌سازی JPEG

برای درک بهتر مراحل گفته شده تلاش می‌کنیم تا ماتریس ۲.۲ - که تصویر کانال خاکستری آن را در شکل ۳.۲ مشاهده می‌شود - را با الگوریتم‌های گفته‌شده فشرده‌سازی کنیم.

$$M = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 60 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix} \quad (2.2)$$

هر عنصر ماتریس ۲.۲ عددی بین ۰ تا ۲۵۵ دارد اما از آنجایی که تابع کسینوسی مقادیر بین ۱- تا ۱ می‌گیرد نیازمندیم تا با کم کردن هر عنصر این ماتریس از ۱۲۸ هر عنصر این ماتریس را به عددی بین ۱۲۸- تا ۱۲۷

تبدیل کنیم. ماتریس ۳.۲ تبدیل شده است.

$$M' = \begin{bmatrix} -55 & -64 & -67 & -58 & -62 & -67 & -73 & -76 \\ -56 & -59 & -43 & -19 & -38 & -73 & -69 & -65 \\ -55 & -62 & -24 & 16 & -15 & -60 & -69 & -66 \\ -59 & -58 & -22 & 26 & -6 & -57 & -70 & -65 \\ -58 & -60 & -40 & -2 & -24 & -60 & -67 & -61 \\ -53 & -70 & -60 & -51 & -58 & -68 & -63 & -49 \\ -45 & -63 & -67 & -73 & -69 & -64 & -57 & -43 \\ -34 & -50 & -52 & -63 & -60 & -59 & -49 & -41 \end{bmatrix} \quad (3.2)$$

در گام بعدی مقادیر ماتریس تبدیل شده را با استفاده از الگوریتم DCT به فضای فرکانس می بریم هر مقدار از این ماتریس جدید برابر با ضریب فرکانس معادل در ماتریس استاندارد است. ماتریس ۴.۲ ماتریس تبدیل شده ما خواهد بود.

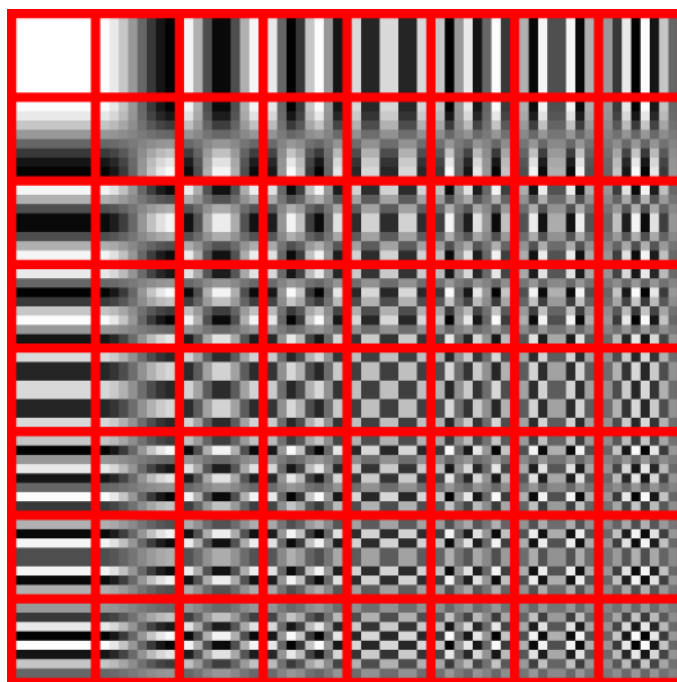
$$F = \begin{bmatrix} 0/46 & -2/39 & -20/10 & 56/12 & 27/24 & -61/20 & -30/19 & -415/38 \\ 4/88 & -8/54 & -7/09 & 13/15 & 10/25 & -60/76 & -21/28 & 4/47 \\ -5/65 & 5/42 & 9/93 & -28/91 & -24/56 & 77/13 & 7/37 & -46/83 \\ 1/95 & 1/83 & 6/30 & -10/24 & -14/76 & 34/10 & 12/07 & -48/53 \\ 3/14 & -2/79 & 1/75 & -1/87 & -3/95 & -13/20 & -6/55 & 12/12 \\ 1/85 & 4/30 & 0/94 & -2/38 & -5/94 & 2/38 & 2/91 & -7/73 \\ -0/66 & 4/12 & -3/02 & -0/88 & -2/42 & 0/42 & 0/18 & -1/03 \\ 1/68 & 0/50 & -0/10 & -1/17 & -4/19 & -1/07 & 0/14 & -0/17 \end{bmatrix} \quad (4.2)$$

