دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

طراحی کدکننده مطلع از محدودیت‌های گیرنده با استاندارد H.264/AVC در کاربردهای تطبیق ویدئو

نگارش:

مهدی سمسارزاده

استاد راهنما:

دکتر محمودرضا هاشمی

استاد مشاور:

دکتر شروین شیرمحمدی

رساله برای دریافت درجه دکتری در رشته

مهندسی کامپیوتر – گرایش معماری کامپیوتر

شهریور ۱۳۹۲

چکیده

با پیشرفت تکنولوژی استفاده از کاربردهای چند رسانه‌ای در ابزارهای قابل حمل گسترش یافته است. این در حالی است که این ابزارها دارای محدودیتهای منابع مختلفی می‌باشند. هر فرد با در دست داشتن یک ابزار چندرسانه‌ای قابل حمل، نظیر تلفن هوشمند یا تبلت، امکان تولید و مصرف محتوای چند رسانه‌ای را خواهد داشت. بدلیل تنوع روز افزون ابزارهای مصرف محتوا، تولیدکنندگان محتوای چند رسانه‌ای نیاز دارند یک محتوای مثلاً ویدئویی را برای گیرنده‌های متعددی با مشخصات مختلف ارسال کنند. یکی از راه حل‌های رایج و مقرون به صرفه برای این منظور آن است که تنها یک رشته بیت تولید شده، با توجه به مشخصات و محدودیتهای هر گیرنده، در تعدادی نقطه میانی تطبیق داده شود. به علاوه، همانطور که ذکر شد با گسترش قابلیت‌های ابزارهای دسترسی، امروزه هر مصرف‌کننده (کدگشا) بالقوه یک تولید کننده بالقوه محتوای ویدئویی (کدکننده) نیز هست. در نتیجه مناسب است که محدودیتهای دستگاههای چندرسانه‌ای نظیر توان مصرفی، قدرت پردازنده و میزان حافظه در حین کدکردن محتوای چندرسانه‌ای مد نظر قرار گیرند.

در این رساله با معرفی این مسئله کلی، لزوم مطلع بودن کدکننده از محدودیتهای منابع تبیین می‌شود. سپس مسئله کلی طراحی کدکننده مطلع از محدودیتهای منابع، بصورت یک روش انتزاعی ارائه شده و حل می‌گردد. در ادامه روش انتزاعی پیشنهادی برای سه زیر مسئله در این رابطه که عبارتند از کدکننده مطلع از محدودیتهای گیرنده، کدکننده مطلع از محدودیتهای گیرنده برای کاربردهای تطبیق و کدکننده مطلع از محدودیت منابع خویش سفارشی سازی خواهد شد. در کدکننده مطلع از محدودیتهای گیرنده، محدودیتهای دستگاه گیرنده در حین کدکردن ویدئو در سمت سرویس دهنده در نظر گرفته می‌شود. به عبارت دیگر، رشته بیتی تولید می‌شود که پس از کدگشایی میزان منابع مشخصی مطابق با محدودیتهای گیرنده را مصرف نماید. کدکننده مطلع از محدودیتهای گیرنده برای کاربردهای تطبیق، کدکننده‌ای است که علاوه بر در نظر گرفتن مشخصات و محدودیتهای گیرنده، مشخصات و رفتار عملیات تطبیق – که در یک گره میانی صورت می‌پذیرد – را نیز مد نظر قرار می‌دهد. بنابراین، این کدکننده رشته بیتی را تولید می‌کند که محدودیتهای گیرنده‌های مختلف را – حتی پس از انجام عملیات تطبیق – برآورده می‌کند. نهایتا مراد از کدکننده مطلع از محدودیت منابع خویش، کدکننده‌ای است که بر روی یک دستگاه با محدودیت منابع – مانند یک کدکننده ویدئو بر روی دستگاه تلفن هوشمند – واقع شده است. این کدکننده نیز می‌تواند محدودیتهای منابع خود را در حین تولید رشته بیت مد نظر قرار دهد.

از جمله مزایای روش انتزاعی پیشنهادی می‌توان به همه منظوره بودن آن اشاره کرد، چرا که برای ترکیبات مختلفی از منابع محدود، قابل استفاده است. به علاوه، این روش با استفاده از یک روش شناسی مناسب اقدام به استخراج پارامترهای کدکردن می‌نماید. برای نشان دادن کارائی روش انتزاعی پیشنهادی، برای هر یک از سه مسئله مطرح شده در این رساله، یک نمونه مطالعاتی – با در نظر گرفتن مصرف توان بعنوان محدودیت – را در نرم افزار مرجع کدکننده و کدگشای H.264/AVC پیاده سازی و ارزیابی کرده‌ایم.

نهایتا در بخش دوم این رساله، به ارائه یک شیوه شناسی تخمین مصرف منابع پرداخته‌ایم. این شیوه شناسی برای پیاده سازی و ارزیابی نمونه‌های مطالعاتی مربوط به پیشنهاد اول این رساله بکار رفته است. برای ارزیابی کارائی شیوه شناسی پیشنهادی، میزان پیچدگی در سمت کدکننده و کدگشا در استاندارد H.264/AVC تخمین زده شده‌اند.

نتایج بدست آمده برای تمامی نمونه‌های مطالعاتی ارائه شده در این رساله، نشان از کارائی روش انتزاعی پیشنهادی و روش پیشنهادی برای تخمین مصرف منابع دارد و قابلیت به کارگیری آن‌ها را در ابزارهای قابل حمل نشان می‌دهد. لازم به ذکر است که هم شیوه انتزاعی پیشنهادی برای طراحی کدکننده مطلع از محدودیت منابع و هم شیوه شناسی تخمین مصرف منابع، که هر دو بصورت عام مطرح شده‌اند، قابلیت استفاده در محدوده وسیعی از کاربردها را دارند. به عبارت دیگر، روش انتزاعی پیشنهادی در هر ساختاری مربوط به ارسال و دریافت ویدئو که دارای گیرنده/فرستنده با منابع محدود باشد - به منظور طراحی کدکننده مطلع از محدودیت منابع در آن ساختار- قابل استفاده است. تنها لازم است که با توجه به شرایط مسئله این روش انتزاعی سفارشی سازی گردد. این خصوصیت برای شیوه شناسی تخمین مصرف منابع نیز صدق می‌کند. برای هر کاربردی که به روند اجرا و الگوریتم آن دسترسی داشته، تعریف یا مدل مناسبی از معیار مصرف منبع مربوطه‌اش داشته باشیم، می‌توان با دنبال کردن چنین روندی به مدل تخمین مناسبی دست یافت.

فهرست مطالب

فهرست اشکال

فهرست جداول

فهرست لیست‌ها

فصل اول

مقدمه

امروزه فراگیر شدن محتوای چندرسانه‌ای با توجه به پیشرفت‌های اتفاق افتاده در زمینه‌های فشرده‌سازی، انتقال، دریافت و کدگشایی این گونه محتوا، رنگ واقعیت به خود گرفته است. با استفاده از امکانات و قابلیت‌های استانداردهای فشرده‌سازی [2][1]، انواع متعددی از ابزارها از جمله دستگاه‌های همراه مانند کامپیوتر قابل حمل، تبلت[1] و تلفن وشمند[2] قادر هستند که محتواهای چندرسانه‌ای را تولید و پخش کنند، در حالی که استانداردهای نوظهور شبکه مانند 3G و 4G می‌توانند مسائل مربوط به انتقال آنها را حل و فصل کنند. این پیشرفتها در زمینه‌های فشرده سازی و انتقال نه تنها باعث شده است که تمام منابع مبتنی بر وب در هر زمان و هر مکان به همدیگر متصل باقی بمانند، بلکه محدودیت‌های ارسال و دریافت نیز از میان رفته‌اند. در چنین وضعیتی هر کسی می‌تواند رویدادهای محلی در سراسر دنیا را بصورت آنی با استفاده از اطلاعات صدا، تصویر و ویدئوی مربوطه در دسترس عموم قرار دهد. در سمت دیگر، با استفاده از دستگاه قابل حملی که توسط شبکه پشتیبانی می‌شود، هرکسی در هر نقطه از جهان می‌تواند بصورت لحظه‌ای در جریان اتفاقات جاری سراسر جهان قرار گیرد و به آنها دستیابی داشته باشد.

اگرچه هر محتوای چندرسانه‌ای مشخصات مربوط به خود را داراست، اما از آنجا که فشرده‌سازی و کدگشایی ویدئو یکی از پیچیده ترین عملیات موجود در تولید محتوای چندرسانه‌ای است، تمرکز این رساله بر روی ویدئو می‌باشد.

---

[1] Tablet
[2] Smart Phone

در بسیاری از کاربردها یک محتوای ویدئویی بصورت همزمان برای چند گیرنده با قابلیتهای مختلف ارسال می‌گردد. بعنوان مثال یک مسابقه زنده فوتبال می‌تواند بصورت همزمان، توسط دستگاه تلویزیون هوشمند[1]، دستگاه کامپیوتر قابل حمل، دستگاه تلفن هوشمند و تبلت دریافت شود. این در حالی است که هر یک از این ابزارها دارای مشخصات و محدودیتهای خاص خود می‌باشند. بعنوان نمونه، نرخ بیتی که یک دستگاه تلویزیون هوشمند می‌تواند دریافت کند با نرخ بیت دریافتی یک تبلت متفاوت است، این مشخصه به مشخصه کیفیتی[2] نیز شناخته می‌شود. به علاوه هر گیرنده قابلیت نمایش ویدئو با وضوح معینی را دارد که به مشخصه مکانی[3] نیز معروف است. همچنین تعداد فریمی از ویدئو که در هر گیرنده قابل نمایش است نیز محدودیت دیگری می‌باشد که به مشخصه زمانی[4] نیز شناخته می‌شود. به عنوان مثالی دیگر، قدرت پردازنده یک سیستم سرویس دهنده، آن را در انجام عملیات تولید محتوای ویدئویی محدود نمی‌سازد، در حالی که برای یک دستگاه ساده قابل حمل، در نظر گرفتن قدرت پردازنده برای تولید یا مصرف محتوای ویدئویی ضروری می‌باشد. به همین ترتیب میزان توان مصرفی یک دستگاه ثابت که به منبع تولید نیرو متصل است اهمیت کمتری دارد، این در حالی است که برای ابزارهای همراه، میزان توان مصرفی به ازای محتوای مختلف اهمیت به سزایی دارد. لازم به ذکر است که در این رساله در برخی موارد، معیار پیچیدگی بجای توان مصرفی بکار خواهد رفت. منظور از معیار پیچیدگی، تعداد پالسهای ساعت مورد نیاز برای اجرای یک پروسه در پردازنده می‌باشد. بعدا در بخش ۴-۲-۱ نشان خواهیم داد که معیار پیچیدگی ارتباط مستقیمی با توان مصرفی دارد. مشخصه‌های مذکور در پاراگراف قبل برای کدکننده‌ای که بر روی دستگاهی که دارای محدودیت منابع می‌باشد - مانند ابزار قابل حمل - نیز اهمیت دارند. به عنوان نمونه توان مصرفی برای تمامی ابزارهای قابل حمل از اهمیت ویژه‌ای برخوردار است، چنانچه کدکننده هر دستگاه در حین عملیات کدکردن میزان مصرف توان خود را نیز مدیریت کند، می‌توان زمان بیشتری از دستگاه استفاده نمود.

از آنجا که هر کاربرد مشخصات خاص خود را داراست، مناسب است که برای هر گیرنده رشته بیتی منطبق با مشخصه‌هایش ارسال کرد. جهت ارسال محتوای چندرسانه‌ای به گیرنده‌های با مشخصات متنوع روش‌های مختلفی وجود دارد که در ادامه آنها را مرور خواهیم کرد.

