کد گذاری VLC صوت و سیگنال صحبت باند وسیع کدشده با استفاده از خواص روان شنیداری سیستم شنوایی و تبدیل موجک

محسن هاشمی
دانشکده‌ی مهندسی برق و کامپیوتر دانشگاه شهید بهشتی
mohsen.hashemi.sbu@googlemail.com

محمد حسن ساوجی
دانشکده‌ی مهندسی برق و کامپیوتر دانشگاه شهید بهشتی
m-savoji@sbu.ac.ir

چکیده: در این مقاله به بررسی کدگذاری VLC (Variable Length Coding) برای فشرده سازی صوت و سیگنال صحبت باند وسیع پرداخته‌ایم. برای این منظور از روش کدگذاری آنتروپی شامل کدگذاری هافمن و حسابی به صورت ایستا و وفقی و همچنین کدگذاری Run-Length استفاده کرده ایم. در اینجا سعی شده است که مشابه JPEG، جداول هافمن ویژه فایل‌های صوتی ایجاد شود و همچنین از الگوریتم حسابی نیز استفاده گردد. نتایج حاصل حاکی از بهبود حدود ٪ ۸ در نرخ بیت ارسال اطلاعات با استفاده از کدگذاری حسابی و بهبود حدود ٪ ۶/۵ با استفاده از کدگذاری هافمن، نسبت به حالتی که از جداول JPEG استفاده شود، می باشد.

کلمات کلیدی: فشرده سازی صحبت، مدل روان شنیداری، تبدیل موجک، کدگذاری هافمن، کدگذاری حسابی، کدگذاری وفقی.

۱- مقدمه

امروزه با ترافیک سنگین ارتباطات چند رسانه ای از طریق خطوط اینترنت و موبایل و حجم زیاد فایل های صوتی و تصویری و از طرف دیگر با محدودیت پهنای باند کانال و حجم حافظه روبرو هستیم. فشرده سازی داده های چند رسانه ای با در نظر گرفتن محدودیت کانال تنها راه حل برای حل این مشکلات است. صدای انسان بر خلاف سایر امواج صوتی مثل موسیقی (که احتیاج به پهنای باند ۱۵ تا ۲۲KHz برای داشتن کیفیت خوب دارند) با پهنای باند در حد ۷KHz کیفیت مطلوبی دارد- کیفیتی در حد کنفرانس ویدئویی - بنابراین کدرهایی که مخصوص صحبت هستند توانایی فشرده سازی بیشتری دارند.

مرحله اول در طراحی کدر، تهیه سیگنال ورودی مناسب است. سیگنال ورودی در کدر داده های ۱۶ بیتی صدای انسان است که با فرکانس نمونه برداری 16^{KHz} از صدای با پهنای باند 7^{KHz} نمونه برداری شده اند. به این ترتیب صدای ورودی دارای مشخصات صحبت باند وسیع و با کیفیت بالا است. دراین مقاله به طور خاص به شرح کدگذاری VLC کدر طراحی شده در مرجع [1] و [۲] می پردازیم. در این پروژه سعی شده است به جای استفاده از جداول JPEG در کدر مذکور، جداول ویژه‌ی صوت برای کدگذاری VLC ایجاد گردد و همچنین استفاده از کدگذاری حسابی و کدگذاری وفقی نیز بررسی شود. در شکل ۱ بلوک دیگرام کلی کدر طراحی شده نشان داده شده است. در قسمت بعد طور خلاصه به شرح کاری که در مرجع [1] و [۲] انجام شده است می پردازیم.