هر کدام از عناصر ماتریس ۴.۲ مقدار ضریب تاثیر فرکانس نشان داده شده در شکل ۴.۲ می باشند. تا به این لحظه هیچ مقداری از داده را برای فشرده سازی از دست نداده ایم اما در این گام می خواهیم قسمت هایی از تصویر که برای چشم انسان قابل تشخیص نیستند را حذف کنیم تا مقدار داده کمتری را ذخیره کنیم. باید توجه داشت که چشم انسان معمولاً قادر به تشخیص و تمییز تصاویر با فرکانس های بالا در تصاویر نیست و همان طور که در ماتریس ۴.۲ هم مشاهده می شود ضریب این فرکانس ها نسبت به فرکانس های پایین بسیار کم است، می توانید به مقدار بسیار بزرگ ۴۱۵ برای فرکانس بسیار پایین در راستای x و y در ماتریس توجه کنید تا این نکته روشن شود. حال باید ضرایب فعلی ماتریس را با تقریبی گرد کنیم و تاثیر فرکانس های بالا را کمتر از تاثیر فرکانس های پایین قرار دهیم، از این رو از جدولی به نام جدول کوانتیزه کردن استفاده می کنیم.

۱.۴.۲ جدول کوانتیزه کردن

جدول کوانتیزه کردن^۲ جدولی است که در آن مقدار تاثیر هر فرکانس برای هر رده فشرده سازی برای عکس های JPEG به صورت جهانی استاندارد سازی و تعیین شده است. برای مثال برای نرخ فشرده سازی ۵۰ درصد جدول کوانتیزه کردن در ماتریس ۵.۲ آمده است. این ماتریس برای کیفیت های مختلف فشرده سازی

^۲ معادل فارسی برای Quantization یافت نشد.



شکل ۴.۲: ماتریس فرکانس‌های استاندارد [۹]

به صورت استاندارد شده وجود دارد.

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad (5.2)$$

حال برای آخرین گام باید مقادیر ماتریس ۴.۲ را بر مقادیر ماتریس ۵.۲ تقسیم کنیم و عدد به دست آمده را گرد کنیم. ماتریس نهایی برای فشرده‌سازی به شکل زیر است.

$$R = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (6.2)$$

همان طور که در ماتریس ۶.۲ مشهود است تعداد بسیار زیادی از عناصر ماتریس جدید مقدار صفر دارند (که بیشتر فرکانس های بالا را شامل می شوند) در ادامه فشرده سازی ماتریس ۶.۲ به وسیله الگوریتم فشرده سازی بدون هدررفت داده هافمن فشرده می شود و به همراه مقداری داده افزوده^۳ مانند درصد فشرده سازی و... ذخیره می شود.

۵.۲ فشرده سازی هافمن

برای این که ماتریس مرحله آخر را به یک رشته متنی تبدیل کنیم می خواهیم از فشرده سازی یا الگوریتم هافمن استفاده کنیم. در این قسمت مختصراً الگوریتم هافمن برای فشرده سازی یک رشته متنی شرح داده می شود.

