



دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

بررسی الگوریتم‌های فشرده‌سازی و کاربردهای آنها

نگارندگان:
محمد مهدی عرفانیان

درس:
ارائه مطالب علمی و فنی

مدرس:
دکتر همت‌یار

۱۷ 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 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 را 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 مقدماً 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.

به عنوان موخره چکیده باید نوشت که این مستند در راستای انجام پروژه مستندسازی درس ارائه مطالب علمی و فنی تهیه شده است و برای سهولت کار استاد محترم درس برای تحصیل اطمینان از درستی مستندسازی و همچنین استفاده دانش‌جویان علاقه‌مند، سیر پیشرفت مستند به همراه کدهای L^AT_EX در Github قرار گرفته اند، لازم به ذکر است که این مستند به صورت متن‌باز ارائه شده و استفاده از آن بدون ذکر منبع برای همگان آزاد است.

کلمات کلیدی: الگوریتم‌های فشرده‌سازی، فشرده‌سازی تصویر، فشرده‌سازی صوت، MP3

JPEG

فهرست مطالب

۱	مقدمه	۱
۲	۱.۱ تعریف	۲
۲	۲.۱ انواع الگوریتم‌های فشرده‌سازی	۲
۲	۱.۲.۱ الگوریتم‌های بدون هدررفت داده	۲
۴	۲.۲.۱ الگوریتم‌های با هدررفت داده	۴
۶	۲ بررسی نحوه فشرده‌سازی در فرمت JPEG	۶
۷	۱.۲ DCT	۷
۷	۱.۱.۲ تعریف	۷
۷	۲.۲ تبدیل دنباله‌ای از پیکسل‌ها به تابع	۷
۷	۳.۲ نحوه کار JPEG	۷
۹	۴.۲ یک مثال از فشرده‌سازی JPEG	۹
۱۱	۱.۴.۲ Quantization Table	۱۱
۱۲	۵.۲ فشرده‌سازی Huffman	۱۲
۱۲	۱.۵.۲ تعریف	۱۲
۱۵	۳ بررسی نحوه فشرده‌سازی در فرمت MP۳	۱۵
۱۶	۱.۳ مقدمه	۱۶
۱۶	۲.۳ نرخ بیتی	۱۶
۱۶	۳.۳ نحوه کار mp۳	۱۶
۱۷	۴.۳ بررسی دستگاه شنوایی انسان	۱۷
۱۷	۱.۴.۳ ضعف شنوایی بزرگسالان	۱۷
۱۷	۲.۴.۳ درک کمتر جزئیات صداهای کم	۱۷
۱۷	۳.۴.۳ آستانه شنوایی انسان	۱۷
۱۷	۴.۴.۳ اثر پوشش‌دهی صداهای بلند	۱۷
۱۹	۵.۳ فشرده‌سازی mp3 در عمل	۱۹
۲۱	کتابنامه	۲۱

فهرست تصاویر

۳	۱.۱	شکل نمونه برای تست میزان کمپرس در فرمت PNG
۸	۱.۲	توصیف فرکانسی توابع $\cos(x)$, $\cos(2x)$ [۸]
۸	۲.۲	توصیف فرکانسی تابع $\cos(x) + \cos(2x)/2$ [۸]
۹	۳.۲	نمونه تصویر برای فشرده‌سازی [۷]
۱۰	۴.۲	ماتریس فرکانس‌های استاندارد [۹]
۱۴	۵.۲	مراحل الگوریتم Huffman [۱۱]
۱۸	۱.۳	آستانه شنوایی انسان برای فرکانس‌های مختلف
۱۸	۲.۳	اثر پوشش‌دهی
۲۰	۳.۳	ساختار فایل MP۳

فهرست جداول

۳	حجم فایل نمونه در فرمت PNG	۱.۱
۵	حجم فایل نمونه در فرمت JPG	۲.۱
۱۲	نوعی نظیرسازی حروف با طول ثابت برای هر حرف	۱.۲
۱۲	نوعی نظیرسازی حروف با طول متغیر برای هر حرف	۲.۲