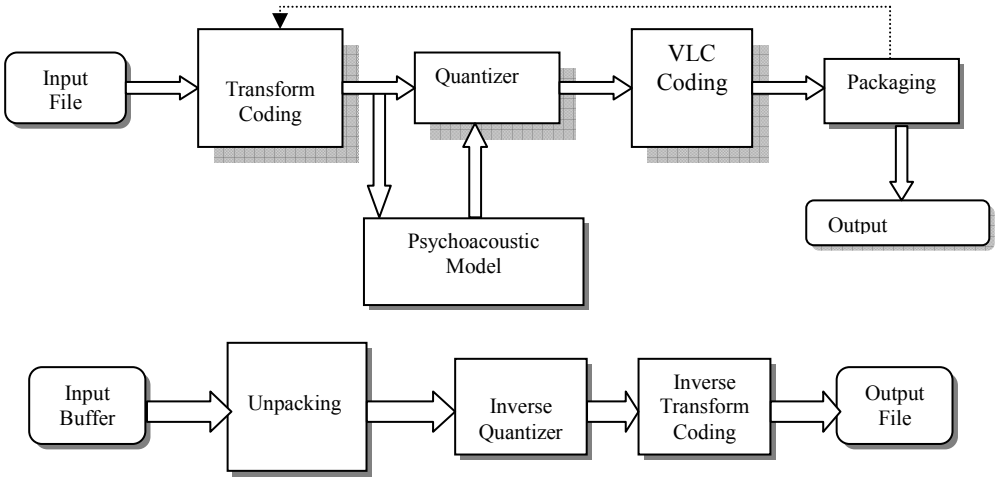
۲-کدگذاری تبدیلی و خواص روان شنیداری

اولین بلوک کدر بلوک کدگذاری تبدیلی است. در اینجا از تبدیلی استفاده می شود تا سیگنال را از حوزه زمان به حوزه دیگری نگاشت کند تا در این حوزه سیگنال به طور مؤثری از نظر فشرده سازی توصیف شود. یکی از این تبدیل ها تبدیل موجک است که سیگنال را به تعداد محدودی ضرایب پر انرژی و تعداد زیادی ضرایب کوچک و نزدیک به صفر تبدیل می کند[3-5]. اگر فقط از ضرایب بزرگ و پرانرژی در هنگام بازسازی سیگنال استفاده شود سیگنال بازسازی شده دارای کیفیتی نزدیک به سیگنال اولیه خواهد بود زیرا قسمت اعظم انرژی سیگنال در همین تعداد محدود ضرایب بزرگ نهفته است. بلوک بعدی بلوک چندی کننده است. چندی کننده ضرایب حقیقی خروجی کدگذار تبدیلی را به اعداد صحیح تبدیل می کند تا با تعداد محدودی بیت بیان شوند. روشن است که در این قسمت مقداری از اطلاعات از بین می‌رود و ضرایب کوچک کدگذار تبدیلی به صفر تبدیل می شوند. وظیفه تخصیص‌دهی بیت به ضرایب خروجی کدگذار تبدیلی بر عهده مدل روان شنیداری است. این مدل از خواص شنوایی گوش انسان مثل پوشیدگی تُن‌های ضعیف مجاور تُن‌های قوی‌تر صدا و باند-های بحرانی شنوایی استفاده می کند[6,7]. مدل روان شنیداری، منحنی ماسک را در باندهای بحرانی شنوایی به دست می‌دهد. با استفاده از این منحنی سطح نویز حاصل از چندی کردن که گوش انسان قادر به شنیدن آن نیست به دست می‌آید. مدل روان شنیداری با محاسبه نسبت سیگنال به ماسک در باندهای مختلف میزان بیت اختصاص داده شده به هر نمونه در هر باند بحرانی را به دست آورده و

در اختیار بلوک چندی کننده قرار می‌دهد. داده های خروجی چندی کننده وارد قسمت کدگذاری VLC می‌شوند.

۳- کدگذاری VLC

وظیفه این قسمت اختصاص کدهایی با طول متغیر به اعداد مختلف برای فشرده سازی بیشتر آنهاست. در این قسمت بدون از دست دادن اطلاعات به فشرده سازی بیشتری دست پیدا می‌کنیم. روش های مختلفی برای پیاده سازی این قسمت وجود دارد[4,8]. از آنجا که همیشه با تعداد زیادی نمونه صفر مواجه هستیم. برخی از کدگذارهای VLC به جای کدکردن تک تک نمونه های صفر به تعداد نمونه‌های صفر قبلی و خانواده عدد غیرصفر که باید کد شود کلمه‌کدی اختصاص می دهند. این خاصیت که Run_Length نام دارد باعث فشرده سازی بیشتری می‌شود. در این مقاله پیاده سازی کدگذاری VLC به چند روش مختاف بررسی شده‌است.



شکل (۱): بلوک دیاگرام کلی کدر :انکدر و دیکدر

۳-۱ کدگذاری با جداول JPEG

در کار انجام شده در مرجع [1] و [۲] برای پیاده سازی این قسمت از کدهای استاندارد JPEG استفاده شده است. این کدها دارای دو خاصیت اصلی کدگذاری آنتروپی و کدگذاری Run_Length هستند و باعث فشرده سازی خوبی می‌شوند. روش کار به این صورت است که اعداد به خانواده های مختلف تقسیم شده‌اند. جدول ۱ یازده عنصر اول و خانواده‌های آنها را که در ساخت پیش‌کدها بکاررفته اند نشان می-دهد. هر پیش‌کد JPEG به تعداد صفرهای قبلی و خانواده عدد غیر صفر کدشونده اشاره می کند. کدها علارغم اینکه دارای طول متغیری هستند به خاطر دارا بودن خاصیت پیشوندی به هنگام استخراج رقم های باینری اشتباهاً به جای یکدیگر استخراج نمی‌شوند. حداکثر تعداد صفری که این کدها قبول می کنند ۱۵ است. این جدول ۱۱ عنصر اول جدول JPEG را که مورد استفاده قرار گرفته است، نشان می دهد. (البته یک کد خاص برای ۱۶ صفر یعنی عدد صفری که قبل از آن ۱۵ صفر قرار دارد وجود دارد که در اینجا مورد استفاده قرار نگرفته است.) بنابراین در حدود ۱۵۰ پیش‌کد مختلف در جدولJPEG پیش بینی شده‌است. بعد از هر پیش‌کد نمایش باینری عدد کدشونده قرار می-گیرد. اگر چه این عدد باینری دارای خاصیت پیشوندی نیست اما به خاطر اینکه خانواده عدد توسط کد پیشوندی قبل از آن مشخص می شود طول کلمه کد نهایی مشخص بوده و هنگام استخراج توسط دیکدر درست استخراج می‌شود. برای توسعه این کلمات‌کد از دو روش زیر استفاده شده‌است تا کدهای استاندارد JPEG انطباق بهتری با کدر مورد استفاده داشته باشد[۲, 1]:

۱- در حالتی که تعداد صفرها بیشتر از ۱۵باشد از یک کد ۱۴ بیتی برای نشان دادن تعداد صفرها استفاده می‌شود. برای تشخیص این نوع کلمه‌کد خاص از دیگر کدها، قبل از آن کد ۴ بیتی '1010'، که همیشه برای نشان دادن انتهای کدگذاری قطعه سیگنال از آن استفاده می شود قرار می‌گیرد. از آنجا که طول قطعه سیگنال به همراه دیگر اطلاعات کناری به دیکدر ارسال می شود در صورتی که دیکدر کد مزبور را در جایی غیر از انتهای قطعه سیگنال ببیند متوجه وجود عدد ۱۴ بیتی تعداد صفرها بعد از این کد می‌شود. در باندهای بالا که تعداد زیادی صفر وجود دارد این روش کدگذاری کمک بزرگی در فشرده سازی بیشتر می‌کند.

۲- گاهی اوقات اعدادی بزرگ در ضرایب خروجی تبدیل بسته موجکی ایجاد می‌شوند که خارج از ۱۱ خانواده اولیه اعداد JPEG قرار می‌گیرند. برای رفع این مشکل پیش‌کدهایی برای یک خانواده اعداد دیگر (خانواده ۱۲) ایجاد و به جدول پیش-کدهای قبلی اضافه شده‌است.

اطلاعات کدر به این ترتیب کامل می‌شود: با افزودن یک سری اطلاعات کناری مثل طول قطعه فعلی سیگنال ، کد هسته موجکی که

استفاده شده است، بردار تعداد بیت‌های تخصیص داده‌شده به نمونه-های ۱۹ باند بحرانی استفاده شده و...، داده‌ها بسته بندی می شوند. این داده‌ها توسط دیکدر باز شده و استخراج می‌شوند و با انجام عکس کارهایی که در قسمت انکدر صورت گرفته است قطعه سیگنال باز سازی می‌شود.

جدول(۱): خانواده‌های مختلف اعداد در جدول استاندارد JPEG

Category	Coefficient table
0	0
1	-1,1
2	-3,-2 2,3
3	-7,…-4 4,…7
4	-15,…-8 8,…15
5	-31,…-16 16,…,31
6	-63,…,-32 32.., 63
7	-127,…-64 64,…127
8	-255,…-128 128,…255
9	-51,…- 256 256,…,511
A	-1023,…,-512 512,…,1023

۳-۲ کدگذاری VLC با استفاده از آمار استخراج شده از فایلهای صوتی

با توجه به اینکه جداول استفاده شده در مراجع [1] و [۲] ویژه‌ی تصاویر JPEG تهیه شده‌اند، طبیعی است که برای فشرده‌سازی صوت عملکرد بهینه‌ای نداشته باشند. به همین دلیل سعی شد کد گذاری VLC با توجه به کدهای ناشی از آمار استخراج شده از خود فایلهای صوتی پیاده سازی شوند.

در این مرحله برای پیاده سازی بخش کدگذار آنتروپی کدگذار VLC از دو الگوریتم مختلف کدگذاری هافمن و کدگذاری حسابی استفاده شده است.

در اینجا هر یک از این دو روش به کار گرفته‌شده را شرح می‌دهیم و در قسمت بعد با استفاده از نتایج بدست‌آمده، این دو روش را از لحاظ میزان و سرعت فشرده‌سازی با هم مقایسه می‌کنیم.

الف- کدگذاری هافمن[9] با استفاده از جداول حاصل از آمار استخراج‌شده از فایلهای صوتی

برای بدست آوردن جداول هافمن ویژه‌ی صوت لازم بود که ابتدا اطلاعات آماری مورد نیاز بدست آیند. در ابتدا این آمار از فایلهای غیر استاندارد فارسی و انگلیسی و موسیقی بدست‌آمد و جداول حاصل از این آمار در کدگذاری فایلهای فارسی و انگلیسی مورد استفاده و بررسی قرارگرفت. اما با توجه به اینکه این اطلاعات آماری باید به گونه‌ای باشند که بتوان از آنها با یک دقت قابل قبول برای توصیف آماری فایلهای صوتی مختلف استفاده کرد، لذا سعی شد که این آمار از فایل‌های دادگان‌های استاندارد مورد استفاده در پردازش صوت، استخراج شود. این آمار از یک دادگان استاندارد ۱۰۰ ثانیه‌ایی از فایلهای صوتی به زبان آلمانی استخراج و جداول هافمن مورد نیاز براساس این دادگان ایجاد شد . اما مشکلی که در رابطه با این دادگان وجود داشت این بود که جملات بیان شده در آن تکراری بودند. اما چون این جملات به لحنهای مختلف بیان شده‌بودند تاحدودی مشکل بیان‌شده برطرف میگردید. اما به هرحال با توجه به محدودیتی که در تهیه دادگان استاندارد وجود داشت، به همین دادگان بسنده شد و همانگونه که در ادامه می آید سعی شد که با اصلاح تابع توزیع آماری بدست آمده از آن، این مشکل تا حدود زیادی حل شود.