۱.۵.۲ تعریف

برای نمایش رشته در حالت اسکی^۴ برای هر حرف ۸ بیت فضا گرفته می شود و تمامی حروف با یک آرایه هشت بیتی نمایش داده می شوند. رشته زیر را در نظر بگیرید.

$$S = \text{bananasc}$$

برای نمایش این رشته در حالت اسکی به $64 = 8 * 8$ بیت فضا نیاز داریم، در صورتی که بخواهیم از طراحی با طول بیت ثابت برای هر حرف در این رشته استفاده کنیم می توانیم نظیرسازی جدول ۱.۲ را در نظر بگیریم.

جدول ۱.۲: نوعی نظیرسازی حروف با طول ثابت برای هر حرف

نماد	حرف
۰۰۱	a
۰۱۰	b
۰۱۱	n
۱۰۰	s
۱۰۱	c

در این روش برای نشان دادن هر حرف به سه بیت فضا نیاز داریم و در کل به $24 = 3 * 8$ بیت فضا نیاز داریم.

اما در صورتی که به صورت دقیق تر به رشته نگاه کنیم متوجه می شویم که تعداد تکرار هر حرف یکسان نیست و می توانیم برای نشان دادن هر حرف از تعداد بیت متغیر استفاده کنیم به شکلی که حروفی که بیشتر تکرار شده اند تعداد بیت کمتری بگیرند و حروفی که کمتر تکرار شده اند از تعداد بیت بیشتری برای ذخیره سازی استفاده کنند. مثلاً نظیرسازی جدول ۲.۲ را برای این رشته در نظر بگیرید.

³ Meta data

⁴ ASCII

جدول ۲.۲: نوعی نظیرسازی حروف با طول متغیر برای هر حرف

نماد	حرف
۰	a
۱۱۰	b
۱۰	n
۱۱۱۰	s
۱۱۱۱	c

در صورتی که رشته اصلی را با این نظیرسازی فشرده کنیم به بیت‌های زیر می‌رسیم.

۱۱۰۰۱۰۰۱۰۰۱۱۱۰۱۱۱۱

همان‌طور که در رشته جدید مشهود است تعداد بیت‌های استفاده‌شده برای نمایش رشته اصلی به ۱۸ بیت کاهش پیدا کرده، نکته اساسی در این فشرده‌سازی این است که رشته به صورت یکتا قابل بازیابی باشد، در صورتی که به رشته بالا دقت کنیم متوجه می‌شویم که تنها حالتی که می‌توان با توجه به حروف جدول برای بازیابی بیت‌ها متصور شد همین حالتی است که به رشته اصلی منجر می‌شود.

۱۱۰, ۰, ۱۰, ۰, ۱۰, ۰, ۱۱۱۰, ۱۱۱۱

این اتفاق به این دلیل رخ می‌دهد که شروع هیچ حرفی زیرمجموعه پیشوندی هیچ رشته دیگری نیست، برای مثال در صورتی که حرف a با ۰ و حرف b با ۱۰ و حرف n با ۱۰ نمایش داده می‌شدند برای رشته بیتی ۰۱۰ دو حالت an و b می‌توانستیم متصور شویم، علت این اتفاق این است که رشته حرف a زیرمجموعه حرف b است.