فصل ۱

مقدمه

توضیحی اولیه مبنی بر تعریف کلی فشردہ سازی، 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 افزونگی آماری 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 کمپرس بدون هدررفت داده به شرح زیر اند. [۲]

- Run-Length Enconding (RLE)
- Lempel-Ziv (LZ)
- Huffman Encoding
- Burrows Wheeler Transform

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

ASCII^۱
Lossless^۲
Lossy^۳

```

      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.

• Free Lossless Audion Codec (FLAC) [4]

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.

۲.۲.۱ الگوریتم‌های با هدررفت داده

در این الگوریتم‌ها پس از هر بار فشرده‌سازی مقداری از داده‌ها از دست می‌روند، معیار ارزیابی این الگوریتم‌ها مقدار فشرده‌سازی با توجه به میزان هدررفت داده می‌باشد، به علت هدررفت مقداری از داده این الگوریتم‌ها معمولاً در مواردی که هدررفت اندک داده توسط انسان یا ماشین قابل تشخیص نباشد استفاده می‌شوند، مثلاً بعضی از تکنیک‌های ذخیره‌سازی تصاویر و ویدئوها در کامپیوترها مبتنی بر الگوریتم‌های با هدررفت داده اند زیرا چشم انسان قادر به تشخیص عوض شدن تعدادی پیکسل در صفحه پس از بازیابی فایل فشرده‌شده نیست [۵]. همچنین به علت این که برای فشرده‌سازی از مقداری از داده‌ها صرف نظر می‌شود این الگوریتم‌ها نرخ فشرده‌سازی بیشتری نسبت به الگوریتم‌های بدون هدررفت داده دارند.

معمولاً در فشرده‌سازی با هدررفت داده از تبدیل فضایی^۴ استفاده می‌شود که داده‌ها را از فضای حقیقی به فضایی دیگر (معمولاً 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 فضایی‌ها تبدیل کسینوسی گسسته^۵ است که در فصول آتی 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^۶ استفاده می‌شوند.

- 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

Format	Size	Quality(%)
BMP	۷.۷ مگابایت	۱۰۰
PNG	۹۸ کیلوبایت	۱۰۰