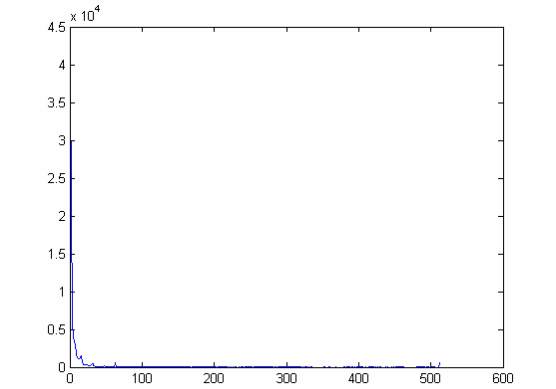
در ایجاد جداول هافمن مورد نیاز، همانند جدول هافمن JPEG، حداکثر Category برابر با ۱۱ در نظر گرفته شده است. از طرف دیگر در تشکیل این جدول هافمن به جای آنکه مشابه JPEG حداکثر Run برابر ۱۵ قرار داده شود در چند مرحله مختلف حداکثر Run، برابر با ۳۱، ۶۳، ۱۲۷، ۲۵۵ در نظر گرفته‌شد و جداول هافمن مختلفی بدست آمد. همچنین با توجه به اینکه سطح صدا در فایلهای صوتی مختلف مورد استفاده برای استخراج آمار مورد نیاز با هم برابر نبود و این دقت آمار استخراج شده را پایین می‌آورد، لذا، فایلهای مورد استفاده ابتدا نرمالیزه شده و سپس جدوال مورد نیاز از آنها استخراج شدند. نحوهٔ نرمالیزه کردن فایلها بدین صورت بود که در هر فایل با تقسیم ماکزیمم مقدار ممکن به ماکزیمم نمونهٔ فایل یک ضریب بدست می‌آمد و با ضرب کل نمونه های فایل در این ضریب، سطح صدای فایل نرمالیزه می‌شد. در شکل ۲ نمودار وقوع Run های مختلف در سیگنالهای صحبت دادگان استاندارد نرمالیزه شده نشان داده شده است.

همچنین به عنوان نمونه در شکل ۳، نمودار احتمال وقوع Run/Cat های مختلف برای سیگنال استاندارد نرمالیزه نشان داده شده است. محور افقی برابر است با x=(Run+1)*cat که در واقع شماره‌ی ردیف هر کد در جدول هافمنی که بر این اساس تشکیل می‌شود نیز می‌باشد. در اینجا Run از ۰ تا ۱۵ و Cat از ۱ تا ۱۱ تغییر می‌کند.

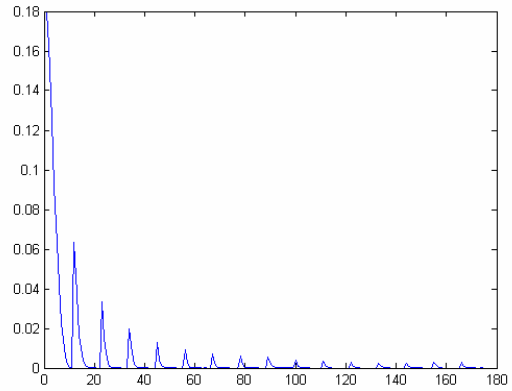
همانطور که از شکل ۳ پیداست برای برخی مقادیر x ، مقدار احتمال صفر است. این امر به خصوص در حالتی که مقدار Run بسیار بزرگ است منجر به ایجا کدهای هافمن با طول بعضاً ۲۵۰۰ بیتی می شود. لذا چنانچه در یک فایل این حالت رخ دهد، می‌تواند به مقدار زیادی باعث افزایش نرخ بیت شود. برای حل این مشکل باید دادگان مورد بررسی را تا حد زیادی گسترش می‌دادیم. اما با توجه به محدودیتی که در این زمینه وجود داشت، سعی‌شد که به صورت مصنوعی یا به عبارتی دستی، تابع توزیع احتمالات استخراج‌شده اصلاح شود.برای این منظور دو روش به کار گرفته شد.