کلیت الگوریتم هافمن پیدا کردن بهترین کدگذاری برای هر حرف در رشته اصلی است که طول رشته نهایی کم‌ترین اندازه را داشته باشد، برای این کار از شبه کد زیر استفاده می‌شود [۱۰].

```

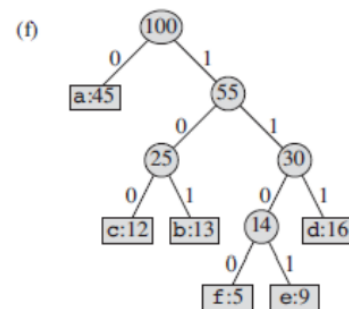
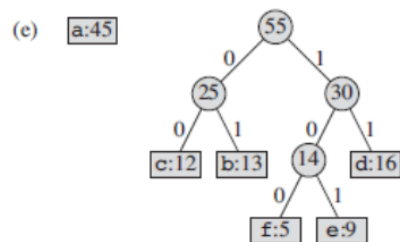
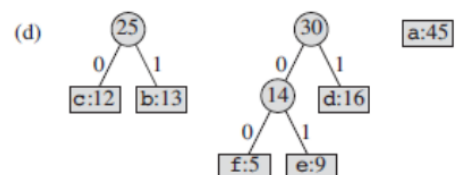
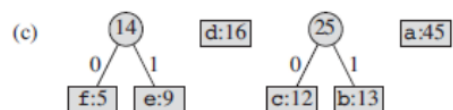
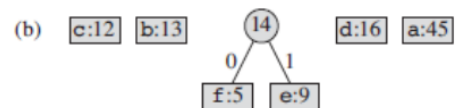
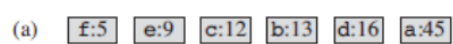
1 //Huffman Algorithm
2 n := |C|;
3 Q := C;
4 for i := 1 to n - 1 do
5     allocate a new node z
6     z.left := x := Extract-Min(Q);
7     z.right := y := Extract-Min(Q);
8     z.freq := x.freq + y.freq;
9     Insert(Q, z);
10 end for
11 return Extract-Min(Q); {return the root of the tree}

```

شکل ۵.۲ مراحل الگوریتم هافمن را برای یک رشته با تعداد تکرار در جدول ۳.۲ نشان می‌دهد.

جدول ۳.۲: جدول تکرار حروف در یک رشته نمونه

تکرار	حرف
۵	f
۹	e
۱۲	c
۱۳	b
d	۱۶
a	۴۵



شکل ۵.۲: مراحل الگوریتم هافمن [۱۱]

فصل ۳

بررسی نحوه فشرده‌سازی در فرمت

MP3

توضیح نحوه کار فشرده‌سازی در فرمت MP3، تحلیل صدای انسان و توضیح کانال‌های فرکانسی

۱۰.۳ مقدمه

در اواخر قرن بیستم و با شروع فراگیری اینترنت در جوامع مختلف، اشتراک‌گذاری مدیاهای مختلف مانند صوت، فیلم و تصویر در فضای اینترنت به یکی از خواست‌های همگانی و نیازهای اصلی مردم تبدیل شد اما مشکل اصلی در این میان، حجم زیاد فایل‌های مدیا و سرعت پایین انتقال داده در فضای اینترنت بود، فرمت‌های فشرده‌سازی مختلفی برای حل این معضل پیشنهاد شدند که هر کدام با تکیه بر تحلیل‌های آماری و یا حذف داده‌هایی غیرضروری در راستای کم کردن حجم فایل‌های مدیا می‌کوشیدند. در زمینه فشرده‌سازی فایل‌های صوتی فرمت ۳MP یقیناً فراگیرترین و موفق‌ترین فرمت به شمار می‌رود. در این فصل نگاهی کلی به نحوه کار این فرمت و الگوریتم‌های با هدررفت داده به کاررفته در این فرمت خواهیم داشت. برای تقریب اذهان به بزرگی حجم فایل‌های صوتی فشرده‌ناشده لازم به توجه است که هر فایل صوتی در حالتی که هیچ مقداری از داده فشرده نشود در هرثانیه حدود ۱۷۶۰۰۰ بایت فضا می‌گیرد.

۲۰.۳ نرخ بیتی

برای درک بهتر مفاهیمی که در ادامه متن با آن‌ها سر و کار داریم لازم است تا ابتدا با مفهوم نرخ بیتی^۱ آشنا شویم، در مفاهیم علوم کامپیوتر، نرخ بیتی معمولاً مقدار بیتی‌ست که در برای واحد زمانی ذخیره می‌شود، مثلاً وقتی bitrate یک موسیقی 320 kb/s اعلام می‌شود به این معنی‌ست که برای هر ثانیه از این صوت ۳۲۰ کیلوبیت داده ذخیره شده است؛ همچنین نرخ بیتی می‌تواند سرعت انتقال داده در یک کانال را بیان کند، مثلاً در طراحی مودم‌های شبکه از این مفهوم برای نمایش سرعت انتقال داده استفاده می‌شود.