JPG	۸.۹۶ کیلوبایت	۹۰
JPG	۹.۶۲ کیلوبایت	۵۰
JPG	۴۹ کیلوبایت	۲۰

^۴ Transformation
^۵ Discrete Cosine Transform (DCT)
^۶ Media

فصل ۲

بررسی نحوه فشرده‌سازی در فرمت JPEG

توضیح نحوه کار فشرده‌سازی در فرمت JPEG، مقدمه‌ای بر DCT و توضیح کلی Huffman encoding

۱.۲ DCT

۱.۱.۲ تعریف

تبدیل کسینوسی گسسته (به انگلیسی: Discrete cosine transform)، 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

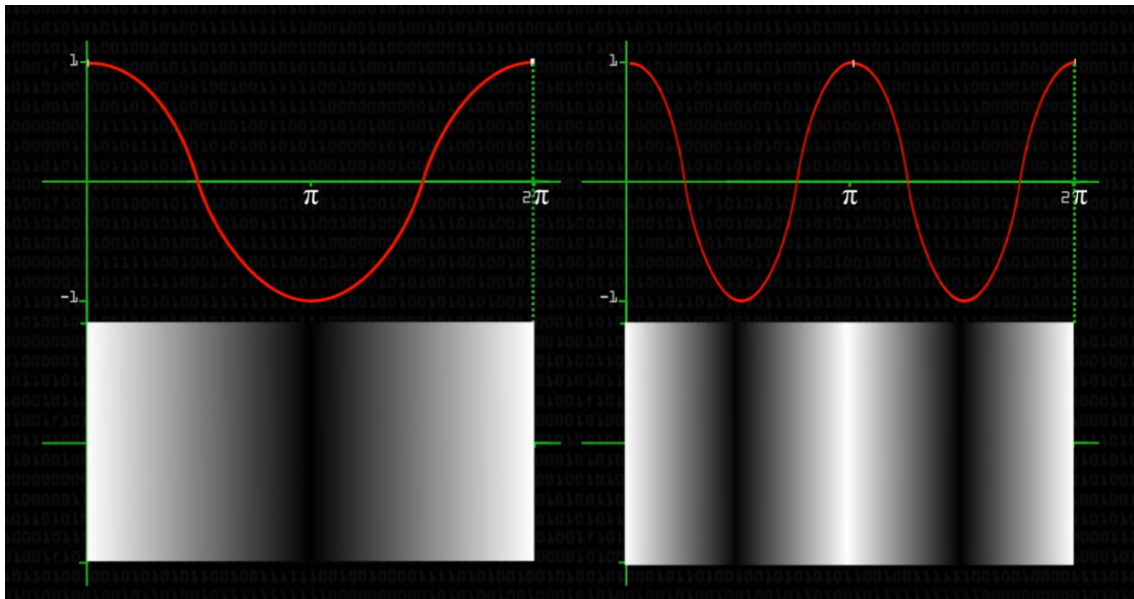
هر تصویر در حقیقت به صورت چهار ماتریس دوبعدی ذخیره می شود که هر ماتریس مقدار رنگی پیکسل را برای آن کانال رنگی (RBG و α) نشان می دهد، برای کمپرس کردن یک عکس ماتریس های کانال های رنگی مختلف را جداگانه فشرده می کنیم، کانال رنگی R را در نظر بگیرید، در این حالت یک ماتریس $n*m$ داریم که هر خانه آن عددی بین 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 - 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 رسیم 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.

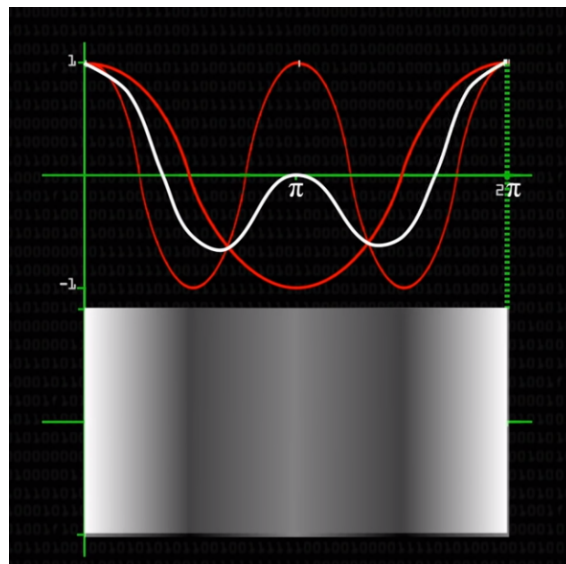
۳.۲ نحوه کار JPEG

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

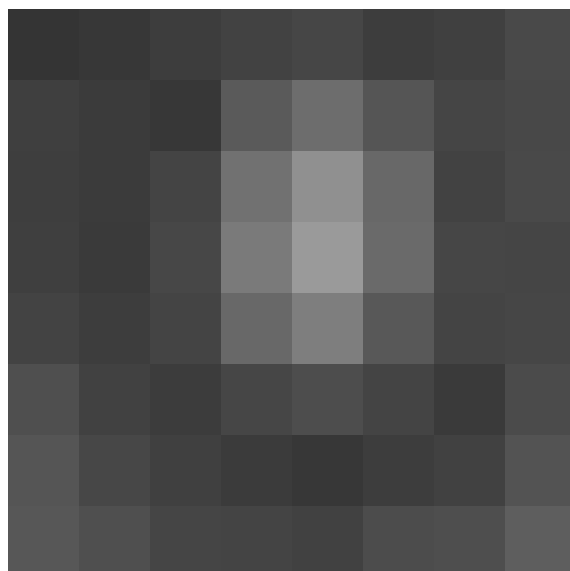
- تبدیل تصویر به بلوک های کوچک
- تبدیل هر بلوک به مجموعه ای از توابع کسینوسی استاندارد با DCT
- تنظیم ضرایب متناسب برای بلوک های DCT
- کمپرس کردن ضرایب با استفاده از Huffman



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



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



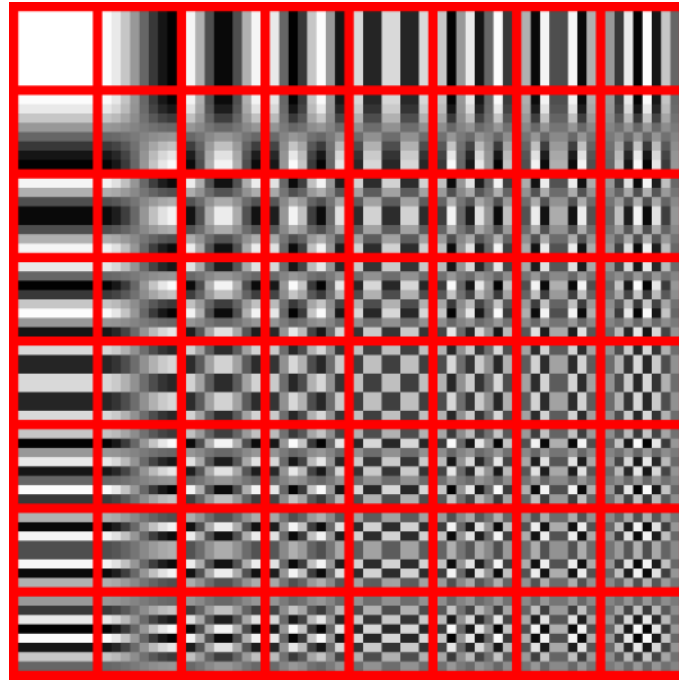
شکل ۳.۲: نمونه تصویر برای فشرده‌سازی [۷]

۴.۲ یک مثال از فشرده‌سازی JPEG

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

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

هر عنصر این ماتریس 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 ۱۲۷ - تا ۱۲۸ تبدیل کنیم. ماتریس تبدیل‌شده به شکل زیر است.