۱- با استفاده از روش‌های برازش منحنی ، یک مقدار غیر صفر به جاهایی که احتمال برابر با صفر است اختصاص داده شد. ابتدا برازش منحنی بر روی تابع حاصل از مقادیر شمارش شده هر نمونه صورت گرفت، سپس با تقسیم این مقادیر بر مجموع کل مقادیر شمارش شده، تابع توزیع احتمالات تصحیح شده بدست آمد. برازش منحنی صورت‌گرفته در دو مرحله انجام شد و در مرحله اول برازش منحنی بر روی منحنی مقادیر ماکزیمم نسبی نمودار وقوع Run/Cat انجام شد. در مرحله‌ی بعد، برازش

منحنی بر روی نمودار مقادیر شمارش شده‌ی Run/Cat که در آن Run=0 و Cat از ۱ تا ۱۱ تغییر می‌کند، با استفاده از تابع گاوسی a1*exp(-((x-b1)/c1)^2)، انجام گرفت، سپس با مقیاس دهی این نمودار برای سایر مقادیر Run، تابع توزیع احتمالات تصحیح‌شده با این روش بدست‌آمد.



شکل (۲): نمودار احتمال قوع Run های مختلف در سیگنالهای صحبت آلمانی استاندارد نرمالیزه شده



شکل (۳): نمودار وقوع Run/Cat در فایلهای صحبت نرمالیزه شده

۲- در این روش ابتدا کل مقادیر شمارش شده در عدد ۲ ضرب شده و سپس با مقدار ۱ جمع شدند. علت این کار این بود که به نحوی مقادیری را که مقدار شمارش شده‌ی آنها صفر است، به مقدار غیر صفر ۱ تغییر دهیم. از طرفی برای آنکه تاثیر این جمع زدن را در تابع توزیع احتمالات کم‌کنیم، ابتدا مقادیر را در عدد ۲ ضرب کردیم. در صورتی که این مقادیر در عددی بزرگتر از ۲ ضرب می‌شدند، احتمالات نزدیک صفر کوچکتر شده و منجر به تولید ک دهای بزرگتری میشد، لذا به همان مقدار ضریب ۲ بسنده شد. به عبارتی اگر منحنی اولیه را C(x) بنامیم، منحنی ثانویه برابر است با C(x)*2+1 . سپس هر یک از مقادیر جدید بر مجموع کل مقادیر جدید تقسیم شده و تابع توزیع احتمالات جدید تصحیح‌شده به دست‌آمد.

در حالی که در عمل روش ۱ حتی عملکرد بدتری نسبت به جداول حاصل از تابع توزیع اصلاح نشده دارد، روش دوم عملکرد بسیار بهتری دارد. با استفاده از این روش کدهایی که در جدول سابق دارای طولی درحدود ۲۵۰۰ بیت بودند به کدهایی با طول ۱۸-۱۹ بیت تبدیل شدند در حالی که طول کدهای کوچکتر زیاد نشد و این نتیجه بسیار مطلوب بود. با توجه به توضیحات داده شده، برای هر آمار استخراج شده ، ۴ نوع تابع توزیع احتمالات بدست آمد (آمار استخراج شده از فایلهای استاندارد اصلی و نرمالیزه شده‌ی آنها به صورت خام و اصلاح‌شده) که جداول هافمن برای هریک از این توابع در ۵ حالت مختلف برای حداکثر Run برابر با ۱۵، ۳۱، ۶۳، ۱۲۷و ۲۵۵ ایجاد شد. در واقع ۲۰ جدول هافمن مختلف ایجاد شد. البته در اینجا برای حالتی که تعداد صفرها بیشتر از حداکثر Run باشد از یک کد ۱۷ بیتی برای نشان دادن تعداد صفرها استفاده می کنیم. برای تشخیص این نوع کلمه‌کد خاص از دیگر کدها قبل از آن کد ۸ بیتی '10101001' را که برای نشان دادن انتهای کدگذاری قطعه سیگنال نیز استفاده می شود قرار می دهیم. علت تغییر این کدها نسبت به کدهای مورد استفاده در جدول JPEG این است که این کدها باید نسبت به بقیه کدهای جدول دارای خاصیت پیشوندی باشند. بنابراین با لحاظ کردن احتمال حالاتی که نیاز به استفاده از این کدها می باشد – در تابع توزیع احتمالات بدست آمده – این کدها با استفاده از الگوریتم هافمن بدست آمدند.

ب- کدگذاری حسابی[10,11] با استفاده از آمار استخراج شده از فایلهای صوتی

با توجه با اینکه در این پروژه طول هر قطعه برای کدگذاری حسابی برابر با ۵۱۲ نمونه در نظر گرفته شده است، تابع توزیع احتمالاتی مورد استفاده براساس حداکثر Run برابر با ۵۱۱ استخراج شد. به عبارتی تعداد نمونه های تابع توزیع مورد استفاده برابر با ۵۶۳۲ = ۱۱ × ۵۱۲ است. تابع توزیع مورد استفاده، تابع اصلاح‌شده‌ای است که از فایلهای صوتی آلمانی نرمالیزه‌شده بدست‌آمد.