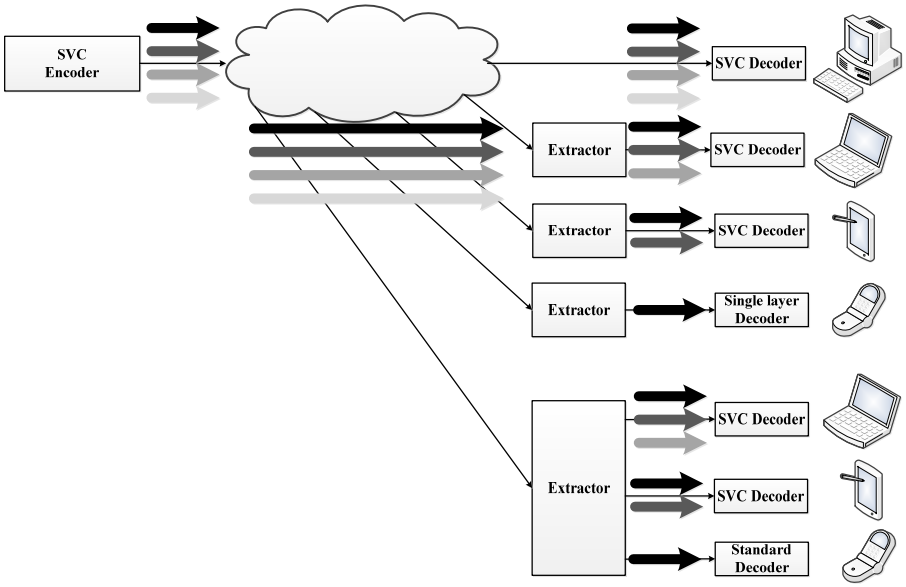
۱-۱ روشهای ارسال یک محتوای ویدئویی برای چند گیرنده

در ساده‌ترین روش که به چندپخشی معروف است، به ازای هر کاربر یک رشته بیت مستقل بر اساس مشخصات دستگاه گیرنده آن کاربر ارسال خواهد شد. به این منظور در حین تولید محتوای ویدئویی، بصورت همزمان به تعداد گیرنده‌های مختلف عملیات کدکردن تکرار می‌شود که این عملیات نیاز به مصرف منابع بسیاری دارد. مهمتر از آن میزان حجم داده‌ای که بر روی شبکه ارسال می‌شود و یا می‌بایست در مبداء ذخیره شود به شدت افزایش می‌یابد. این روش خصوصاً برای

[1] Smart TV
[2] Quality - PSNR
[3] Spatial
[4] Temporal

کاربردهای چندرسانه‌ای که به ازای هر رشته بیت حجم زیادی از اطلاعات جابجا می‌شود عملی نبوده و بهتر است از روش‌های دیگری استفاده کرد.

راه حل دیگری که پیشنهاد شده است کدکردن بصورت مقیاس پذیر[1] می‌باشد. در این روش چند لایه مختلف از ویدئو تولید می‌شود که با ترکیب لایه‌های بالاتر با لایه پایه می‌توان به ویدئو با مشخصات متفاوت رسید. بنابراین در این ویدئو لایه پایه دارای ویدئو با حداقل مشخصات مکانی، زمانی و کیفیتی خواهد بود و با افزودن لایه‌های بالاتر به ویدئو لایه پایه، به ویدئویی با مشخصات مکانی، زمانی و یا کیفیتی بالاتر خواهیم رسید[3]. ساختار کلی ارسال ویدئو بصورت مقیاس پذیر در شکل ۱-۱ نشان داده شده است. همانطور که در شکل آمده است، پس از تولید ویدئوی مقیاس پذیر توسط سرویس دهنده ویدئو، رشته بیت تولید شده بر روی شبکه ارسال شده و با استفاده از یک استخراج کننده[2] به هر گیرنده با توجه به قابلیت‌هایش (نظیر پهنای باند، وضوح نمایش و نرخ فریم)، زیر مجموعه‌ای از لایه‌های ویدئو ارسال خواهد شد. در گیرنده‌هایی که علاوه بر لایه پایه، لایه‌های دیگری را دریافت می‌کنند نیز یک کدگشای مقیاس پذیر کدگشایی لایه‌های ویدئو را انجام خواهد داد.

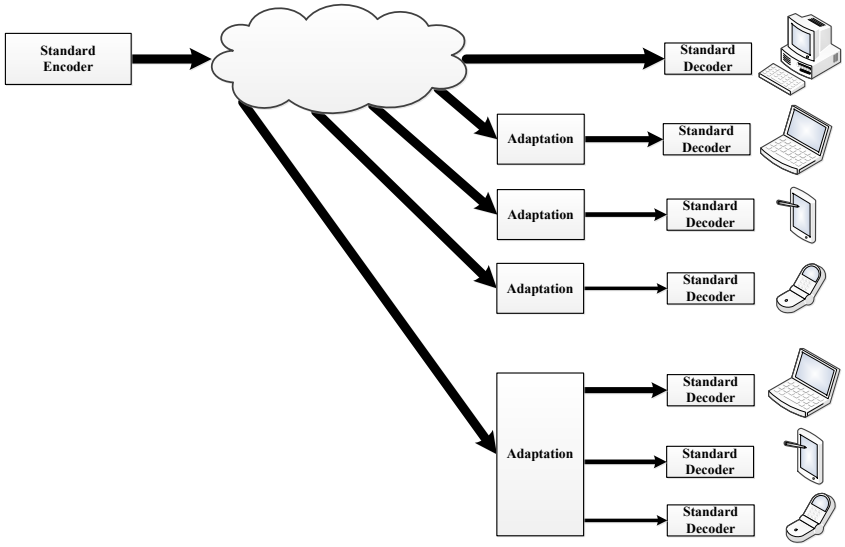


شکل ۱-۱ نحوه ارسال ویدئو بصورت مقیاس پذیر

روش دیگر عبارت است از تولید و ارسال رشته بیت برای گیرنده دارای بیشترین نرخ بیت، نرخ فریم و وضوح نمایش و تغییر آن در یک یا چند گره میانی بر اساس مشخصات و محدودیت‌های گیرنده‌های دیگر. به عملیات تغییر رشته بیت فشرده شده – که در یک گره میانی برای برآوردن محدودیت گیرنده انجام می‌شود – در اصطلاح عملیات تطبیق[3] گفته می‌شود. روشن است که عملیات تطبیق بایستی به تعداد گیرنده‌های مختلف در گره‌های میانی صورت پذیرد. بنابراین برای عملی بودن این روش، یا نقطه تطبیق کننده بایستی قدرت پردازش بالایی داشته باشد و یا عملیات تطبیق پیچیدگی محاسباتی زیادی

[1] Scalable Video Coding (SVC)
[2] Extractor
[3] Adaptation

نداشته باشد و یا اینکه عملیات تطبیق در چندین نقطه صورت پذیرد. ساختار کلی تولید ویدئو و تطبیق آن در گره میانی در شکل ۱-۲ آمده است. در این روش ویدئو خام با استفاده از یک کدکننده استاندارد تک لایه[1] و بر اساس مشخصات گیرنده با بهترین مشخصات، فشرده شده و ارسال می‌شود. رشته بیت تولید شده در یک سری نقاط میانی بر اساس مشخصات گیرنده تطبیق داده شده و سپس برای گیرنده ارسال می‌گردد. همانطور که در شکل نشان داده شده است در این روش از کدکننده و کدگشای استاندارد استفاده شده است. لازم به ذکر است که عملیات نقطه تطبیق نیز بر اساس روش تطبیق متفاوت خواهد بود.



شکل ۱-۲ نحوه ارسال ویدئو بصورت تطبیق پذیر

عملیات تطبیق در بسیاری از کاربردها چه برخط[2] و چه برون خط[3] استفاده می‌شود. فرض کنید که یک مسابقه فوتبال بصورت برخط با نرخ بیت و وضوح نمایش HD در حال پخش باشد. ممکن است کاربری که از طریق اینترنت به شبکه وصل است بخواهد این مسابقه را با وضوح SD[4] مشاهده کند و کاربری که در حال حرکت است بخواهد از طریق تلفن هوشمند خود با نرخ بیت و نرخ فریم پایینتری به این محتوای ویدئویی دسترسی یابد.

در کاربردهای برون خط نیز تطبیق مورد استفاده قرار می‌گیرد. یک ویدئوی فشرده شده با وضوح بالا را می‌توان بصورتی تطبیق داد که به وضوح مورد نظر برسد بدون اینکه نیاز به کدگشایی و کدکردن مجدد داشته باشیم.

۱-۲ مقایسه روشهای تطبیق و کدکردن مقیاس پذیر

در روش مقیاس‌پذیر، لایه پایه حداقل مشخصات ویدئو را مشخص می‌کند. به عبارت دیگر نمی‌توان به نرخ بیت، وضوح نمایش، نرخ فریم و کیفیتی کمتر از آنچه در لایه پایه وجود دارد رسید. به این ترتیب چنانچه کاربری با مشخصاتی پایینتر از

[1] Single layer encoder
[2] Online
[3] Offline
[4] Standard Definition

لایه پایه داشته باشیم یا بایستی برای آن کاربر خاص عملیات تطبیق را انجام دهیم و یا اینکه لایه پایه را مجدداً و بر اساس گیرنده با کمترین مشخصات تولید کنیم. در این روش در صورت خراب شدن لایه پایه، لایه‌های بالاتر قابل کدگشایی بصورت مستقل نخواهند بود. به علاوه، کارائی کدکردن در روش مقیاس‌پذیر نسبت به روش متداول تک لایه نیز پایین‌تر است، چرا که در روش مقیاس‌پذیر لایه‌های بالاتر بر اساس اطلاعات لایه پایه تولید می‌شوند که این مسئله منجر به تولید یک سری اطلاعات تکراری در تولید لایه‌ها می‌شود. لذا، در نرخ بیت مساوی کارایی کدکردن روش مقیاس پذیر از روش تک لایه کمتر خواهد بود [5]. همچنین عملیات کدکردن و کدگشایی ویدئو بصورت مقیاس‌پذیر بسیار پیچیده‌تر از روش متداول تک لایه است [5]. از آنجا که در بسیاری از کاربردها میزان مصرف توان از اهمیت ویژه‌ای برخوردار است لذا استفاده از کدکننده و کدگشای مقیاس‌پذیر منجر به افزایش مصرف توان و کاهش رضایتمندی کاربر خواهد شد. علاوه بر این در بسیاری از کاربردها نظیر ابزارهای قابل حمل، گیرنده دارای قدرت پردازشی محدودی می‌باشد و نمی‌تواند بصورت برخط رشته بیت مقیاس‌پذیر را فشرده‌سازی یا کدگشایی کند. کدگشایی ویدئو مقیاس‌پذیر نیز توسط یک کدگشای متداول تک لایه عملی نخواهد بود. بنابراین برای کدگشایی رشته بیت مقیاس‌پذیر بایستی تمامی دستگاههای موجود به این کدگشا مجهز گردند که در بسیاری از کاربردها عملی نمی‌باشد و یا هزینه بسیاری تحمیل خواهد کرد.

روش تطبیق نیز دارای معایب مختلفی می‌باشد. یکی از کاستی‌های عملیات تطبیق، کاهش کیفیت ویدئو بدلیل انجام عملیات کدکردن و کدگشایی مجدد است. اگرچه، همانطور که اشاره شد، روش مقیاس پذیر نیز کارائی کدکردن را کاهش می‌دهد، اما طبق بررسی‌های ارائه شده در [6] در برخی نرخ بیتها، روش مقیاس پذیر از روش تطبیق عملکرد بهتری دارد. عیب دیگر روشهای تطبیق، پیچیدگی عملیات تطبیق در نقطه تطبیق دهنده می‌باشد. چرا که در این نقطه حداقل یک عملیات کدگشایی و کدکردن جزئی انجام می‌شود که مسلما از عملیات ساده استخراج لایه‌های مختلف در روش مقیاس پذیر پیچیده‌تر خواهد بود [6].

با وجود کاستی‌های روش تطبیق، با در نظر گرفتن پاره‌ای از ملاحظات، در برخی از کاربردها روش تطبیق بر روش مقیاس‌پذیر ارجحیت دارد. در کاربردهایی که به دلیل سادگی ساختار پردازنده، قدرت پردازشی پائین است ترجیح داده می‌شود که بجای استفاده از یک کدگشای مقیاس پذیر با پیچیدگی محاسباتی بالا، از یک کدگشای استاندارد تک لایه با پیچیدگی کمتر استفاده شود. به علاوه بسیاری از تولید کنندگان ادامه استفاده از کدکننده و کدگشای استاندارد را – با توجه به فراگیر بودن و پیچیدگی کمتر آنها – به استفاده از کدکننده و کدگشای مقیاس پذیر ترجیح می‌دهند [5]. همچنین برای کاربردهایی که عملیات تطبیق در نقطه‌ای خارج از دستگاه گیرنده صورت می‌پذیرد نیز استفاده از عملیات تطبیق اولویت دارد. چرا که پیچیدگی محاسباتی انجام عملیات تطبیق برای دستگاه گیرنده اهمیت نخواهد داشت، این در حالی است که نمی‌توان عملیات کدگشایی مقیاس پذیر را در خارج از گیرنده انجام داد. به این ترتیب است که در بسیاری از کاربردها بجای ارسال یک رشته بیت بصورت مقیاس پذیر، یک رشته بیت معمولی تولید شده و عملیات تطبیق بر روی آن صورت خواهد پذیرفت. در این رساله نیز برای تغییر رشته بیت بر اساس مشخصات و محدودیتهای گیرنده، روش تطبیق را مد نظر قرار خواهیم داد.

۱-۳ انواع روش‌های تطبیق

عملیات تطبیق عموماً برای مشخصاتی نظیر نرخ بیت، نرخ فریم و وضوح نمایش صورت می‌پذیرد [4]. اگرچه روشهای متنوعی برای انجام عملیات تطبیق ارائه شده است [11]-[5]، اما در ادامه این بخش مروری بر روش‌های معروف تطبیق خواهیم داشت.