شکل ۴.۲: ماتریس فرکانس‌های استاندارد [۹]

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

در گام بعدی مقادیر این ماتریس را با استفاده از الگوریتم DCT به فضای فرکانس می‌بریم هر مقدار از این ماتریس جدید برابر با ضریب فرکانس معادل در ماتریس استاندارد است.

$$F = \begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.28 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix}$$

هر کدام از عناصر ماتریس بالا مقدار ضریب تاثیر فرکانس نشان داده شده در شکل ۴.۲ می‌باشند. تا به این لحظه هیچ مقداری از داده را برای فشرده‌سازی از دست نداده‌ایم اما در این گام می‌خواهیم

قسمت‌هایی 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 تمیز تصاویر با فرکانس‌های بالا در تصاویر نیست و همان‌طور که در ماتریس هم مشاهده می‌شود ضریب این فرکانس‌ها نسبت به فرکانس‌های پایین بسیار کم است، می‌توانید 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 فرکانس‌های بالا را کمتر از تاثیر فرکانس‌های پایین قرار دهیم، از این رو از جدولی به نام Quantization Table استفاده می‌کنیم.

۱.۴.۲ Quantization Table

Quantization table جدولی است که در آن مقدار تاثیر هر فرکانس برای هر رده فشرده‌سازی برای عکس‌های JPEG به صورت جهانی استانداردسازی و تعیین شده است. برای مثال برای نرخ کمپرس ۵۰ درصد Quantization Table به شکل زیر است.

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

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

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

همان‌طور که در ماتریس R مشهود است تعداد بسیار زیادی از عناصر ماتریس جدید مقدار صفر دارند (که بیشتر فرکانس‌های بالا را شامل می‌شوند) در ادامه فشرده‌سازی این ماتریس به وسیله الگوریتم فشرده‌سازی بدون هدررفت داده Huffman فشرده می‌شود و به همراه مقداری داده افزوده^۱ مانند درصد فشرده‌سازی و... ذخیره می‌شود.