در اینکدر طراحی شده، با خواندن هر نمونه‌ی غیر صفر، شماره ردیف Run/Length موردنظر در جدول تابع توزیع احتمالات با فرمول x=(Run+1)*cat محاسبه می‌شود. سپس در یک ماتریس افقی این مقادیر پشت سر هم قرار داده می‌شود و در واقع عمل کدگذاری حسابی بر روی این ماتریس انجام می‌شود. مشخص است که اگر آخرین نمونه‌ی قطعه غیرصفر نباشد، صفرهای قبل آن کدگذاری نمی‌شوند، لذا برای حل این مشکل در ابتدای کدگذاری هر قطعه در یک کد ۱۶ بیتی هم طول قطعه و هم تعداد صفرهای انتهایی آن فرستاده می‌شود. پس از این مرحله، خود مقادیر اصلی غیرصفر بردار چندی‌شده که در یک ماتریس افقی دیگر پشت‌سرهم قرار داده شده‌اند، کد می‌شوند.

همچنین در صورتی که کل قطعه برابر صفر باشد، در همان کد ۱۶ بیتی مقدار ۵۱۲ ارسال شده و کدگذاری دیگری صورت نمی‌گیرد.

۳-۳ کدگذاری وفقی

در این پروژه سعی شد تا با پیاده سازی الگوریتم‌های هافمن و حسابی به صورت وفقی، میزان تاثیر این روش‌ها بر نرخ بیت بررسی شود. در زیر نحوه‌ی پیاده سازی هریک از این الگوریتم‌ها به صورت وفقی توضیح داده می‌شود.

الف- کدگذاری هافمن وفقی

برای پیاده سازی الگوریتم هافمن به صورت وفقی از الگوریتم معروف FGK [12] استفاده شد. بر خلاف کدگذاری هافمن به صورت ایستا که از قبل تمامی کدها تولید شده و در یک جدول LookUp ذخیره شده است، در این روش پس از کدکردن هر نمونه، ساختار درخت به روز شده و ممکن است تغییر کند. لذا کدها در حین کدگذاری کل فایل ثابت نبوده و برای کدکردن هر نمونه باید درخت دوباره پیمایش شود تا کد مربوطه استخراج شود.

با توجه به اینکه به‌روزکردن درخت هافمن پس از کد کردن هر نمونه زمان زیادی می‌برد، لذا سعی شد برای بالا بردن سرعت اجرای الگوریتم، تا حد ممکن تعداد برگهای درخت کم باشد. برای این منظور ابتدا جدول احتمالات Run/Cat مربوط به حداکثر Run برابر با ۲۵۵ به صورت نزولی مرتب شد و از آنجایی که مشاهد شد مجموع احتمالات ۶۳ نمونه‌ی (کد Run/Cat) با احتمال بالاتر، برابر با ۰/۹۷۴ می باشد (به عبارت دیگر در یک فایل، ۹۷/۴ درصد نمونه ها از این ۶۳ نمونه خواهند بود)، در پیاده سازی این الگوریتم، درخت هافمن فقط برای این ۶۳ نمونه تشکیل شد و یک برگ اضافه به نام LP (Low Probability : اشاره به ۲۷۵۵ نمونه‌ی با احتمال پایین) نیز ,وجود دارد تا هر بار که یکی از آن ۲۷۵۵ نمونه‌ی با احتمال کمتر رخ داد، ابتدا کد مربوط به برگ LP و به دنبال آن شماره ردیف کد مربوطه در یک کد ۱۲ بیتی ارسال شود. در دیکدر نیز به همین ترتیب عمل دیکدینگ انجام می‌شود.

ب- کدگذاری حسابی وفقی

در کدگذاری حسابی برای هر نمونه یک کد به طور جداگانه وجود ندارد بلکه کل نمونه های یک قطعه با هم کد می‌شوند. بنابراین در پیاده سازی الگوریتم حسابی وفقی، پس از کد کردن هر ۵۱۲ نمونه (یک قطعه) تابع توزیع احتمالات را به‌روزرسانی می‌کنیم. برای به‌روزکردن تابع توزیع احتمالات از یک ضریب ff به نام Forgetting Factor طبق فرمول زیر استفاده می‌شود.

$$P_{new} = (1-ff)\times P_{old} + ff\times P_{SEG} \qquad (۱)$$

P_{SEG}=احتمال نمونه در قطعه‌ای که کد شده است

در این رابطه با افزایش مقدار ff وابستگی احتمال جدید به احتمال قدیمی کمتر شده و اثر احتمال نمونه در قطعه کدشده، در احتمال جدید بیشتر می‌شود. همین روش به همین ترتیب در دیکدر نیز استفاده شده و تابع توزیع پس از دیکدکردن هر قطعه به‌روزرسانی می‌شود.

۳-۴ کدگذاری ثانویه با استفاده از کدگذاری حسابی

برای دستیابی به نرخ بیت پایین تر، سعی شد که کد نهایی تولیدشده که فقط شامل بیتهای ۰ و ۱ است، مجدداً با استفاده از الگوریتم کدگذاری حسابی کدگذاری شود. اما با توجه به اینکه احتمال بیتهای ۰ و ۱ تولید شده از مرحله قبل، به تقریب بسیار بالایی با هم برابر بودند لذا عملاً هیچ بهبودی حاصل نشد. هر چند باز هم سعی شد که با تقسیم بندی کل فایل خروجی به بلوکهای با طول ثابت یا متغیر این توازن در تعداد صفر و یک ها شکسته شود، اما باز هم عملاً هیچ بهبودی در میزان فشرده سازی فایل حاصل نشد.