۱-۳-۱ تطبیق با استفاده از کدگشایی و کدکردن متوالی[1]

در صورتیکه با محدودیت منابع و زمان روبرو نباشیم می‌توان برای تولید رشته بیت با مشخصات جدید، ویدئو فشرده شده را بطور کامل کدگشایی کرد. سپس دوباره عملیات کدکردن بر روی ویدئو بازسازی شده را با مشخصات جدید (نظیر نرخ بیت، نرخ فریم، وضوح و ...) انجام داد و رشته بیت فشرده جدید را به گیرنده مورد نظر ارسال کرد. همانطور که واضح است از آنجا که این عملیات عموماً در یک گره میانی انجام می‌شود لذا زمان انجام محاسبات و همچنین میزان پیچیدگی اهمیت خواهد یافت و در صورت بالا بودن محاسبات برای کاربردهای برخط مناسب نخواهد بود. این روش به روش Cascade معروف است و عموماً بصورت عملی پیاده‌سازی نمی‌شود. روش Cascade بعنوان حد بالای نمودار نرخ-اعوجاج برای روش‌های عملی و با پیچیدگی کمتر مورد توجه قرار می‌گیرد و تمامی روش‌های جدید تطبیق نمودار نرخ-اعوجاج خود را با این روش مقایسه می‌کنند. به این ترتیب این روش محک مناسبی برای مقایسه کارائی روش‌های مختلف تطبیق خواهد بود.

با توجه به پیچیدگی بالای روش Cascade، روش‌های دیگری ارائه شده است تا حجم محاسبات را کاهش دهند. در روش‌های موجود معمولاً هر چه حجم محاسبات کمتری انجام شود نمودار نرخ-اعوجاج با نمودار Cascade فاصله بیشتری پیدا خواهد کرد. در روش‌های جایگزین بیشتر سعی شده است که عملیات زمان بر در کدگشایی و کدکردن حذف شوند تا از پیچیدگی محاسباتی کاسته شود.

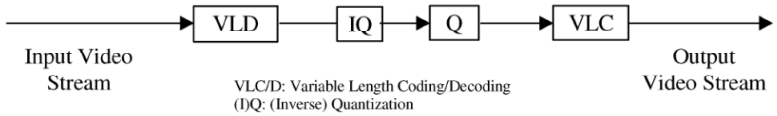
با توجه به کاربرد بیشتر تطبیق نرخ بیت، بیشتر روش‌های جایگزین Cascade در مورد نرخ بیت معرفی شده‌اند، هر چند که رویه مشابهی برای تطبیق به نرخ فریم و اندازه ویدئو صورت می‌گیرد. در ادامه به معرفی دو روش در زمینه تطبیق نرخ بیت خواهیم پرداخت.

۱-۳-۲ تطبیق حلقه باز[2]

در این روش پس از دریافت رشته بیت فشرده شده، عملیات کدگشایی بصورت جزئی انجام می‌شود و پس از اعمال تغییرات لازم بر روی رشته بیت (از جمله تغییر بردار حرکت یا چندی سازی مجدد داده‌های مانده [3]) مجدداً رشته بیت کدشده و ارسال می‌گردد. از آنجا که هر فریم، مرجعی برای کدگشایی فریم‌های آتی است و در این روش تغییر فریم جاری در نظر گرفته نمی‌شود، لذا کدگشا و تطبیق دهنده دارای فریم‌های مرجع متفاوتی خواهند بود و تا رسیدن به فریم I بعدی این تفاوت

[1] Cascade
[2] Open loop transcoding
[3] Re-Quantization

باقی خواهد ماند. این مسئله به رانش[1] مشهور است و کیفیت ویدئو را به میزان قابل توجهی تنزل خواهد داد. اگرچه این روش دارای پیچیدگی بسیار کمی در مقایسه با روش Cascade می‌باشد اما بخاطر مشکل رانش کمتر مورد استفاده قرار می‌گیرد. بلوک دیاگرام کلی این روش در شکل ۱-۳ آمده است.



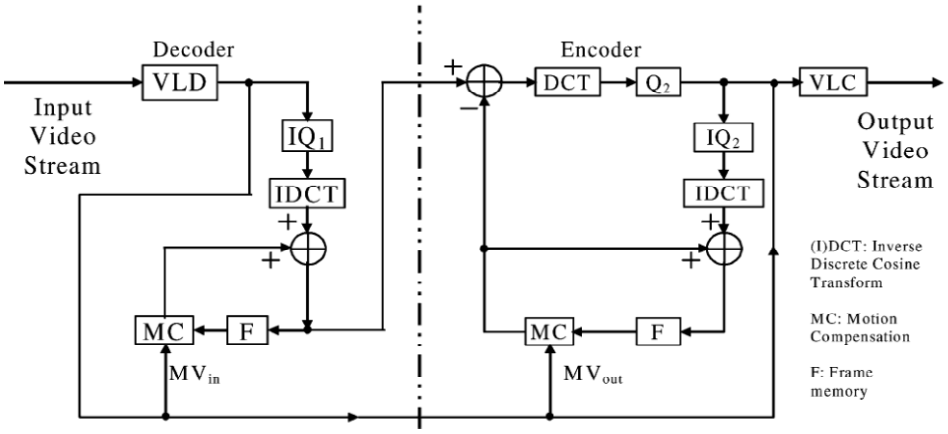
شکل ۱-۳ تطبیق حلقه باز با استفاده از چندی سازی مجدد [5]

۱-۳-۳ تطبیق حلقه بسته در حوزه پیکسل

روش حلقه بسته هم از نظر پیچیدگی محاسباتی و هم از نظر کیفیت نهایی ویدئو بین دو روش Cascade و حلقه باز قرار دارد. در این روش سعی می‌شود که از عملیات پیچیده و وقت گیر کدکردن و کدگشایی صرفنظر شود. اما تطبیق هر فریم با توجه به همان فریم مرجعی صورت می‌گیرد که کدگشا نیز به آن دسترسی خواهد داشت. بعبارت دیگر عملیات جبران حرکت در این روش بصورت کامل انجام می‌شود اما از انجام عملیات تخمین حرکت صرفنظر می‌شود.

به عبارت دقیقتر، در این روش ابتدا رشته بیت دریافتی کدگشایی شده و سپس بجای کدکردن کامل و انجام عملیات زمانبری مثل تخمین حرکت، از بردارهای حرکت موجود در رشته بیت بعنوان بردار حرکت استفاده می‌شود. اما بقیه عملیات کدکردن (خصوصا عملیات جبران حرکت) صورت می‌پذیرد تا مشکل رانش بوجود نیاید. از آنجا که عملیات تخمین حرکت درصد قابل توجهی از عملیات کدکردن را به خود اختصاص می‌دهد (از ۶۰ تا ۷۰ درصد کل عملیات [5]) بنابراین این روش دارای پیچیدگی بسیار کمتری نسبت به روش Cascade است. این در حالی است که بدلیل عدم انجام عملیات تخمین حرکت این روش نمودار نرخ-اعوجاج بدتری نسبت به روش Cascade خواهد داشت. یکی از روش‌های پایه‌ای موجود در این دسته از روش‌های تطبیق به روش MV-Reuse معروف است که بلوک دیاگرام کلی آن در شکل ۱-۴ آمده است. لازم به ذکر است که این روش تطبیق، بعدا در فصل ۶ برای طراحی یک نمونه مطالعاتی کدکننده مطلع از محدودیتهای گیرنده برای کاربردهای تطبیق به کار گرفته خواهد شد.

[1] Drift



شکل ۱-۴ تطبیق حلقه بسته با استفاده مجدد از بردار حرکت (MV_Reuse) [5]

۱-۴ بیان مسئله

تمرکز این رساله بیشتر بر روی کاربردهایی است که یک محتوای ویدئویی برای چند گیرنده ارسال می‌شود. ساختار چنین کاربردی را پیشتر در شکل ۱-۲ نشان داده‌ایم. چنین ساختاری شامل سرویس دهنده، نقاط تطبیق و سرویس گیرنده می‌باشد. سرویس دهنده مسئول تولید محتوای ویدئویی است که آن را بر روی شبکه ارسال می‌کند. سرویس گیرنده نیز مسئول دریافت و کدگشایی ویدئو دریافتی می‌باشد. سرویس گیرنده‌هایی که قابلیت کدگشایی رشته بیت با مشخصات (مثلاً نرخ بیت) ارسالی را دارند، عملیات کدگشائی را بدون هیچ تغییر و دستکاری بر روی رشته بیت دریافتی انجام می‌دهند. برای سرویس گیرنده‌های دیگر که امکان کدگشائی با مشخصات موجود را ندارند رشته بیت تولید شده در تعدادی نقاط میانی تطبیق داده شده و سپس به آنها ارسال می‌شود. در این ساختار، گیرنده‌ها علاوه بر مصرف محتوای ویدئویی امکان تولید و ارسال آن به سرویس دهنده یا سرویس گیرنده‌های دیگر را نیز دارند.

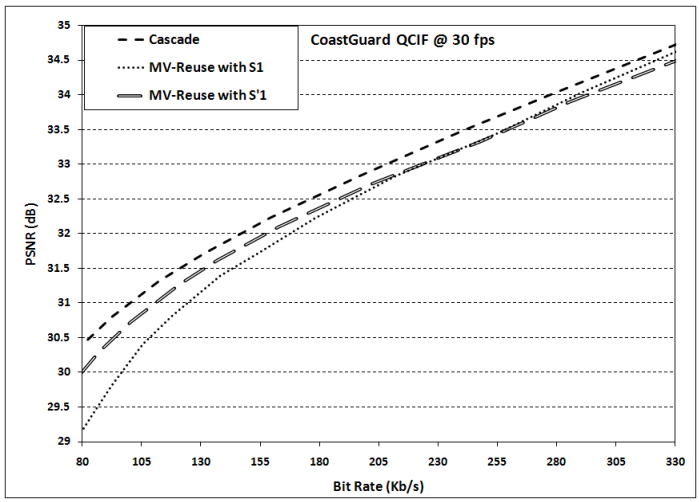
هر گیرنده دارای مشخصات و محدودیتهای مختلفی می‌باشد که در این رساله به منابع[1] شناخته خواهند شد. با توجه به شرایط دستگاه ممکن است منابع متفاوتی اهمیت یابد. معروفترین منبع دستگاه گیرنده نرخ بیت است. از جمله منابع دیگر می‌توان به قابلیت پردازش، توان مصرفی یا پیچیدگی، حافظه موقت، حافظه نهان، نرخ فریم و وضوح نمایش اشاره کرد..

در روش معمول برای این ساختار، تنها مشخصه یک منبع که عملا نرخ بیت یک گیرنده (گیرنده‌ای که بالاترین نرخ بیت را خواهان است) می‌باشد، توسط کدکننده واقع در سرویس دهنده لحاظ می‌شود. نرخ بیت مد نظر گیرنده‌های دیگر با بکارگیری تطبیق دهنده‌ها بدست خواهد آمد. این در حالی است که کدکننده سمت سرویس دهنده می‌توانست بجای در نظر گرفتن یک نرخ بیت، محدودیت نرخ بیت تمامی گیرنده‌ها را در حین تولید رشته بیت در نظر بگیرد. برای ارزیابی ضرورت در نظر گرفتن نرخ بیت تمامی گیرنده‌ها در حین کدکردن یک مثال را در ادامه بررسی می‌کنیم. در این مثال، ویدئو را در یک شبیه سازی با کدکننده استاندارد و با در نظر گرفتن نرخ بیت تنها یک گیرنده فشرده می‌نمائیم که منجر به تولید رشته بیت S'_1 خواهد شد. در شبیه سازی دیگری نرخ بیت تمامی گیرنده‌ها را در حین کدکردن در نظر می‌گیریم، رشته بیت تولید شده

[1] Resource

در این شبیه سازی را S_1 می‌نامیم. سپس رشته بیت تولید شده را به گیرنده‌های مختلف ارسال کرده و در صورت لـزوم پـیش از کدگشایی عملیات تطبیق MV-Reuse را بر روی رشته بیت دریافتی انجام خـواهیم داد. بـا گـزارش نـرخ بیـت و کیفیـت ویدئوهای کدگشایی شده برای نرخ بیتهای مختلف در هر دو روش مذکور، رفتار دو روش را ارزیابی خواهیم کرد.

رفتار این دو روش برای ویدئو نمونه CoastGuard در شکل ۱-۵ آمده است.



شکل ۱-۵ نمودار نرخ بیت – کیفیت برای روشهای تطبیق Cascade، MV-Reuse با رشته بیت اولیه S_1 وMV-Reuse با رشته بیت اولیه S'_1 برای ویدئو نمونه CoastGuard با وضوح نمایش QCIF

همانطور که در این نمودار دیده می‌شود، بصوت کلی یک کدکننده مطلع از محدودیت نرخ بیت تمامی گیرنده‌هـا، بـه میـزان قابل توجهی نسبت به کدکننده معمول بهتر کار می‌کند به عبارت دیگر، طبق شبیه سازیهای صورت پذیرفتـه، کدکننـده‌ای کـه محدودیت نرخ بیت تمامی گیرنده‌ها را در حین کدکردن در نظر می‌گیرد، مجموع اعوجاج کمتری تحمیل می‌نماید. این رفتار بهتر، تنها از در نظر گرفتن نرخ بیتهای تمامی گیرنده‌ها در حین کدکردن ناشی می‌شود. در فصل ۳ جزئیات بیشتری در رابطـه با این شبیه سازی ارائه خواهیم کرد.