^۱ meta data

۵.۲ فشرده‌سازی Huffman

برای این که ماتریس مرحله آخر را به یک رشته متنی تبدیل کنیم می‌خواهیم از فشرده‌سازی یا الگوریتم Huffman استفاده کنیم. در این قسمت مختصراً الگوریتم Huffman برای فشرده‌سازی یک رشته متنی شرح داده می‌شود.

۱.۵.۲ تعریف

برای نمایش رشته در حالت ascii برای هر حرف ۸ بیت فضا گرفته می‌شود و تمامی حروف با یک آرایه هشت‌بیتی نمایش داده می‌شوند. رشته زیر را در نظر بگیرید.

$$S = \text{bananasc}$$

در این رشته در حالت ascii به $8 * 8 = 64$ بیت فضا نیاز داریم، در صورتی که بخواهیم از طراحی با طول بیت ثابت برای هر حرف در این رشته استفاده کنیم می‌توانیم نظیرسازی زیر 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 $8 * 3 = 24$ بیت فضا نیاز داریم.

اما در صورتی که به صورت دقیق‌تر به رشته نگاه کنیم متوجه می‌شویم که تعداد تکرار هر حرف یکسان نیست و می‌توانیم برای نشان دادن هر حرف از تعداد بیت متغیر استفاده کنیم به شکلی که حروفی که بیشتر تکرار شده‌اند تعداد بیت کمتری بگیرند و حروفی که کمتر تکرار شده‌اند از تعداد بیت بیشتری برای ذخیره‌سازی استفاده کنند. مثلاً نظیرسازی زیر را برای این رشته در نظر بگیرید.

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

نماد	حرف
۰	a
۱۱۰	b
۱۰	n
۱۱۱	s
۱۱۱۰	c

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

110010001001111110

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، 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 ۰۱۰ و حرف n با ۱۰ نمایش داده می‌شدند برای رشته ۰۱۰ دو حالت an و b می‌توانستیم متصور شویم، علت این اتفاق این است که رشته حرف a زیرمجموعه حرف b است.

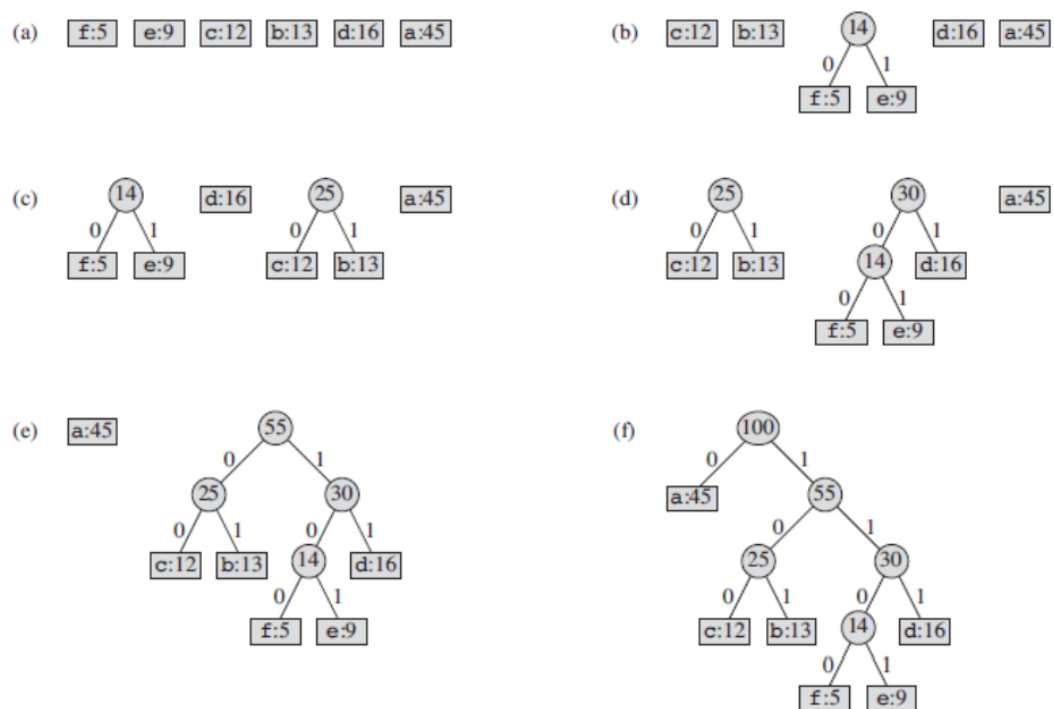
کلیت الگوریتم Huffman پیدا کردن بهترین کدگذاری برای هر حرف در رشته اصلی است که طول رشته نهایی کم‌ترین اندازه را داشته باشد، برای این کار از شبه کد زیر استفاده می‌شود. [۱۰]