۳۰.۳ نحوه کار ۳mp

تمرکز فرمت ۳mp در فشرده‌سازی بر حذف اصواتی‌ست که توسط گوش انسان شنیده نمی‌شوند، همان‌طور که از فصل دو به یاد دارید برای فشرده‌سازی تصاویر می‌توانستیم از بیت‌هایی که نمایانگر تصاویری با فرکانس بالا بودند صرف نظر کنیم زیرا توسط چشم انسان قابل تشخیص نبودند، در فشرده‌سازی صوت نیز می‌توانیم با مطالعه ساختار شنوایی انسان اصواتی که به طور معمول توسط گوش انسان شنیده نمی‌شود را از صوت حذف می‌کنیم تا حجم فایل کاهش یابد.

به شکل خلاصه می‌توان مراحل فشرده‌سازی در فرمت ۳mp را به شکل زیر خلاصه کرد.

- تبدیل صوت به قسمت‌های کوچک
- حذف صوت‌های خارج از محدوده شنوایی انسان
- نمونه‌برداری از موسیقی با توجه به نرخ بیتی خواسته شده
- اضافه کردن افزونه‌ها و فشرده‌سازی نمونه

^۱bitrate

۴.۳ بررسی دستگاه شنوایی انسان

برای فشرده‌سازی صوت باید در ابتدا اصواتی که توسط گوش انسان قابل شنیدن نیستند یا گوش انسان با جزییات کمتری آن‌ها را درک می‌کند را حذف کنیم، برای این کار به تحلیل صوت‌شناسی^۲ نیاز دارد. نتایج تحلیل‌های صوت‌شناسی برای دستگاه شنوایی انسان موارد مفید زیر را اعلام می‌کند.

- ضعف شنوایی بزرگسالان

- درک کمتر جزییات صداهای کم

- آستانه شنوایی انسان

- اثر پوشش‌دهی صداهای بلند

به تفصیل موارد بالا و کاربردهای آنان در فشرده‌سازی را بررسی خواهیم کرد.

۱.۴.۳ ضعف شنوایی بزرگسالان

در کودکی انسان‌ها معمولاً می‌توانند اصواتی را که بین فرکانس‌های ۱۰ تا ۲۰ کیلوهرتز باشند را بشنوند، اما با بزرگ شدن انسان معمولاً حد بالای شنوایی به مقدار ۱۵ کیلوهرتز می‌رسد و در صورتی که به درصد فشرده‌سازی بالایی نیاز داشته باشیم می‌توانیم از فرکانس‌های بیش از ۱۵ کیلوهرتز صرف نظر کنیم.

۲.۴.۳ درک کمتر جزییات صداهای کم

دستگاه شنوایی انسان معمولاً به جزییات صداهای بلند بیش از صداهای آرام توجه می‌کند، در نتیجه می‌توانیم برای صداهایی که سطح فشار صوت^۳ کمتری دارند از bitrate پایین‌تری استفاده کنیم.