این مسئله قابل گسترش به تمامی منابع برای تمامی گیرنده‌ها نیز می‌باشد که هدف اصلی این رساله می‌باشد. به عبارت دیگر، در این رساله نشان خواهیم داد که چنانچه کدکننده در حین تولید رشته بیت محدودیتهای منابع مختلف گیرنده‌ها و همچنـین مشخصات نقطه تطبیق را در نظر بگیرد، رشته بیتی تولید خواهد شد که اعوجاج کلی آن کمتـر از حـالتی خواهـد بـود کـه از کدکننده استاندارد استفاده کنیم. برای ساختار مد نظر این رساله مبحث مطلع بودن کدکننده از محدودیت منابع، قابل اعمـال و تعریف به صورت سه مسئله مجزا می‌باشد. مسئله اول عبارتست از کدکننده‌ای که از محدودیتهای متنوع یـک گیرنـده مطلـع باشد. مسئله دوم طراحی کدکننده مطلع از محدودیتهای متنوع تعداد مختلفی گیرنده ضمن در نظر گرفتن نقطه تطبیق می‌باشد و نهایتا مسئله سوم طراحی کدکننده مطلع از محدودیتهای خویش می‌باشد.

از آنجا که ساختار کلی سه مسئله فوق یکسان می‌باشد، لذا یک روش انتزاعی که بصورت عام روندی برای طراحی کدکننـده مطلع از محدودیت منابع ارائه دهد نیاز است. این روند انتزاعی بصورت سطح بالا اقدام بـه حـل مسـئله و طراحـی کدکننـده خواهد نمود. همچنین این روند انتزاعی با بکارگیری یک روش شناسی مناسب، پارامترهای موثر بر مصرف منـابع را انتخـاب خواهد نمود. مزیت دیگر این روش انتزاعی قابلیت گسترش آن است که آن را بـرای طراحـی کدکننـده مطلـع از محـدودیت

تعداد متنوعی از منابع قابل استفاده می‌نماید. با دنبال کردن این روند انتزاعی و سفارشی سازی آن برای هر یک از سه مسئله بالا می‌توان به کدکننده مد نظر هر مسئله دست یافت. علاوه بر این از آنجا که در پیاده سازی و ارزیابی نیاز به تخمین میزان مصرف منابع داریم، لذا به یک روش شناسی برای تخمین میزان مصرف منابع نیز نیاز خواهیم داشت. این روش نیز مشخصات همه منظوره بودن و انتخاب پارامترهای مناسب برای تخمین مصرف منابع را برآورده خواهد کرد.

با تعریف این دو روند بصورت عام، قابلیت استفاده مجدد آنها در محدوده وسیعی از کاربردها فراهم خواهد شد. به عبارت دیگر، روش انتزاعی پیشنهادی برای طراحی کدکننده مطلع از محدودیتهای منابع در هر ساختاری مربوط به ارسال و دریافت ویدئو که دارای گیرنده/فرستنده با منابع محدود باشد، قابل استفاده است. تنها لازم است که با توجه به شرایط مسئله این روش انتزاعی سفارشی سازی گردد. به همین ترتیب، برای هر کاربردی که به روند اجرا و الگوریتم آن دسترسی داشته، تعریف یا مدل مناسبی از معیار مصرف منبع مربوطه‌اش داشته باشیم، می‌توان با دنبال کردن روش شناسی پیشنهادی برای تخمین مصرف منابع، به مدل تخمین مناسبی دست یافت. در ادامه این بخش، این دو مسئله را با جزئیات بیشتری تشریح خواهیم کرد.

۱-۴-۱ روش انتزاعی طراحی کدکننده مطلع از محدودیت منابع

در این بخش هدف طراحی یک روش سطح بالا برای طراحی کدکننده‌ای می‌باشد که در حین کدکردن محدودیتهای مختلف منابع را لحاظ نماید. چنین روشی در سه حوزه مختلف قابل طرح است که در ادامه به هر یک می‌پردازیم.

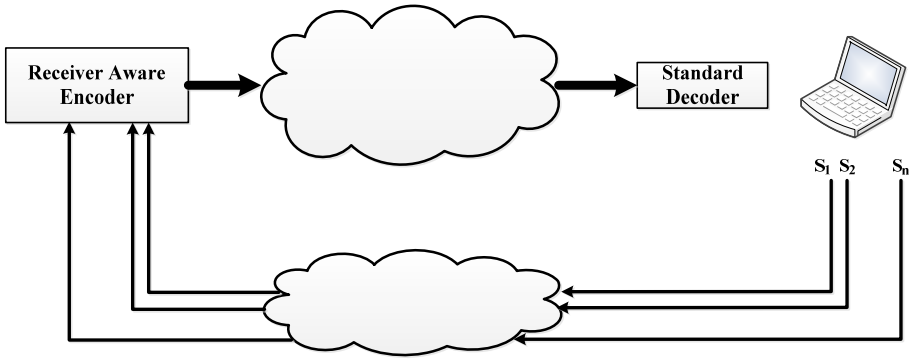
۱-۴-۱-۱ کدکننده مطلع از محدودیتهای گیرنده

در ابتدا اجازه بدهید ساختار ساده تری از شکل ۱-۲ را در نظر بگیریم که در آن تنها کدکننده سمت سرویس دهنده و یک گیرنده (بدون در نظر گرفتن نقطه تطبیق) مشارکت دارند. چنین ساختاری را در شکل ۱-۶ آورده‌ایم.

شکل ۱-۶ ساختار کلی یک کدکننده تک لایه در سمت سرویس دهنده و یک کدگشای تک لایه در سمت گیرنده

در صورتی که کدکننده استفاده شده در این ساختار یک کدکننده تک لایه متداول باشد، تنها مشخصه‌ای نظیر نرخ بیت مد نظر قرار می‌گیرد. این در حالی است که با گسترش کاربردهای چندرسانه‌ای، امکان نمایش ویدئو بر روی ابزارهای مختلف میسر گشته است. لذا علاوه بر مشخصه مذکور، مشخصات دیگری نیز اهمیت یافته‌اند که از جمله می‌توان به توان مصرفی، قدرت پردازنده، و میزان حافظه دستگاه اشاره کرد. از آنجا که عموما محدودیت‌های مذکور بیشتر در سمت گیرنده وجود دارد، لذا بهتر است به محدودیت‌های گیرنده توجه بیشتری داشت. بعنوان مثال چنانچه توان مصرفی گیرنده مدنظر قرار نگیرد، ممکن است دستگاه گیرنده نتواند رشته بیت کدشده را کدگشایی نماید و یا پس از کدگشایی رشته بیت فشرده شده، میزان توان موجود به میزان قابل توجهی کاهش یابد. به طور مشابه، چنانچه پردازنده گیرنده نتواند تعداد محاسبات لازم را در زمان خاصی انجام دهد، امکان کدگشایی رشته بیت بصورت برخط غیر ممکن خواهد بود یا تعداد فریم کمتری در واحد زمان قابل

پخش خواهد بود. چنانچه کدکننده‌ای علاوه بر نرخ بیت، محدودیتهای دیگر گیرنده نظیر آنچه اشاره شد را در حین کدکردن ویدئو در نظر بگیرد به کدکننده مطلع از محدودیتهای گیرنده شناخته خواهد شد. ساختار کلی یک کدکننده مطلع از محدودیتهای گیرنده در ساختار کلی یک کدکننده مطلع از محدودیتهای گیرنده در شکل ۱-۷ نشان داده شده است. همانطور که در شکل آمده است در این ساختار کدکننده متداول تک لایه با یک کدکننده مطلع از محدودیتهای گیرنده[1] جایگزین شده است. این کدکننده با دریافت مشخصات منابع مختلف گیرنده $(S_1, S_2, \dots, S_n)$، رشته بیتی تولید می‌کند که پس از کدگشایی، محدودیتهای گیرنده را لحاظ می‌نماید. طراحی چنین کدکننده‌ای هدف اصلی این بخش می‌باشد.



شکل ۱-۷ ساختار کلی کدکننده مطلع از محدودیتهای گیرنده

۱-۴-۱-۲ کدکننده مطلع از محدودیتهای گیرنده برای کاربردهای تطبیق

مبحث مطرح شده برای کدکننده سمت سرویس دهنده و تنها یک گیرنده، برای کل ساختار آمده در شکل ۱-۲ نیز قابل تعمیم است. از آنجا که در این ساختار تمامی نسخه‌های مربوط به کاربردهای مختلف از روی یک رشته بیت مشترک تولید می‌شوند بنابراین در این حالت کدکننده می‌تواند از مشخصه‌های تمامی‌گیرنده‌ها مطلع باشد تا رشته بیتی تولید کند که پس از تطبیق شدن، محدودیت هر گیرنده را به تناسب مشخصاتش لحاظ کرده باشد. علاوه بر این، کدکننده از نحوه عملیات تطبیق نیز می‌تواند مطلع باشد تا تاثیر آن را بر رشته بیت در نظر بگیرد. این در حالی است که در روش معمول، تنها مشخصات نرخ بیت یک گیرنده در حین کدکردن لحاظ می‌شود.

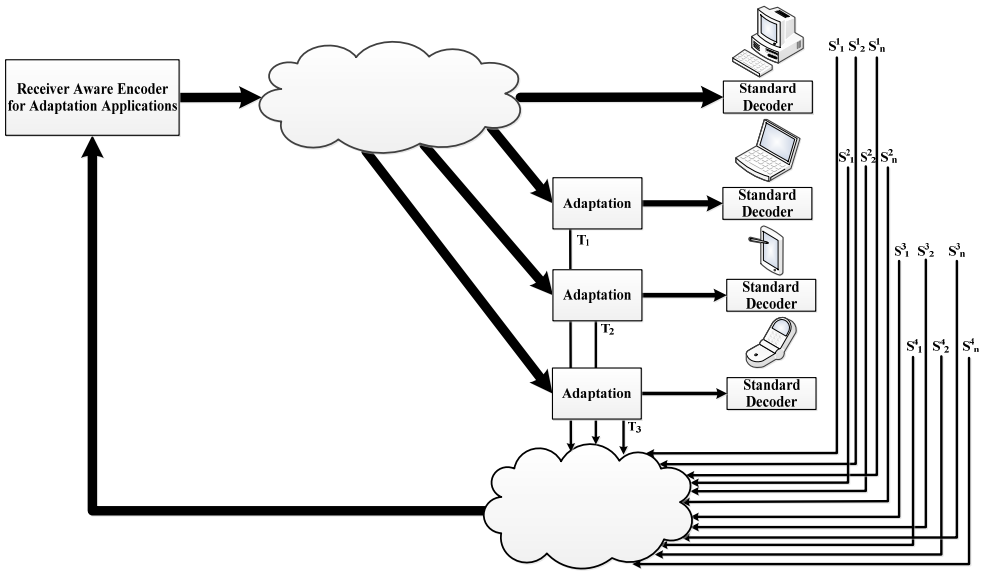
کدکننده‌ای که یک رشته بیت را برای چند کاربرد با مشخصات مختلف به نحوی تولید کند که در عین برآوردن محدودیت‌های تمامی‌گیرنده‌ها مجموع رضایتمندی کاربران بیشینه شود را کدکننده مطلع از محدودیتهای گیرنده برای کاربردهای تطبیق[2] می‌نامیم. چنین ساختاری در شکل ۱-۸ نشان داده شده است. در این ساختار کدکننده RAEA[2] علاوه بر دریافت مشخصات منابع گیرنده‌های مختلف $(S_1^i, S_2^i, \dots, S_n^i)$، نحوه عملیات تطبیق $(T_1, T_2, \dots, T_n)$ را نیز در حین فشرده‌سازی مد نظر قرار می‌دهد. به این ترتیب رشته بیتی که به هر گیرنده می‌رسد، با محدودیتهای گیرنده متناسب خواهد بود.

---

[1] Receiver Aware Encoder (RAE)

[2] Receiver Aware Encoder for Adaptation Applications (RAEA[2])

طراحی کدکننده‌ای که در ساختار شکل ۱-۸ آمده یکی دیگر از مباحث این رساله است. همانطور که مشاهده می‌شود این مسئله نیز بعنوان زیر مجموعه‌ای از روش انتزاعی طراحی کدکننده مطلع از محدودیتهای منابع قرار می‌گیرد.