```

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 Huffman را نشان می‌دهد.



شکل ۵.۲: مراحل الگوریتم Huffman [۱۱]

فصل ۳

بررسی نحوه فشرده‌سازی در فرمت

MP۳

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

۱.۳ 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 MP۳ 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.

۲.۳ 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.

۳.۳ نحوه کار mp۳

تمرکز فرمت mp۳ 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 mp۳ 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

^۱ bitrate

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 ۱۰۰ 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.

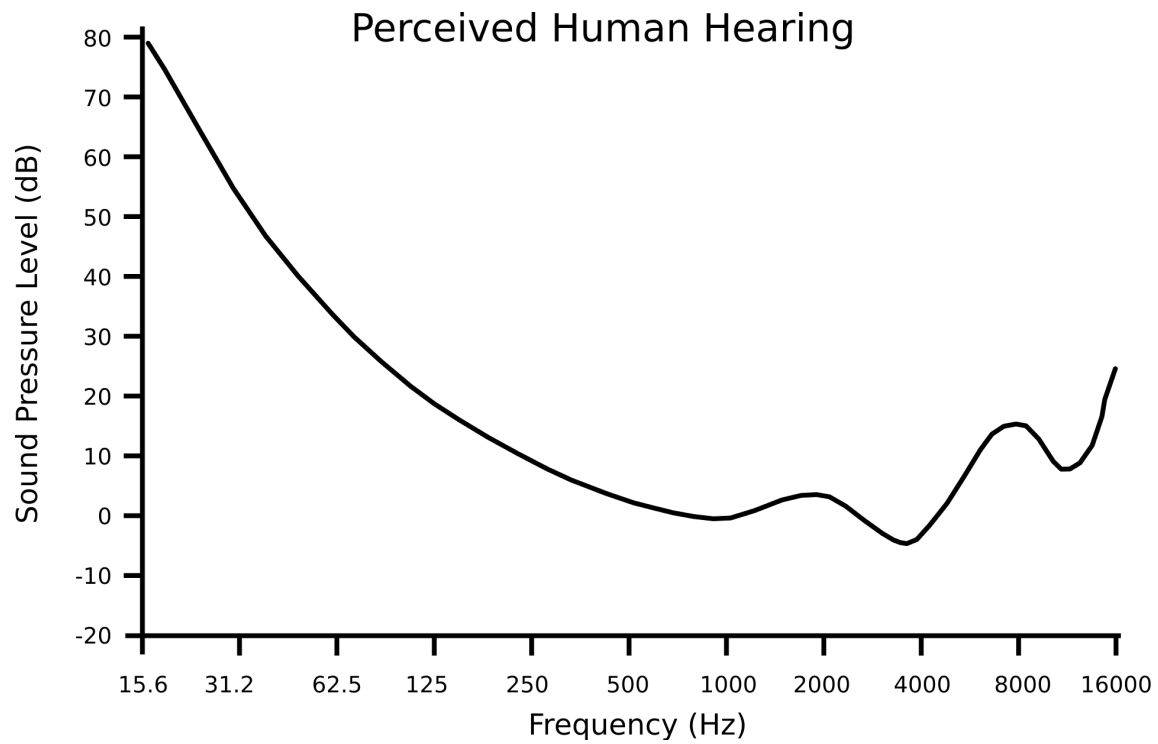
۳.۴.۳ آستانه شنوایی انسان

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.