۴- نتایج

در ادامه نتایج کدگذاری 5 فایل مختلف به زبان آلمانی (gTest1…gTest5) آمده است

الف- در جداول (۲) و (۳) میزان بهبود نرخ بیت با استفاده از جداول هافمن جدید نسبت به نرخ بیت حاصل از کدگذاری با جدول JPEG، بر حسب درصد آمده‌است.

با توجه جدول ۲، کدگذاری با جداول هافمنی که حداکثر Run در آنها از ۶۳ بیشتر باشد، به جای بهبود در میزان فشرده سازی، وضعیت کدگذاری بدتر شده است که علت آن وجود همان کدهای بسیار بلند است. در حالی که با توجه به جدول ۳ با استفاده از جداول هافمن که از تابع توزیع اصلاح شده بدست امده‌اند با افزایش حداکثر Run بهبود بیشتری در میزان فشرده‌سازی حاصل می‌شود.

ب- در جداول (۴) و (۵) نتایج کدگذاری VLC فایلهای آلمانی با استفاده از کدگذاری حسابی و در کنار نتایج حاصل از کدگذاری هافمن (با استفاده از جدول بدست آمده از تابع توزیع اصلاح شده‌ی بدست آمده از فایلهای صوتی آلمانی نرمالیزه‌شده با حداکثر Run برابر با ۲۵۵) و کدگذاری با جداول JPEG آمده‌است تا بتوان آنها را از لحاظ میزان بهبود نسبت به جدول JPEG و زمان کدینگ و دیکدینگ مقایسه کرد.

ج- نتایج کدگذاری وفقی: در جدول ۶ نتایج کدگذاری حسابی وفقی با مقدار ضریب ff =۰/۱ آمده است. در آزمایش‌های انجام شده مشاهده شد تغییر این ضریب اثر چندانی در فشرده سازی ندارد و حتی افزایش آن تأثیر منفی دارد

جدول (۲): بررسی کدگذاری با جداول مختلف هافمن بدست آمده از دادگان استاندارد آلمانی

	Improvement %					
File Name / Max Run	15	31	63	127	255	JPEG Bit Rate
gTest1 (4sec)	3.3973	5.6668	6.7909	6.095	1.5542	27295
gTest2 (4sec)	2.9632	5.5045	6.4555	6.1289	3.9228	23371
gTest3 (4sec)	3.5731	5.374	6.251	6.0675	3.1244	33132
gTest4 (4sec)	3.1714	5.3904	5.7881	2.7202	-7.1583	28513
gTest5 (4sec)	3.4297	5.3634	6.4189	5.7815	2.6911	27957

جدول(۳): بررسی کدگذاری با جداول مختلف هافمن بدست آمده از دادگان استاندارد آلمانی نرمالیزه شده با تابع توزیع اصلاح شده

	Improvement %					
File Name / Max Run	15	31	63	127	255	JPEG Bit Rate
gTest1 (4sec)	3.5798	5.7355	7.1678	7.5025	7.5777	27295
gTest2 (4sec)	3.1281	5.3482	6.6129	6.8924	6.9781	23371
gTest3 (4sec)	3.4394	5.3211	6.3734	6.6325	6.668	33132
gTest4 (4sec)	3.404	5.516	6.8686	7.1706	7.2039	28513
gTest5 (4sec)	3.2963	5.2802	6.8379	7.2318	7.2757	27957

جدول (۴): درصد بهبود میزان فشرده سازی با کدگذاری هافمن و حسابی نسبت به bit rate حاصل از کدگذاری با جدول JPEG

	Improvement %		
File Name	HUFFMAN	ARITHMETIC	JPEG bit rate
gTest1 (4sec)	6.9191	8.5131	27295
gTest2 (4sec)	6.1908	8.2751	23371
gTest3 (4sec)	6.1648	7.6663	33132
gTest4 (4sec)	6.5774	7.9654	28513
gTest5 (4sec)	6.6032	8.4227	27957

جدول (۵): مدت زمان کدینگ و دیکدینگ فایلهای مورد آزمایش در کدگذاری هافمن و حسابی و جدول JPEG

	Coding time (s)			Decoding time (s)		
File Name	JPEG	HUFFMAN	ARITHMETIC	JPEG	HUFFMAN	ARITHMETIC
gTest1 (4sec)	8.672	8.092	8.652	15.583	13.489	46.838
gTest2 (4sec)	8.542	8.002	8.322	14.631	12.878	41.099
gTest3 (4sec)	9.063	8.271	8.743	17.135	14.682	54.728
gTest4 (4sec)	9.063	8.161	8.512	16.244	13.95	48.91
gTest5 (4sec)	8.772	8.442	8.442	15.633	14.36	47.038

جدول(۶): درصد بهبود میزان فشرده سازی وزمان کدگذاری با الگوریتمهای هافمن و حسابی وفقی نسبت به bit rate حاصل از کدگذاری با جدول JPEG