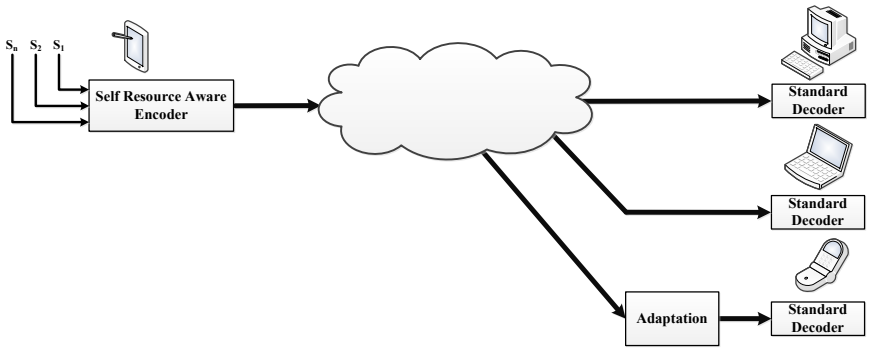


شکل ۱-۸ ساختار کلی کدکننده مطلع از محدودیتهای گیرنده برای کاربردهای تطبیق

۱-۴-۱-۳ کدکننده مطلع از محدودیتهای منابع خویش

در ساختاری که در شکل ۱-۲ آمده است، هر دستگاه نیز مستقلا می‌تواند اقدام به تولید محتوای چندرسانه‌ای بنماید. به عبارت دیگر تمامی دستگاه‌ها هم کدکننده و هم کدگشا را در خود جای داده‌اند. از آنجا که ابزارهای مختلف دارای محدودیتهای متفاوتی می‌باشند، لذا مشخصات و محدودیتهای هر دستگاه نیز می‌تواند در حین کدکردن ویدئو - توسط آن دستگاه – مد نظر قرار گیرد. ساختار کلی این مسئله در شکل ۱-۹ آمده است. در چنین ساختاری، کدکننده دستگاه محدودیتهای خود $(S_1, S_2, \dots, S_n)$ را نیز در حین کدکردن مد نظر قرار می‌دهد. چنین کدکننده‌ای را کدکننده مطلع از منابع خویش[1] می‌نامیم.

طراحی کدکننده‌ای که علاوه بر محدودیت نرخ بیت، محدودیتهای دیگری را از ابزار در حال استفاده در نظر بگیرد مبحث دیگری است که در این رساله بررسی خواهد شد.



شکل ۱-۹ ساختار کلی کدکننده مطلع از منابع خویش

[1] Self Resource Aware Encoder (SRAE)

از آنجا که ماهیت کلی هر سه مسئله اشاره شده بسیار شبیه می‌باشد و در تمامی این مسائل می‌خواهیم محدودیتهای ابزارهای با منابع محدود را در حین کدکردن مد نظر قرار دهیم، لذا در این رساله بنا داریم تا با ارائه یک روند کلی، روشی انتزاعی برای کنترل محدودیت‌های کدکننده و کدگشا در حین کدکردن ارائه بنماییم. سپس روند انتزاعی پیشنهادی را برای هر یک از سه مسئله فوق سفارشی خواهیم کرد. در ادامه برای ارزیابی این روند، برای هر یک از سه مسئله مذکور یک نمونه مطالعاتی با در نظر گرفتن زیر مجموعه‌ای از محدودیتها و نقاط تطبیق پیاده سازی کرده و با انجام شبیه سازیهایی کارایی روند پیشنهادی را نشان خواهیم داد. از آنجا که برای پیاده سازی هر یک از سه مسئله فوق نیاز به تخمین میزان مصرف منابع مختلف داریم و از آنجا که روند تخمین مصرف منابع نیز قابل ارائه بصورت یک رویه سطح بالا می‌باشد، لذا روش شناسی تخمین منابع نوآوری دیگری است که در این رساله آن را پیشنهاد کرده‌ایم و در ادامه به آن خواهیم پرداخت.

۱-۴-۲ روش شناسی تخمین میزان مصرف منابع

در انجام پیاده سازی نمونه‌های مطالعاتی مختلف نیاز به تخمین میزان مصرف منابع مختلف داریم. بعنوان نمونه در کدکننده متداول تک لایه، میزان مصرف نرخ بیت و اعوجاج حاصله در هنگام انجام عملیات انتخاب مد مورد نیاز است. برای این کار سه راه حل قابل تصور است. در ساده ترین روش که در کدکننده مرجع استاندارد H.264/AVC نیز استفاده شده است، می‌توان میزان مصرف واقعی منابع را بصورت دقیق و کامل[1] بدست آورد. به عبارت دیگر در مثال مذکور، نرخ بیت با محاسبه میزان بیتهای تولید شده توسط کدکننده آنتروپی و اعوجاج نیز پس از کدگشایی ویدئو استخراج می‌شود. در روش دیگر می‌توان با استفاده از یک مدل سطح بالا که دقت بالایی نداشته و مشخصات محتوای را در نظر نمی‌گیرد – مانند روابط معمول برای نرخ بیت و اعوجاج – میزان مصرف منابع را تخمین زد. نهایتا در روش سومی می‌توان یک تخمین دقیق با در نظر گرفتن تاثیر هر جزء از عملیات کدکردن و کدگشایی در میزان مصرف منابع، انجام داد. در این رساله روش سوم برای تخمین مصرف برخی منابع – در فرآیند پیاده سازی و ارزیابی – به کار گرفته شده است. با داشتن یک روش تخمین دقیق نه تنها مزیت روش اول (و عیب روش دوم) که محاسبه مصرف منابع با دقت قابل قبول است، بلکه مزیت روش دوم (عیب روش اول) که پیچیدگی محاسباتی انجام عملیات تخمین می‌باشد، نیز برآورده خواهد شد. از آنجا که کلیات عملیات تخمین میزان مصرف منابع در هر سه مسئله فوق یکسان است، لذا یک روش شناسی تخمین میزان مصرف منابع نیز در بخش دوم این رساله پیشنهاد شده است. با دنبال کردن این روند پیشنهادی می‌توان میزان مصرف هر منبعی را تخمین زد.

۱-۵ ساختار رساله

ساختار کلی رساله از دو قسمت تشکیل شده است، در یک قسمت روش انتزاعی طراحی کدکننده مطلع از محدودیت منابع را ارائه خواهیم کرد که خود از سه زیر مسئله تشکیل شده است و در قسمت دیگر، روش شناسی تخمین میزان مصرف منابع را خواهیم داشت. برای نشان دادن عملی بودن این روشها، تعدادی نمونه مطالعاتی نیز ارائه کرده‌ایم که متناسب با مسئله

[1] Exhaustive

مربوطه سفارشی سازی شده‌اند. در فصل دوم رساله به ادبیات تحقیق خواهیم پرداخت و کارهایی که در زمینه دو قسمت مذکور و یا زیر مجموعه‌ای از آنها وجود دارند را معرفی کرده، ضمن برشمردن مزایا و معایب هر یک، نوآوریهای خود را در مقایسه با هر یک از آنها برخواهیم شمرد. در فصل سوم، روش انتزاعی طراحی کدکننده مطلع از محدودیت منابع را خواهیم داشت. پس از ارائه یک روند کلی، این روش را برای مسائل مطرح شده در بخش ۱-۴-۱ سفارشی سازی خواهیم کرد. در بخش چهارم به روش شناسی تخمین میزان مصرف منابع خواهیم پرداخت و یک روند کلی در این زمینه پیشنهاد خواهیم کرد. در همین فصل دو نمونه مطالعاتی را با استفاده از روش شناسی پیشنهادی بررسی و ارزیابی خواهیم کرد. از این دو نمونه مطالعاتی بعدا در فصول پنج تا هفت استفاده خواهیم کرد. در فصل پنجم تا هفتم به پیاده سازی و ارزیابی تعدادی نمونه مطالعاتی که بر اساس روش انتزاعی طراحی کدکننده مطلع از محدودیت منابع طراحی شده اند خواهیم پرداخت. به این ترتیب که در فصل پنجم؛ کدکننده مطلع از محدودیتهای گیرنده را طراحی، پیاده سازی و ارزیابی خواهیم کرد. همین روند را برای کدکننده مطلع از محدودیتهای گیرنده برای کاربردهای تطبیق در فصل ششم انجام خواهیم داد و در فصل هفتم نیز کدکننده مطلع از محدودیتهای منابع خویش، پیاده سازی و ارزیابی خواهد شد. نهایتا در فصل هشتم، به جمع بندی رساله خواهیم پرداخت و کارهای آینده ممکن در رابطه با این رساله پیشنهاد خواهند شد.

۱-۶ جمع بندی

در این فصل فضای کلی این رساله که در ارتباط با ارسال یک محتوای ویدئویی برای چند کاربرد با مشخصات متفاوت می‌باشد تبیین شد. در این فضا روش انتزاعی برای طراحی کدکننده مطلع از محدودیت منابع بعنوان مسئله اصلی این رساله معرفی شد. برای این مسئله سه زیر مسئله نیز معرفی شد که برای ارزیابی روش انتزاعی پیشنهادی مد نظر قرار خواهند گرفت. برای حل این مسائل نیاز به تخمین منابع مختلف برای انجام پیاده سازی می‌باشد که برای چنین تخمینی نیز یک روش شناسی سطح بالا مد نظر قرار خواهد گرفت که با دنبال کردن آن می‌توان منابع مختلف را تخمین زد. حل این مسئله نیز نوآوری دیگر این رساله می‌باشد. در بخش دیگری از این فصل به ساختار رساله و روند ارائه مطالب اشاره شد که در ادامه به هر یک از آنها خواهیم پرداخت. در ابتدا در فصل بعدی به ادبیات تحقیق و کارهای مشابه در زمینه مسائل مطرح شده در این فصل خواهیم پرداخت.

فصل دوم

ادبیات تحقیق

در این فصل به ارائه ادبیات تحقیق خواهیم پرداخت. از آنجا که در این رساله دو مبحث کلی شامل روش انتزاعی طراحی کدکننده مطلع از محدودیت منابع و شیوه شناسی تخمین مصرف منابع مطرح شده است، لذا در این فصل نیز مقالات مربوط به هر یک از این دو مبحث را جداگانه بررسی خواهیم کرد. البته لازم به ذکر است که بر اساس بررسی‌های صورت پذیرفته مقاله‌ای که دقیقا به یکی از دو مبحث این رساله اشاره کند ارائه نشده است بلکه در هر زمینه تعدادی نمونه مطالعاتی ارائه شده است. در بخشهای پیش رو این نمونه‌های مطالعاتی را از منظر این دو صورت مسئله کلی این رساله ارزیابی خواهیم کرد.

۲-۱ ادبیات تحقیق در رابطه با روش انتزاعی طراحی کدکننده مطلع از محدودیت منابع

برای طراحی کدکننده مطلع از محدودیتهای منابع روشهای متنوعی ارائه شده است. برخی از این مقالات در زمینه طراحی کدکننده مطلع از محدودیت‌های خویش [45]-[12] و برخی دیگر در زمینه محدودیت گیرنده و طراحی کدکننده مطلع از محدودیتهای گیرنده [65]-[46] فعالیت نموده‌اند. در برخی از این روشها [19]-[12] عملیاتی برای کنترل مصرف منابع صورت نپذیرفته است، بلکه تنها به روشی جهت کاهش مصرف منابع بسنده شده است. در روشهای دیگر، تلاش شده است که با استفاده از پارامترهای کد کردن، مصرف منابع را در کدکننده و کدگشا کنترل نمایند. در ادامه این دو دسته را بیشتر بررسی خواهیم کرد.

۲-۱-۱ دسته اول – روشهای ارائه شده جهت کاهش مصرف منابع

این دسته از مقالات با بررسی الگوریتمهای کدکردن سعی در کاهش مصرف منابع با ساده کردن واحدهای با محاسبات بالا، نظیر تخمین حرکت، دارند. بعنوان نمونه در مقاله [20]، هر ماکروبلوک با ماکربلوک متناظرش در فریم قبلی مقایسه می‌شود و در صورت شباهت با آن، استنتاج می‌شود که هر دو ماکروبلوک بایستی مد یکسانی داشته شوند. در این حالت از انجام عملیات انتخاب مد صرفنظر خواهد شد. به این ترتیب در این دسته از مقالات یک روند کلی برای کنترل محدودیتهای کدکننده دنبال نشده است، پارامترهای کدکردن بصورت سلیقه‌ای انتخاب شده‌اند و مهمتر از آن میزان تاثیر هر پارامتر بر نرخ بیت و اعوجاج ارزیابی نشده است.

۲-۱-۲ دسته دوم – کنترل مصرف منابع با استفاده از پارامترهای کدکننده

این دسته شامل روش‌هایی می‌شود که مصرف منابع را با تنظیم پارامترهای کدکننده کنترل می‌کنند. در [31]، پیچیدگی کدکننده H.264/AVC برای کاربردهای زمان واقعی کنترل شده است. در این روش، فریم‌هایی که زمان بیشتری برای کدکردن نیاز دارند، حذف می‌شوند. به عبارت دیگر این روش با کاهش نرخ فریم اقدام به کنترل پیچیدگی می‌کند. این روش در [32] گسترش یافته که در آن یک کنترل کننده پیچیدگی دو مرحله‌ای پیشنهاد شده است. در مرحله اول، تمامی ماکروبلوکها با مد ۱۶×۱۶ کد شده‌اند و با توجه به محدودیت توان، تعداد ماکروبلوکهایی که بایستی به مد Skip کد شوند استخراج می‌شود. در مرحله دوم، ماکروبلوکهایی که بایستی به مد Skip کد شوند، انتخاب خواهند شد.