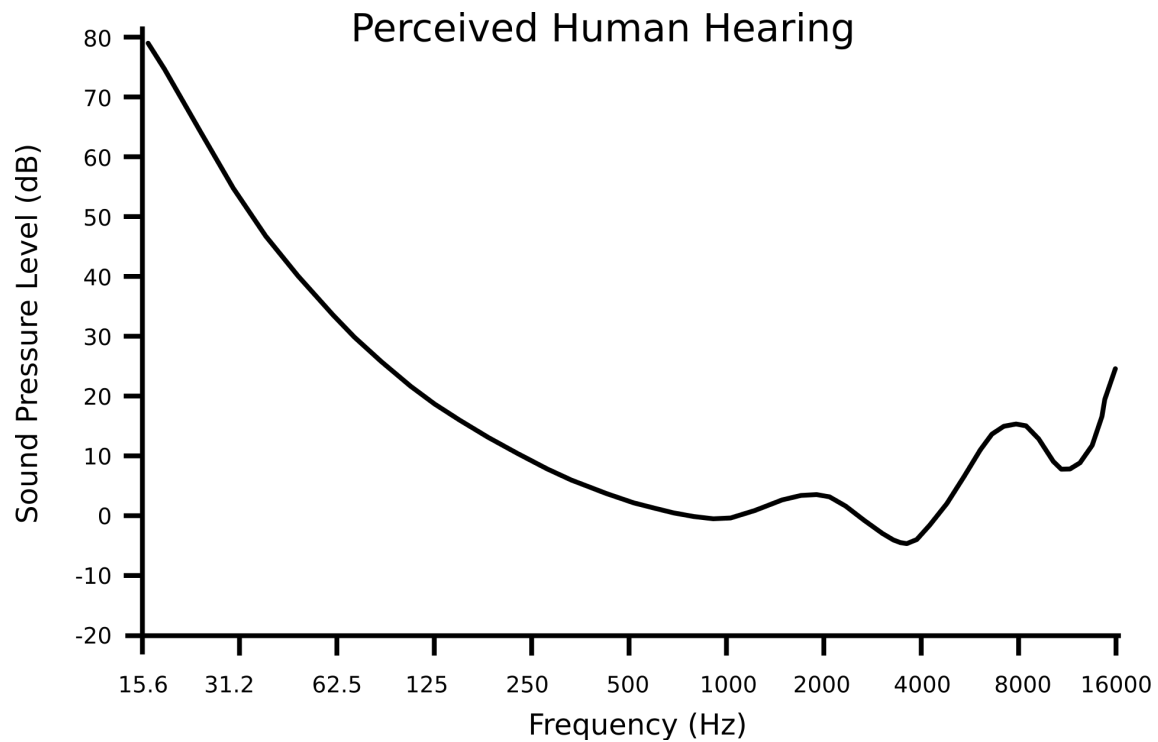
۳.۴.۳ آستانه شنوایی انسان

گوش انسان برای هر فرکانس صوتی آستانه شنوایی دارد که اگر سطح فشار صدا از آن کمتر باشد آن را نمی‌شنود، شکل ۱.۳ این آستانه را برای فرکانس‌های مختلف نشان می‌دهد؛ در صورتی که سطح صوتی فرکانسی در هر قسمت از مقدار آستانه آن کمتر باشد آن صدا شنیده نمی‌شود و می‌توان آن را حذف کرد.