	Improvement %		Coding time (s)	
File Name	Huffman	Arithmetic	Huffman	Arithmetic
gTest1 (4sec)	5.2263	6.5131	89.669	8.423
gTest2 (4sec)	4.5859	6.5471	81.216	8.362
gTest3 (4sec)	4.8482	6.1187	223.91	8.873
gTest4 (4sec)	4.9122	5.6983	202.38	8.613
gTest5 (4sec)	5.0209	6.4761	194.65	8.593

۵- نتیجه گیری

در این مقاله به بررسی کدگذاری VLC فایلهای صوتی پرداخته و سعی شده است به جای استفاده از جدول JPEG، از کدگذاری حسابی یا کدگذاری هافمن با ایجاد یک جدول هافمن ویژه‌ی صوت همانند آنچه که برای JPEG انجام شده است، استفاده شود. برای این منظور بررسی و مطالعه بر روی انواع فایلهای صوتی مختلف و به زبان های مختلف (فارسی، انگلیسی، آلمانی، موسیقی) انجام شد. در طی این بررسی سعی شد که با اعمال یک سری اصلاحات در فایلهای دادگان و همچنین تابع توزیع بدست‌آمده، جداول با عملکرد بهتری بدست‌آیند. با توجه به نتایج بدست‌آمده، مشاهده می‌شود که جداول حاصل از دادگان نرمالیزه‌شده با تابع توزیع اصلاح‌شده بهترین عملکرد را دارند.

با افزایش حداکثر مقدارRun در جداول بدست آمده از دادگان با تابع توزیع اصلاح‌شده میزان فشرده سازی بهبود می‌یابد، اما با توجه به افزایش شدید حجم جدول، میزان بهبود حاصل ناچیز است. همچنین در یک بررسی دیگر با مقایسه‌ی میزان و زمان کدگذاری و کدگشایی کدگذاری حسابی با بهترین حالت کدگذاری هافمن، مشخص شد که با وجود عملکرد بهتر کدگذاری حسابی از لحاظ نرخ بیت خروجی، سرعت کدگذاری هافمن به ویژه هنگام کدگشایی بسیار بیشتر از کدگذاری حسابی است. همچنین با بررسی هر دو الگوریتم حسابی و هامفن در حالت وفقی مشخص شد که هم از لحاظ زمانی و هم نرخ بیت، مدل ایستا بر مدل وفقی برتری دارد و تنها مزیت مدل وفقی عدم نیاز به حافظه‌ی بیشتر برای ذخیره جدول look up است. همچنین در کدگذاری حسابی وفقی مشاهده شد با افزایش ضریب ff میزان بهبود در فشرده‌سازی فایلها کاهش می‌یابد.

۶- تقدیر و تشکر

این پژوهش با استفاده از اعتبارات پژوهشی دانشگاه شهید بهشتی انجام شده است.

۷- مراجع

[1] Taha Mortazavi , M.H. Savoji "*Adaptive Wavelet Coding of Audio and High Quality Speech at 32Kb/s Using Psycho-Acoustic Noise Masking Effects* ", ICA2004.

[۲] سید طه مرتضوی، محمد حسن ساوجی ″ فشرده سازی سیگنال صحبت باند وسیع و صوت با استفاده از تبدیل موجک″ نشریه مهندسی برق و کامپیوترایران، پاییز و زمستان.

[3] Daubechies, I.; "*Ten lectures on wavelets*", SIAM, (1992).

[4] Stephane Mallat; " *A Wavelet Tour of Signal Processing*"; Academic Press, 1999.

[5] C.Sidney Burrus, Ramesh A. Gopinath, Haitao Guo; "*Introduction to Wavelets and Wavelet Transforms*"; Prentice Hall, 1993.

[6] P. Srinivasan, H. Jamieson; "*High Quality Audio Compression Using an Adaptive Wavelet Packet Decomposition and Psychoacoustic Modeling*"; IEEE Tran. Signal processing, VOL. 46, No. 4, April 1998.

[7] E. Ambikairajah. A.G. Davis & W.T.K. Wong; "*Auditory Masking & MPEG-1 Audio Compression"*; Electronics & Communication Engineering Journal, Aug. 1997.

[8] D. Sinha, A.H. Tewfik; "*Low Bit Rate Transparent Audio Compression using Adaptive Wavelets*"; IEEE Trans. Signal Processing, Vol. 41, No. 12, Dec. 1993.

[9] D.A. Huffman, "*A method for construction of minimum redundancy codes*," Proc. IRE, vol. 40,pp. 1098-1101, 1952.

[10] J.J. Rissanen, "*Generalized Kraft inequality and arithmetic coding*," IBM J. Res. Develop ,vol. 20, no. 3, pp. 198-203, May 1976.

[11]] I.H. Witten, R.M. Neal, and J.G. Cleary, "*Arithmetic coding for data compression,*" Commun. ACM, vol. 30, no. 6, pp. 520-540, June 1987.

[12] D.E. Knuth, "*Dynamic Huffman coding*," J. of Algorithms, Vol. 6, pp. 163-180, 1985