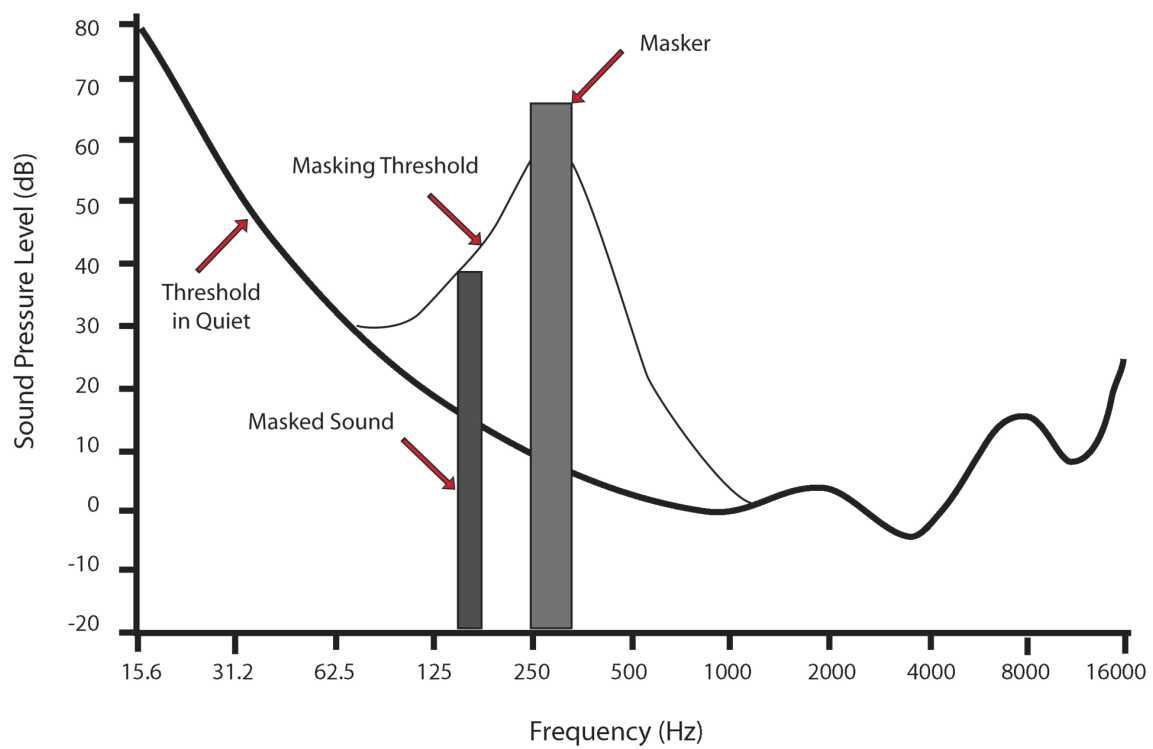
۴.۴.۳ اثر پوشش‌دهی صداهای بلند

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 فشرده‌سازی mp3 در عمل

پس از بررسی انتزاعی اتفاقاتی که برای فشرده‌سازی در فرمت mp۳ میافتد نیازمند آنیم تا با یک مثال جزئیات فنی پیاده‌سازی این فرمت را نیز بررسی کنیم.

هنگامی که یک صوت برای فشرده شدن انتخاب می‌شود ابتدا به تکه‌های صوتی کوچک‌تر تقسیم می‌شود و با استفاده از نسخه‌اندکی تغییر یافته الگوریتم DCT که در فصل دوم معرفی شد به فضای فرکانس برده می‌شود، پس از آن اصواتی که فرکانس‌هایی پایین‌تر از آستانه شنوایی انسان دارند حذف می‌شوند و همین‌طور برای هر کانال صوتی عامل پوشش‌دهنده ۵ و عوامل پوشش‌پذیر ۶ شناسایی می‌شوند و بیت‌های مربوط به عوامل پوشش‌پذیر حذف می‌شوند، همچنین برای صداها با بلند مقداری از جزئیات اصوات آرام‌تر را حذف می‌کنیم زیرا توسط ذهن انسان تشخیص داده نمی‌شوند.

پس از این تغییرات دوباره به فضای زمان برمی‌گردیم تا نمونه‌گیری را با توجه به bitrate خواسته شده انجام دهیم؛ دقت کنید که تا این‌جا کار هیچ مقداری از بیت‌هایی که توسط دستگاه شنیداری انسان قابل شنیدن باشند از دست نرفته‌اند.