روش R-D-C پیشنهاد شده در [33] با تعریف سطوح مختلف پیچیدگی برای مراحل انتخاب مد و تخمین حرکت اعشاری میزان پیچیدگی را کنترل می‌کند. در [34] پیچیدگی واحد تخمین حرکت با اختصاص وزنهای متفاوت به مدهای مختلف در استاندارد H.264/AVC، میزان محاسبات کنترل می‌گردد.

مولفین [35] و [36] با تنظیم کردن پارامتر محدوده جستجو پیچیدگی را کنترل می‌کنند، اگرچه میزان اعوجاج تحمیل شده برای انتخاب یک مقدار خاص برای این پارامتر گزارش نشده است. در [37] و [38] بهینه‌سازی پیچیدگی–اعوجاج با به کار گرفتن پارامترهایی نظیر دقت عملیات تخمین حرکت، تعداد فریم‌های مرجع و نحوه عملیات چندی سازی انجام گرفته است. در [37] نقاط پیچیدگی–اعوجاج نزدیک به نقاط بهینه و پارامترهای کدکردن متناظر آنها برای هر ویدئو استخراج شده‌اند. با افزایش تعداد پارامترهای کدکردن، نسخه کاملتری از این روش در [38] آمده است. اگرچه این روش بیشتر پارامترهای کدکردن را مد نظر قرار داده است اما فرض کرده است که تاثیر هر یک از پارامترهای کدکردن بر پیچیدگی و اعوجاج مستقل از پارامترهای دیگر است، که فرض نادرستی می‌باشد. به علاوه، این روش منحنی‌های پیچیدگی–اعوجاج را در حالت برخط استخراج نمی‌کند به همین دلیل برای کاربردهای زمان واقعی مناسب نمی‌باشد.

مولفین مقالات [41] و [42] رفتار توان–نرخ بیت–اعوجاج را با بررسی مصرف توان واحدهای کدکننده مطالعه کرده‌اند. مشابه با [37] و [38]، در اینجا نیز برخی از پارامترها نظیر محدوده جستجو و دقت تخمین حرکت در نظر گرفته شده‌اند و تاثیر پارامترهای دیگر از جمله تعداد فریم‌های مرجع را نادیده گرفته‌اند. به علاوه، یک مدل پیچیدگی مستقل بر اساس پارامترهای کدکردن نیز ارائه نشده است. ضمن اینکه، ساز و کار کنترل پیچیدگی در سطح ماکروبلوک نیز در این مقاله ارائه نشده است.

عدم کنترل پیچیدگی در پائینترین سطح ممکن (سطح ماکروبلوک) ریزدانه بودن کنترل کننده را تحت تاثیر قرار خواهد داد و بنابراین امکان کنترل پیچیدگی در حین عملیات انتخاب مد – که عملا مصالحه بین مصرف منابع و اعوجاج در آن انجام می‌شود – میسر نخواهد بود.

در [43]، دو روش بهینه‌سازی نرخ بیت-اعوجاج-پیچیدگی برای بخشهای تخمین حرکت و انتخاب مد ارائه شده است. در این مقاله، مقدار ثابتی برای برخی از پارامترهای کدکردن، نظیر تعداد فریم‌های مرجع و محدوده جستجو، فرض شده است و بنابراین تاثیر آنها در پیچیدگی واحد تخمین حرکت در نظر گرفته نشده است که این مسئله منجر به کاهش دقت روش پیشنهادی آنها شده است.

با استفاده از برخی ابزارهای انجام تخمین حرکت سریع (مانند: ختم زودرس و حذف نقاط جستجو)، ۵ حالت متفاوت برای پیچیدگی در مقاله [44] معرفی شده است. در ابتدای هر فریم، بر اساس پیچیدگی موجود یکی از ۵ سطح پیچیدگی انتخاب شده و پارامترهای واحد تخمین حرکت بر اساس آن سطح تنظیم خواهند شد. واضح است که با داشتن تنها ۵ سطح پیچیدگی، امکان کنترل ریزدانه پیچیدگی ممکن نمی‌باشد که این مسئله، عیب عمده روش آمده در [44] می‌باشد. یک روش کنترل پیچیدگی متوسط دانه، در [45] پیشنهاد شده است که در آن بر اساس پیچیدگی موجود در سطح ماکروبلوک، برای مدهای مختلف یک ماکروبلوک، تعداد فریم‌های مرجع متفاوتی جستجو خواهد شد. پس از این مرحله و بر اساس میزان توان باقیمانده، مدهای Intra ارزیابی خواهند شد.

در [50] پس از ارائه یک مدل پیچیدگی برای کدگشای مقیاس پذیر، این مدل را در طراحی بهینه ساز نرخ بیت، اعوجاج و پیچیدگی کدگشا به کار گرفته‌اند. به عبارت دیگر، سرویس دهنده با استفاده از مدل پیچیدگی، توان مصرفی رشته بیت را استخراج کرده، برخی از بخشهای رشته بیت (از جمله برخی زیر رشته‌های لایه‌های زمانی مختلف) را برای ارسال حذف می‌کند تا محدودیت‌های توانی گیرنده برآورده شود. در [57] هدف طراحی رشته بیتی می‌باشد که توان مصرفی را در سمت گیرنده کاهش دهد. مدل پیچیدگی پیشنهادی تنها بخش درون یابی کدگشا را بعنوان نماینده‌ای برای مصرف توان در نظر گرفته است. در طراحی کنترل کننده پیچیدگی و تعریف پارامتر لاگرانژ برای پیچیدگی از نتایج تجربی و شبیه‌سازی استفاده شده است. در این مقاله رابطه‌ای لگاریتمی بین میزان پیچیدگی و پارامتر لاگرانژ پیچیدگی معرفی شده است. از آنجا که مدل ارائه شده برای پیچیدگی، مدل دقیق و همه جانبه‌ای نمی‌باشد، لذا رابطه بین پیچیدگی و پارامتر لاگرانژ نیز نمی‌تواند رابطه مناسبی باشد. لازم به ذکر است که در این مقاله پارامتر لاگرانژ برای نرخ بیت نیز بررسی نشده است.

برای کاهش مصرف توان در سمت گیرنده مولفین [58] اقدام به طراحی یک روش مقیاس‌پذیر کرده‌اند که طبق آن ویدئو با اندازه کوچک شده کدگشایی می‌گردد. برای انجام این کار، عملیات تخمین حرکت و IDCT دستکاری شده‌اند تا بتوان عملیات کاهش اندازه را بصورت انتخابی انجام داد. واضح است که با کاهش اندازه ویدئو میزان مصرف توان نیز کاهش خواهد یافت. برای کنترل پیچیدگی در این مقاله، ۴ سطح پیچیدگی تعریف شده است که سطح مناسب با توجه به توان مصرفی انتخاب خواهد شد.

در [61] پس از طراحی یک مدل پیچیدگی برای واحد جبران حرکت، به کاربرد مدل پیشنهادی در کنترل پیچیدگی گیرنده اشاره شده است. این مقاله از جمله مقالاتی است که صریحا به طراحی کدکننده مطلع از محدودیت گیرنده و لزوم تعریف

توامان پارامترهای لاگرانژ λ_r و λ_c اشاره کرده است. مولفین این مقاله برای حل مسئله لاگرانژ از نتایج بدست آمده در [57] سود جسته‌اند. همچنین در این مقاله پیچیدگی بعنوان تابعی از مد ماکروبلوک و QP در نظر گرفته شده است، اما از آنجا که پارامتر لاگرانژ را مطابق با [57] در نظر گرفته‌اند، لذا رابطه ریاضی بین پیچیدگی، مد ماکروبلوک و QP ارائه نکرده‌اند. با توجه به اینکه در این مقاله تنها به پیچیدگی واحد جبران حرکت بسنده شده است، کنترل کننده پیچیدگی نیز به همین نسبت ساده بوده و در طراحی کنترل کننده، تاثیر واحدهای دیگر در توان مصرفی مد نظر قرار نگرفته است.

در [62] مدلی برای پیچیدگی واحد کدگشای آنتروپی پیشنهاد شده است، این مدل در [63] تکمیل شده و همچنین کنترل کننده پیچیدگی نیز برای طراحی کدکننده مطلع از محدودیت‌های گیرنده ارائه شده است. در این مقاله همچنین، رابطه‌ای درجه دوم بین پیچیدگی و نرخ بیت معرفی شده است. به این ترتیب با تبدیل پیچیدگی به نرخ بیت دیگر نیازی به تعیین پارامتر لاگرانژ پیچیدگی نبوده و عملا مسئله بهینه‌سازی لاگرانژ برای دو محدودیت (نرخ بیت و پیچیدگی) به بهینه‌سازی برای یک محدودیت (نرخ بیت) تبدیل خواهد شد.

همانطور که واضح است، مقالات موجود در دسته اول، تنها کاهش مصرف منابع را مد نظر قرار داده اند در حالیکه به مباحث کنترل مصرف منابع و انتخاب ترکیبات بهینه اشاره‌ای نکرده‌اند. مقالات موجود در دسته دوم عموما تلاش کرده اند با به کار گرفتن تعدادی از پارامترهای کدکننده، مصرف منابع را کنترل کنند. البته تمامی این روشها تنها بر روی تعدادی نمونه مطالعاتی، خصوصا محدودیت پیچیدگی، تمرکز کرده‌اند و از ارائه یک روند انتزاعی برای کنترل محدودیت منابع غافل بوده‌اند. علاوه بر این در نمونه‌های مطالعاتی خاص نیز کاستی‌هایی دارند. بعنوان نمونه مقالاتی که در زمینه کدکننده مطلع از محدودیتهای خویش پیشنهاد شده اند [43]-[31] پارامترهای کدکردن را بدون در نظر گرفتن تاثیر آنها بر میزان مصرف منبع مربوطه (در اینجا پیچیدگی) انتخاب کرده‌اند. به این ترتیب در برخی از آنها [36]-[31]، پارامترهایی که تاثیر کمتری بر مصرف منابع دارند نظیر پارامتر چندی سازی و مد، در لیست پارامترها وجود دارند. این در حالی است که پارامترهای پر اهمیت‌تر مانند تعداد فریمهای مرجع لحاظ نشده‌اند. علاوه بر این در اکثر مقالات آمده مدل منابع مختلف و نحوه استخراج پارامترهای لاگرانژ نیز ارائه نشده است. ارائه نکردن مدل منابع مختلف و مقدار پارامترهای لاگرانژ امکان پیاده سازی روش پیشنهادی را برای محققین دیگر غیر ممکن می‌سازد.

به طور خلاصه در روشهای پیشنهاد شده در رابطه با طراحی کدکننده مطلع از محدودیتهای منابع مهمترین کاستی عدم ارائه یک روند همه منظوره می‌باشد. بدلیل عدم ارائه چنین روشی، انتخاب پارامترهای کدکننده بدون در نظر گرفتن تاثیر آنها بر میزان مصرف منابع و بصورت سلیقه‌ای صورت گرفته است و لذا موثرترین پارامترها استخراج نشده‌اند. به علاوه، اکثر روشهای ارائه شده قابل گسترش نمی‌باشند. به عبارت دیگر، از آنجا که در روشهای مذکور عموما یک محدودیت منبع (علاوه بر نرخ بیت) مد نظر قرار گرفته است، لذا طراحی کنترل کننده مصرف منابع و بخصوص حل مسئله بهینه سازی با انجام ساده سازی و بدون بررسی کردن زوایای مختلف مسئله صورت پذیرفته است. بعنوان نمونه بجای بدست آوردن روابط مربوط به پیچیدگی و نرخ بیت و همچنین استخراج پارامترهای بهینه سازی برای هر منبع، اقدام به نگاشت پیچیدگی به نرخ

بیت و ساده سازی مسئله نموده‌اند. واضح است که چنین روندی صرفنظر از دقت پائینتر، در صورت اضافه شدن محدودیت منابع دیگر (علاوه بر محدودیتهای فرض شده) قابل استفاده نخواهد بود.

در این رساله روندی ارائه می‌نماییم که کاستی‌های روشهای موجود، از جمله همه منظوره نبودن، نداشتن روش شناسی مناسب برای استخراج پارامترها، عدم امکان گسترش و ریزدانه نبودن، را برطرف نماید. به این منظور، یک روند انتزاعی برای طراحی کدکننده مطلع از محدودیت منابع – خواه این منابع در سمت فرستنده یا گیرنده باشند – پیشنهاد می‌شود. چنین کدکننده‌ای با طی روندی شامل انتخاب پارامترهای کدکردن بر اساس تاثیر آنها بر مصرف منابع، بدست آوردن مدل هر منبع، حل مسئله بهینه سازی و طراحی کنترل کننده طراحی خواهد شد. برای نشان دادن کارائی این روش انتزاعی سه نمونه مطالعاتی مختلف – که مشابه با مقالات موجود در این بخش، محدودیت پیچیدگی را مد نظر قرار داده اند – را پیشنهاد و کدکننده مطلع از محدودیت منابع برای هر یک پیشنهاد خواهیم کرد.