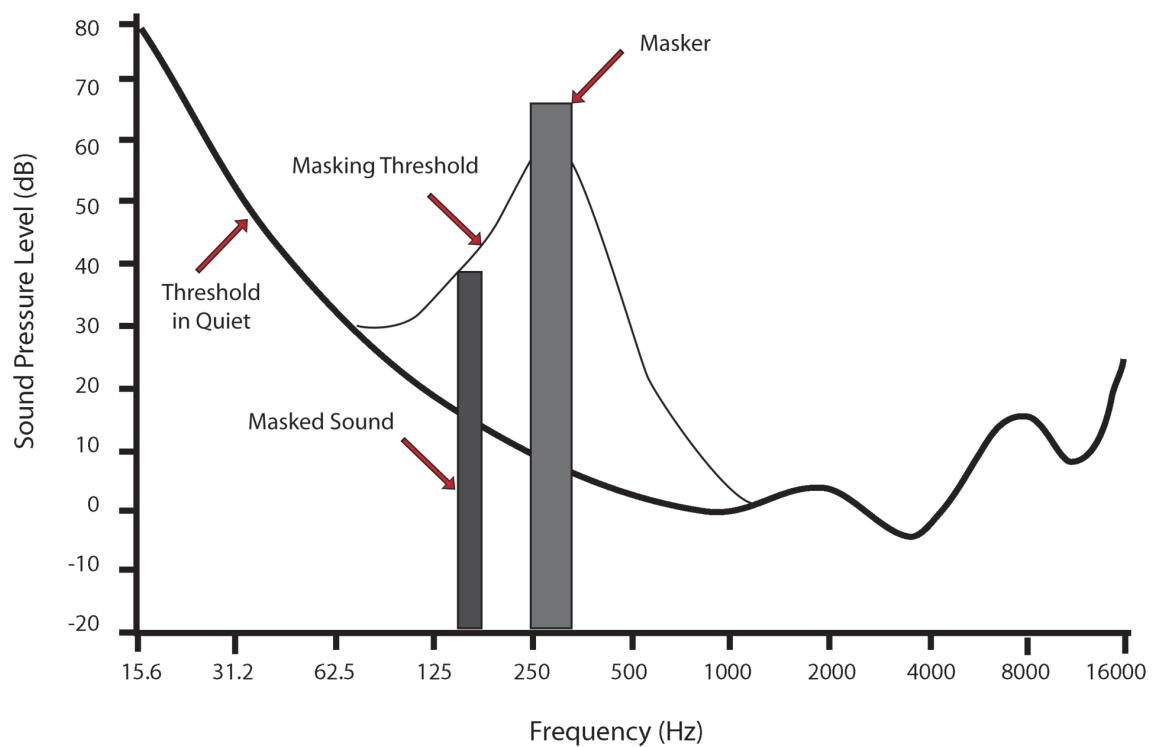
۴.۴.۳ اثر پوشش‌دهی صداهای بلند

هنگامی که در یک محدوده فرکانسی یک صدای بلند رخ دهد گوش انسان نمی‌تواند صداهای آرام با فرکانس‌های نزدیک به آن را تشخیص دهد و بشنود حتی اگر سطح صوتی آن از آستانه شنوایی انسان بیشتر باشد، به این اثر در گوش انسان اثر پوشش‌دهی^۴ گفته می‌شود. مثالی از اثر پوشش‌دهی در شکل ۲.۳ نشان داده شده است.

Psychoacoustics^۲
Sound Pressure Level (db)^۳
Masking Effect^۴



شکل ۱.۳: آستانه شنوایی انسان برای فرکانس‌های مختلف



شکل ۲.۳: اثر پوشش‌دهی

۵.۳ فشرده سازی mp3 در عمل

پس از بررسی انتزاعی اتفاقاتی که برای فشرده سازی در فرمت ۳mp میافتد نیازمند آنیم تا با یک مثال جزئیات فنی پیاده سازی این فرمت را نیز بررسی کنیم.

هنگامی که یک صوت برای فشرده شدن انتخاب می شود ابتدا به تکه های صوتی کوچک تر تقسیم یم شود و با استفاده از نسخه اندکی تغییر یافته الگوریتم DCT که در فصل دوم معرفی شد به فضای فرکانس برده می شود، پس از آن اصواتی که فرکانس هایی پایین تر از آستانه شنوایی انسان دارند حذف می شوند و همین طور برای هر کانال صوتی عامل پوشش دهنده ^۵ و عوامل پوشش پذیر ^۶ شناسایی می شوند و بیت های مربوط به عوامل پوشش پذیر حذف می شوند، همچنین برای صداهای بلند مقداری از جزئیات اصوات آرام تر را حذف می کنیم زیرا توسط ذهن انسان تشخیص داده نمی شوند.

پس از این تغییرات دوباره به فضای زمان برمی گردیم تا نمونه گیری را با توجه به bitrate خواسته شده انجام دهیم؛ دقت کنید که تا این جای کار هیچ مقداری از بیت هایی که توسط دستگاه شنیداری انسان قابل شنیدن باشند از دست نرفته اند.