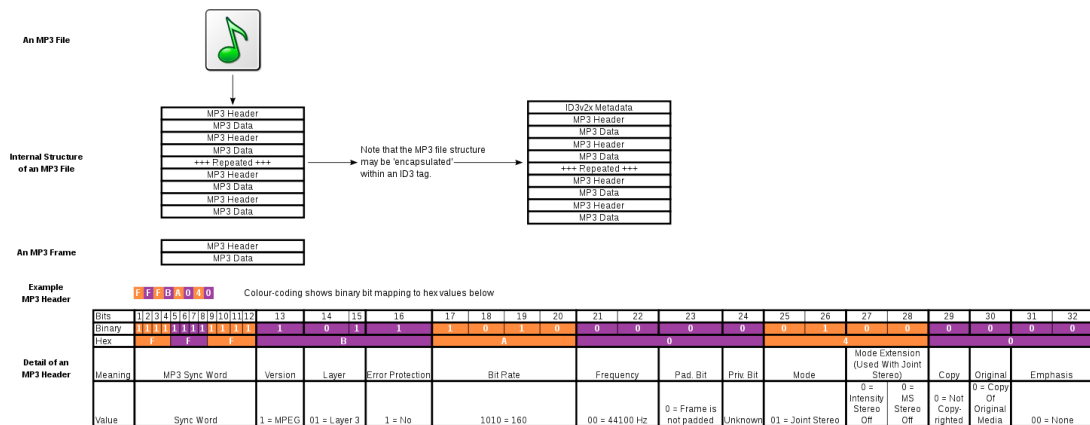
در گام سوم با توجه به bitrate از هر قسمت کوچک ساخته شده نمونه برداری می‌کنیم و سپس به گام آخر می‌رسد، در این گام اطلاعاتی که برای بازیابی هر قسمت مورد نیاز است به همراه تعدادی بیت که برای تشخیص خطا قرار می‌گیرند را در Header هر قسمت قرار می‌دهیم و سپس داده اصلی را قرار می‌دهیم. مختصراً می‌توان گفت که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

• سرتیتر

- کد همگام‌سازی
- نسخه
- لایه صوتی
- بیت تشخیص خطا
- نرخ بیتی
- فرکانس
- بیت padding
- بیت خصوصی
- حالت صوت
- داده کپی‌شده
- داده اصلی
- بیت تاکید

• داده صوتی

موارد بالا به صورت جدولی در شکل ۳.۳ آورده شده‌اند.



شکل ۳.۳: ساختار فایل MP۳

کتاب نامه

Mahdi, O.A.; Mohammed, M.A.; Mohamed, A.J. (Marhc 2013) *Implementing a [۱]
Novel Approach an Convert Audio Compression to Text Coding via Hybrid Technique.*

International Journal of Computer Science Issues.

Pujar, J.H.; Kadlaskar, L.M. (May 2010). "A New Lossless Method of Image Com- [۲]
pression and Decompression Using Huffman Coding Techniques" (PDF). Journal of
Theoretical and Applied Information Technology.

Mahdi, E: [۳]
[https://github.com/
merfanian/DataCompressionDoc/blob/master/LatexFiles/figs/compressed.png](https://github.com/merfanian/DataCompressionDoc/blob/master/LatexFiles/figs/compressed.png)

Web Archive, [۴]
[https://web.archive.org/web/20090202063734/http://synthetic-
soul.co.uk/comparison/lossless/index.asp](https://web.archive.org/web/20090202063734/http://synthetic-soul.co.uk/comparison/lossless/index.asp)

Arcangel, Cory. (March 2013) *On Compression* [۵]

Ahmed, Nasir (January 1991). *How I Came Up With the Discrete Cosine Transform* [۶]

By en>User:Cburnett - Own work in Inkscape based on the following data:, CC BY- [۷]
SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=3333955>

Computerphile, https://www.youtube.com/watch?v=n_uNPbdenRs [۸]

Wikipedia, <https://commons.wikimedia.org/wiki/File:Dctjpeg.png> [۹]

Huffman, D. (1952). "A Method for the Construction of Minimum-Redundancy [۱۰]
Codes"

Radu Trîmbițaș (2012), *Huffamn Codes* [۱۱]