۲-۲ ادبیات تحقیق در رابطه با روش شناسی تخمین میزان مصرف منابع

بسیاری از مقالات آمده در بخش قبل [65]-[49] اقدام به ارائه مدلی برای تخمین میزان مصرف منابع نیز کرده‌اند. این مقالات عموما به تخمین پیچیدگی واحدهای مختلف کدگشا پرداخته‌اند. از آنجا که هر دو نمونه مطالعاتی آمده در این رساله پیچیدگی را مدل می‌کنیم، در این بخش نیز بیشتر مقالاتی را می‌آوریم که در زمینه تخمین پیچیدگی ارائه شده‌اند.

برای اینکه یک مدل تخمین مصرف منابع قابل استفاده در طراحی یک کدکننده مطلع از محدودیت منابع باشد، لازم است که دو خصوصیت را دارا باشد: اولا باید معیار مصرف منابع قابل تبدیل به معیار واقعی داشته باشد، ثانیا بتواند با توجه به خصوصیات پیاده‌سازی و سکو مجددا پیکربندی شود یا به عبارت دیگر یک مدل تخمین همه منظوره باشد.

مدلهای بررسی شده برای پیچیدگی خصوصیات فوق را در نظر نمی‌گیرند. در واقع، اکثر این مدلها بر پایه یک سکو و پیاده‌سازی خاص عمل می‌کنند [65]-[58]. علاوه بر این، برخی از مدلهای مذکور به منظور دستیابی به یک تخمین دقیق‌تر نحوه پیاده‌سازی را تغییر داده و بهینه‌سازی کرده اند [64]-[59].

یکی از مقالات ابتدایی در مدل کردن پیچیدگی کدگشا مقاله [49] می‌باشد، که در آن یک مدل تخمین پیچیدگی با آنالیز کردن یک پیاده سازی نرم افزاری کدگشا، صورت پذیرفته است. این تخمین با در نظر گرفتن تعداد عملیات پایه محاسباتی نظیر، جمع، ضرب، خواندن/نوشتن حافظه بدست آمده است. اگرچه این روش به سکو وابسته نمی‌باشد اما دقت این روش پایین است و تا ۳,۶ برابر کمتر از مقادیر واقعی می‌باشد.

در [57] روشی برای تولید یک رشته بیت که منجر به کاهش مصرف توان در سمت گیرنده می‌شود، ارائه شده است. در این مقاله یک مدل تخمین پیچیدگی برای کدگشا در نظر گرفته شده است که تنها واحد درون یابی را در نظر گرفته است. در مدل پیشنهاد شده حتی پیچیدگی عملیات تخمین حرکت با دقت ¼ و دسترسی به حافظه لحاظ نشده است.

نویسندگان [61] یک مدل تخمین خطی برای عملیات جبران حرکت پیشنهاد کرده اند که در آن از پارامترهایی نظیر تعداد عدم اصابت حافظه نهان، تعداد عملیات درون یابی و تعداد بردارهای حرکت در هر ماکروبلوک استفاده شده است. اگرچه

مدل تخمین پیشنهادی خطایی کمتر از ۱۰ درصد دارد اما مدل تخمین پیشنهادی تنها برای یک پیاده سازی خاص و بهینه سازی شده، قابل استفاده است. پیچیدگی مدل تخمین پیشنهادی نیز کاستی دیگر این روش می‌باشد که نه تنها به یک مدل برای حافظه نهان، بلکه نیاز به انجام محاسبات بسیار زیادی برای آموزش دادن[1] مدل نیز می‌باشد.

در [62] یک مدل نمایی برای تخمین پیچیدگی واحد آنتروپی با استفاده از تعداد بیتهای مصرفی در هر رشته بیت ارائه شده است. این مدل تخمین نیز یک مدل همه منظوره نبوده و قابل استفاده در پیاده سازی و سکوهای مختلف نمی‌باشد.

یک مدل تخمین پیچیدگی خطی برای کدگشای آنتروپی در [63] پیشنهاد شده است که در آن واحدهای CAVLC و UVLC جداگانه مدل شده‌اند. پیچیدگی واحد CAVLC با تعریف پارامترهایی نظیر تعداد اجرای عملیات CAVLC، تعداد ضرایب غیر صفر و تعداد اجراهای run_before تخمین زده شده است. برای واحد UVLC، پارامترهای پیچیدگی نظیر ماکروبلوکهای Skip نشده، تعداد بلوکهای Intra، تعداد بردارهای حرکت و تعداد فریمهای مرجع به کار گرفته شده‌اند. اگرچه برای مدل تخمین ارائه شده در این مقاله میزان خطا کمتر از ۱۰ درصد گزارش شده است، اما نرم افزار مرجع H.264/AVC سفارشی سازی شده است و علاوه بر آن واحد کدگشای آنتروپی بصورت زبان ماشین پیاده سازی شده است. به این ترتیب مدل تخمین پیشنهادی یک مدل همه منظوره نخواهد بود.

یک مدل تخمین پیچیدگی خطی برای فیلتر بلوک زدایی در [64] ارائه شده است. در این مقاله، پیچیدگی با توجه به پارامترهایی نظیر تعداد ماکروبلوکها، تعداد کل لبه‌هایی که فیلتر می‌شوند و تعداد عملیات فیلتر کردن بالا گذر و پایین گذر بدست آمده است. البته در این مقاله تعداد دسترسیهای به حافظه در بدست آوردن تخمین پیچیدگی صرفنظر شده است. به علاوه تعدادی ساده سازیهای غیر واقعی در حین محاسبه تعداد لبه‌های فیلتر شونده نیز در این مقاله به چشم می‌خورد. میزان خطای تخمین این روش نیز در حدود ۱۰ درصد می‌باشد، در حالی که مدل پیشنهادی علاوه بر معایب فوق یک مدل همه منظوره نخواهد بود.

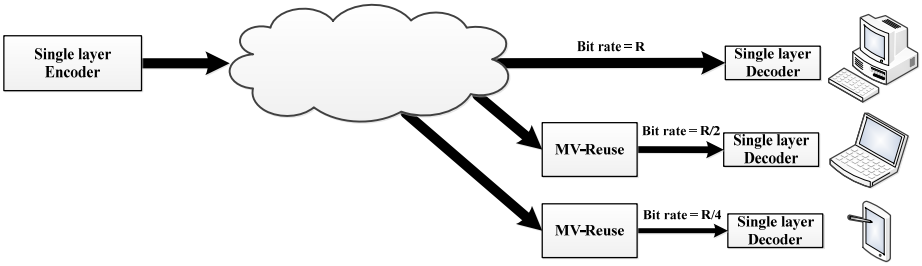
در این رساله با ارائه یک روش شناسی تخمین میزان مصرف منابع، یک روند کلی تخمین ارائه می‌کنیم. در این روش شناسی مشخصاتی که در ابتدای این بخش به آنها اشاره کردیم را لحاظ خواهیم کرد. به عبارت دیگر، روش شناسی پیشنهادی روش تخمینی ارائه می‌کند که مستقل از سکو و پیاده سازی باشد تا بتوان از آن در شرایط مختلف استفاده نمود. پس از ارائه این روش شناسی، با در نظر گرفتن محدودیت پیچیدگی، دو نمونه مطالعاتی برای تخمین مصرف منابع در کدکننده و کدگشای H.264/AVC ارائه خواهیم کرد. همانگونه که در بخشهای مربوطه نشان خواهیم داد، این نمونه‌های مطالعاتی هیچ یک از کاستی‌های روش‌های موجود را نخواهد داشت.

[1] Training

فصل سوم

روش انتزاعی طراحی کدکننده مطلع از محدودیت منابع

در این رساله در فضایی متناسب با کاربردی که در فصل اول معرفی کردیم، مسائل را مطرح و حل خواهیم کرد. در این کاربرد یک محتوای ویدئویی برای تعدادی گیرنده ارسال می‌شود که هر یک دارای منابع محدود مختلفی می‌باشند. ضمن اینکه هر گیرنده خود نیز امکان تولید محتوای ویدئویی را دارد. هدف کلی رساله طراحی کدکننده‌ای برای این فضا است که از محدودیت گیرنده مطلع باشد، خواه اینکه یک گیرنده داشته باشیم، یا تعدادی گیرنده، عملیات تطبیق انجام شود یا خیر و عملیات کدکردن توسط کدکننده در سمت سرویس دهنده صورت گیرد یا در سمت سرویس گیرنده. برای حل این مسائل بصورت سطح بالا، در این فصل یک روش انتزاعی برای طراحی کدکننده مطلع از محدودیت منابع ارائه خواهیم کرد. پیشنهاد کلی در این فصل اطلاع کدکننده از محدودیتهای منابع می‌باشد. با مطلع بودن کدکننده از محدودیتهای منابع محدودیتهای گیرنده(ها)/فرستنده بر آورده خواهد شد ضمن اینکه میزان اعوجاج مجموع نیز کاهش خواهد یافت. برای روشن‌تر شدن موضوع، مسئله را با یک مثال بیان می‌کنیم. همانگونه که در شکل ۳-۱ نشان داده ایم، در این مثال فرض می‌کنیم که یک محتوای ویدئویی برای تعدادی گیرنده که تنها محدودیت آنها نرخ بیت می‌باشد ارسال می‌شود. محدودیت نرخ بیت گیرنده اول R، گیرنده دوم R/2 و گیرنده سوم را R/4 فرض کرده ایم. لازم به ذکر است که رشته بیتهای مربوط به گیرنده دوم و سوم از روی رشته بیت اصلی و با انجام عملیات تطبیق MV-Reuse تولید خواهد شد.



شکل ۳-۱ ساختار کلی کدکننده استاندارد با سه گیرنده با محدودیت نرخ بیت معین

در صورت استفاده از یک کدکننده متداول برای تولید رشته بیت اولیه تنها نرخ بیت R مدنظر قرار خواهد گرفت. سپس تطبیق دهنده، بدون تغییر مد و بردار حرکت ماکروبلوک‌ها تنها عملیات چندی سازی را با پارامتر چندی سازی بزرگتر تکرار خواهد کرد تا رشته بیتهای با نرخ بیت R/2 و R/4 تولید گردند. این در حالی است که درصد توزیع مدهای مختلف در نرخ بیتهای متنوع متفاوت است. به عبارت دیگر در نرخ بیتهای بالا بیشتر ماکروبلوکها به مدهای کوچکتر و در نرخ بیتهای پایین بیشتر ماکروبلوکها به مدهای بزرگتر کد خواهند شد. حال آنکه روش تطبیق بکار گرفته شده هیچ تغییری در توزیع مدها نمی‌دهد و به این ترتیب رشته بیت تطبیق شده به نرخ بیت R/4 از منظر توزیع مدهای مختلف با رشته بیت ایده آل با همین نرخ بیت، تفاوت قابل توجهی دارد. برای ملموس شدن این مسئله درصد توزیع مدهای مختلف در نرخ بیتهای متفاوت را بررسی کرده‌ایم. جدول ۳-۱ توزیع مدهای مختلف را برای تعدادی ویدئو نمونه برای نرخ بیت بالا نشان می‌دهد. توجه کنید که این نتایج برای اندازه QCIF، تعداد ۱۰۰ فریم و همچنین یک فریم مرجع و میانگینی از نرخ بیتهای ۲۵۶ و ۵۱۲ کیلوبیت بر ثانیه بدست آمده است.

جدول ۳-۱ توزیع مدهای مختلف برای نرخ بیتهای بالا

ویدئو نمونه	توزیع مدهای مختلف (درصد)				
	Skip	۱۶×۱۶	۱۶×۸	۸×۱۶	۸×۸
Akiyo	۵۲,۷۵	۲۱,۱۰	۱,۴۶	۱,۷۹	۲۱,۳۷
Carphone	۱۲,۹۴	۲۱,۸۱	۱۰,۰۹	۱۰,۹۷	۴۰,۴۸
Coastguard	۵,۹۲	۳۴,۰۶	۱۲,۷۳	۱۱,۰۳	۳۵,۰۹
Football	۱۸,۰۹	۱۵,۰۴	۶,۶۰	۷,۳۲	۴۵,۹۳
Foreman	۹,۸۱	۲۶,۷۴	۱۰,۰۱	۱۰,۵۳	۴۱,۵۱
Garden	۷,۸۷	۲۷,۱۳	۱۲,۶۲	۷,۶۹	۴۳,۵۵
Grandma	۲۲,۴۶	۳۷,۰۴	۵,۳۹	۵,۶۶	۲۴,۵۶
Mobile	۶,۱۹	۲۸,۷۲	۸,۳۴	۸,۶۲	۴۷,۱۳
Stefan	۱۳,۴۹	۲۷,۸۲	۱۱,۳۷	۸,۴۰	۳۶,۹۱
میانگین	**۱۶,۶۱**	**۲۶,۶۱**	**۸,۰۰**	**۸,۷۳**	**۳۷,۳۹**

در جدول ۳-۲ توزیع مدهای مختلف برای نرخ بیتهای پایین نشان داده شده است. این نتایج برای اندازه QCIF، تعداد ۱۰۰ فریم و همچنین یک فریم مرجع و میانگینی از نرخ بیتهای ۳۲، ۶۴ و ۸۰ کیلوبیت بر ثانیه بدست آمده است.