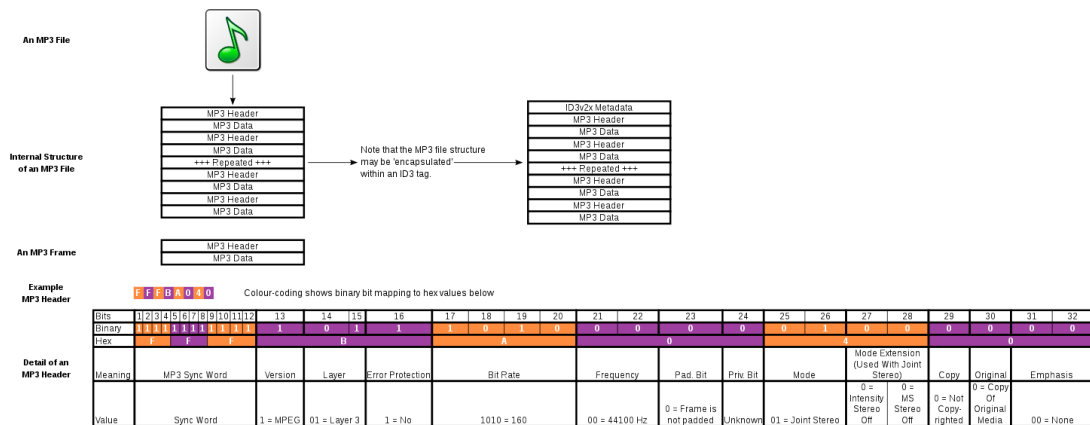
در گام سوم با توجه به bitrate از هر قسمت کوچک ساخته شده نمونه برداری می کنیم و سپس به گام آخر می رسد، در این گام اطلاعاتی که برای بازیابی هر قسمت مورد نیاز است به همراه تعدادی بیت که برای تشخیص خطا قرار می گیرند را در Header هر قسمت قرار می دهیم و سپس داده اصلی را قرار می دهیم. مختصراً می توان گفت که هر بلوک داده موارد زیر را در خود دارد.

• سرتیتر

- کد همگام سازی
- نسخه
- لایه صوتی
- بیت تشخیصی خطا
- نرخ بیتی
- فرکانس
- بیت padding
- بیت خصوصی
- حالت صوت
- داده کپی شده
- داده اصلی
- بیت تاکید

• داده صوتی

موارد بالا به صورت جدولی در شکل ۳.۳ آورده شده اند.



شکل ۳.۳: ساختار فایل MP۳

٤ مراجع

- [1] Mahdi, O.A.; Mohammed, M.A.; Mohamed, A.J. (Marhc 2013) *Implementing a Novel Approach an Convert Audio Compression to Text Coding via Hybrid Technique*. International Journal of Computer Science Issues.
- [2] Pujar, J.H.; Kadlaskar, L.M. (May 2010). "A New Lossless Method of Image Compression and Decompression Using Huffman Coding Techniques" (PDF). Journal of Theoretical and Applied Information Technology.
- [3] Mahdi, E: <https://github.com/merfanian/DataCompressionDoc/blob/master/LatexFiles/figs/compressed.png>
- [4] Web Archive, <https://web.archive.org/web/20090202063734/http://synthetic-soul.co.uk/comparison/lossless/index.asp>
- [5] Arcangel, Cory. (March 2013) *On Compression*
- [6] Ahmed, Nasir (January 1991). *How I Came Up With the Discrete Cosine Transform*
- [7] By en>User:Cburnett - Own work in Inkscape based on the following data:, CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=3333955>
- [8] Computerphile, https://www.youtube.com/watch?v=n_uNPbdenRs
- [9] Wikipedia, <https://commons.wikimedia.org/wiki/File:Dctjpeg.png>
- [10] Huffman, D. (1952). "A Method for the Construction of Minimum-Redundancy Codes"
- [11] Radu Trîmbițaș (2012), *Huffamn Codes*
- [12] Stéphane Grumbach, FarizaTahi (November 1994), *A new challenge for compression algorithms: Genetic sequences*