جدول ۳-۲ توزیع مدهای مختلف برای نرخ بیتهای پایین

ویدئو نمونه	توزیع مدهای مختلف (درصد)				
	Skip	۱۶×۱۶	۱۶×۸	۸×۱۶	۸×۸
Akiyo	۷۲,۸۳	۹,۷۵	۲,۶۶	۳,۳۰	۱۰,۳۳
Carphone	۳۹,۶۱	۲۷,۱۴	۸,۹۰	۹,۷۰	۱۲,۱۳
Coastguard	۳۶,۸۷	۳۲,۹۷	۱۰,۴۵	۹,۶۰	۸,۲۶
Football	۴۴,۲۸	۲۳,۳۹	۸,۳۹	۱۰,۴۳	۶,۹۶
Foreman	۳۳,۵۳	۳۰,۵۵	۱۰,۳۸	۱۳,۱۲	۱۱,۰۹
Garden	۳۳,۴۷	۳۶,۷۴	۱۲,۹۴	۸,۵۷	۶,۹۷
Grandma	۶۵,۴۳	۱۴,۶۵	۳,۹۹	۵,۱۶	۹,۳۳
Mobile	۵۲,۱۰	۲۷,۰۲	۵,۸۵	۶,۷۹	۷,۱۶
Stefan	۴۶,۲۶	۲۶,۹۳	۹,۶۹	۷,۶۳	۷,۴۰
میانگین	**۴۷,۱۵**	**۲۵,۴۶**	**۸,۲۶**	**۸,۱۴**	**۸,۸۵**

بر طبق جداول بالا با کاهش نرخ بیت درصد میانگین ماکروبلوکهای کدشونده به مد ۸×۸ و کوچکتر از آن از ۳۷,۳۹ درصد در نرخ بیتهای بالا به ۸,۸۵ درصد در نرخ بیتهای پایین تغییر یافته است. به این ترتیب واضح است که با استفاده مجدد از بردار حرکت و مد مربوط به رشته بیت کد شده به نرخ بیت بالا (در این مثال R)، برای تولید مد و بردار حرکت برای رشته بیت جدید با نرخ بیت پایین (در این مثال R/2 و R/4)، درصد توزیع مدها مطابق جداول بالا تغییر نکرده و لذا مطابق با نمودار نرخ–اعوجاج حرکت نخواهیم کرد. این مسئله ناشی از عدم در نظر گرفتن مشخصات نرخ بیت گیرنده دوم و سوم در حین تولید رشته بیت اولیه می‌باشد. به عبارت دیگر اگر در حین کدکردن، مشخصه نرخ بیت گیرنده‌های دوم و سوم را نیز در نظر می‌گرفتیم، مدهای ماکروبلوک انتخاب شده با توجه به محدودیت نرخ بیت سه گیرنده انتخاب می‌شد و در نتیجه پس از عملیات تطبیق نیز تفاوت اعوجاج رشته بیتها کاهش می‌یافت. برای نشان دادن عملکرد یک کدکننده مطلع از محدودیت منابع یک مثال دیگر را نیز در اینجا می‌آوریم. در این مثال، همان ساختار آمده در شکل ۳-۱ را در دو حالت فرض می‌کنیم. در حالت اول کدکننده یک کدکننده متداول است و در حالت دوم کدکننده، یک کدکننده مطلع از محدودیت نرخ بیت سه گیرنده می‌باشد. رشته بیتهایی که توسط کدکننده معمولی تولید می‌شوند را S_1، S_2 و S_3 و رشته بیتهای تولید شده توسط کدکننده مطلع از محدودیت نرخ بیت را S'_1، S'_2 و S'_3 می‌نامیم. برای مقایسه این دو روش ابتدا توسط کدکننده متداول رشته بیت S_1 برای نرخ بیت ۴۵۰ کیلوبیت بر ثانیه کد شده است و سپس در نقطه تطبیق رشته بیتهایی با نرخ بیت ۱۷۸ و ۸۴ کیلو بیت بر ثانیه که حدودا نرخ بیتی معادل نصف و ربع رشته بیت S_1 را دارند تولید می‌شوند. همین عملیات توسط کدکننده مطلع از محدودیت نرخ بیت انجام شده و ابتدا رشته بیت S'_1 با نرخ بیت ۴۵۰ کیلوبیت بر ثانیه (با در نظر گرفتن اینکه بعدا قرار است S'_1 به رشته بیتهایی با نرخ بیت ۱۷۸ و ۸۴ کیلوبیت بر ثانیه تطبیق داده شود) تولید شده و سپس در نقطه تطبیق رشته بیتهای S'_2 و S'_3 تولید شده‌اند. تفاوت کیفیت این دو کدکننده در جدول ۳-۳ آمده است.

جدول ۳-۳ نتایج شبیه سازی برای روش معمولی و روش پیشنهادی بر روی ویدئو نمونه CoastGuard

نرخ بیت (Kbps)	کدکننده متداول		کدکننده مطلع از محدودیت نرخ بیت		تفاوت PSNR (dB)
	رشته بیت	PSNR (dB)	رشته بیت	PSNR (dB)	
۴۵۰		۳۷,۲۸۴		۳۷,۰۰۹	−۰,۲۷۵
۱۷۸		۳۲,۲۲۷		۳۲,۳۳۸	+۰,۱۱۱
۸۴		۲۹,۳۷۱		۳۰,۱۶۸	+۰,۷۹۷

همانطور که در جدول نشان داده شده است، رشته بیت S'_1 نسبت به رشته بیت S_1 مقداری افت کیفیت دارد اما S'_2 از S_2 و S'_3 از S_3 کیفیت بهتری دارند، بطوریکه در مجموع ۰,۶۳۳ dB بهبود بدست آمده است.

این مسئله را نیز باید مد نظر قرار داد که اگر چه ما در این مقایسه‌ها افت کیفیت را در نرخ بیتهای مختلف یکسان در نظر گرفتیم اما در عمل اینچنین نیست. به عبارت دیگر افت کیفیت در نرخ بیتهای پایین از اهمیت بیشتری نسبت به افت کیفیت در نرخ بیتهای بالا دارد. این مشخصه وضعیت یک کدکننده مطلع از محدودیت نرخ بیت را نسبت به کدکننده معمولی بهبود می‌دهد. علاوه بر این تعداد گیرنده‌های رشته بیتهای مختلف نیز در این شبیه سازیها مساوی فرض شده است. این در حالیست که ممکن است یک رشته بیت خاص توسط تعداد زیادی گیرنده دریافت شود، لذا بایستی به خراب شدن آن وزن بیشتری اختصاص داد. بعنوان مثال اگر تعداد زیادی گیرنده با نرخ بیت پایین داشته باشیم آنگاه تفاوت عملکرد این دو کدکننده بیشتر مشخص خواهد شد.

۳-۱ روش انتزاعی پیشنهادی

همانگونه که در فصل ۲ – پس از بررسی روشهای موجود و کاستی‌های آنها – اشاره شد، وجود روشی همه منظوره برای طراحی کدکننده مطلع از محدودیتهای منابع ضروری می‌نماید. به عبارت دیگر، نیاز به روشی داریم که بصورت سطح بالا، اقدام به طراحی کدکننده مطلع از محدودیتهای منابع بنماید. در کنار این خصوصیت اصلی، مشخصه مهم دیگر انتخاب پارامترهای کدکننده با دنبال کردن یک روش شناسی مناسب می‌باشد. انتخاب سلیقه‌ای پارامترهای کدکردن منجر به عدم تناسب بین مصرف منابع و اعوجاج تحمیلی و در نتیجه کاهش کارائی کدکننده خواهد شد. عدم امکان گسترش روشهای موجود عیب دیگری بود که در طراحی آنها به چشم می‌خورد. به بیان دیگر، یک کدکننده انتزاعی مطلع از محدودیتهای منابع، مناسب است که فارغ از تعداد منابع موجود و رابطه آنها با یکدیگر طراحی شود و با تغییر تعداد یا روابط منابع مختلف، همچنان قابل استفاده باشد.

حال اجازه بدهید در ابتدا مسئله طراحی کدکننده مطلع از محدودیتهای منابع را بصورت ریاضی بیان نماییم. در این مسئله انتزاعی هدف طراحی کدکننده‌ای است که ضمن برآوردن محدودیتهای مختلف، حداقل مجموع اعوجاج را نیز تحمیل نماید. به این ترتیب مسئله بصورت آمده در رابطه (۳-۱) خواهد بود.

$$\begin{cases} \text{Min } \mathrm{D} = \left(\sum_{i=1}^{n} \mathrm{D_i}\right), \\ \text{s.t. } \{ S^{i,j} < \mathrm{S_c^{i,j}}, i = 1..n, j = 1..m \}, \end{cases} \qquad (۱-۳)$$

در این رابطه، D نشانگر اعوجاج کل، i عبارتست از نمایه برای n گیرنده/فرستنده، j عبارتست از نمایه برای m محدودیت مختلف، $S^{i,j}$ میزان مصرف نهایی منبع j ام از گیرنده/فرستنده iام، $S_c^{i,j}$ مقدار هدف برای محدودیت منبع j ام از گیرنده/فرستنده iام می‌باشند.

یکی از روشهای معمول برای حل مسائل بهینه سازی روش لاگرانژ می‌باشد، که در این رساله نیز از همین روش استفاده کرده و با تبدیل روابط مقید بالا به روابط غیر مقید، مسئله را بصورت انتزاعی حل خواهیم کرد.

برای حل این مسئله بصورت انتزاعی، یک شیوه شناسی پیشنهاد می‌کنیم که روند آن در شکل ۳-۲ آمده است. از آنجا که برای حل یک مسئله بهینه سازی نیاز به دانستن رابطه بین محدودیتهای مختلف (منابع) و تابع هدف (اعوجاج) می‌باشیم لذا به مدلهایی برای این محدودیتها نیاز خواهد بود. از طرفی دیگر، مدل منابع عموما بصورت تابعی از پارامترهای کدکردن تعریف می‌شوند تا بتوان به ازای هر یک از ترکیبات پارامترهای ممکن، میزان مصرف هر منبع را بدست آورد. علاوه بر این، واحد کنترل کننده در کدکننده مطلع از محدودیتهای منابع نیاز به ابزاری برای کنترل مصرف منابع دارد که این ابزار عملا پارامترهای مختلف کدکردن ویدئو خواهند بود. بنابراین در اولین قدم برای حل مسئله بالا، لازم است که پارامترهای کدکردن – با دنبال کردن یک روش شناسی مناسب – انتخاب شوند. در انتخاب پارامترهای کدکردن هدف انتخاب پارامترهایی است که تغییر آنها تاثیر مشهودی بر میزان مصرف منابع و اعوجاج تحمیلی دشته باشد.

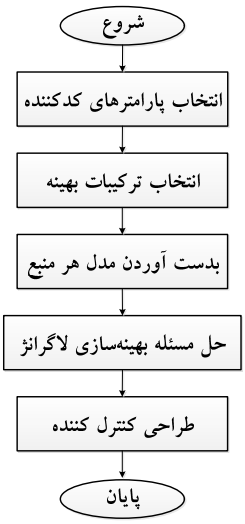
ترکیبات مختلف از پارامترهای متنوع منجر به مصرف منابع و تحمیل اعوجاج متفاوتی خواهند شد، بنابراین اینکه چه ترکیباتی را برای کنترل مصرف منابع مد نظر قرار دهیم، قسمت مهم دیگری است که پس از انتخاب پارامترهای کدکردن باید به آن پرداخته شود.

با انتخاب پارامترهای کدکننده، می‌توان مدل منابع مختلف را بر اساس این پارامترها بدست آورد. این مرحله سومین مرحله از شیوه شناسی پیشنهادی می‌باشد.

با داشتن مدل منابع مختلف بر اساس زیر مجموعه مناسبی از پارامترهای کدکننده، در مرحله چهارم می‌توان مسئله مقید بالا را با روش لاگرانژ حل کرده و پارامترهای لاگرانژ مربوطه را استخراج نمود.

مراحل سوم و چهارم عملا مشخصه گسترش پذیری روش انتزاعی پیشنهادی را برآورده می‌کنند. به بیان دیگر، مدل هر منبع بصورت مستقل و نه بعنوان تابعی از تنها یک منبع دیگر، بدست آمده و همچنین پارامترهای لاگرانژ متناظر با هر محدودیت استخراج خواهند شد.

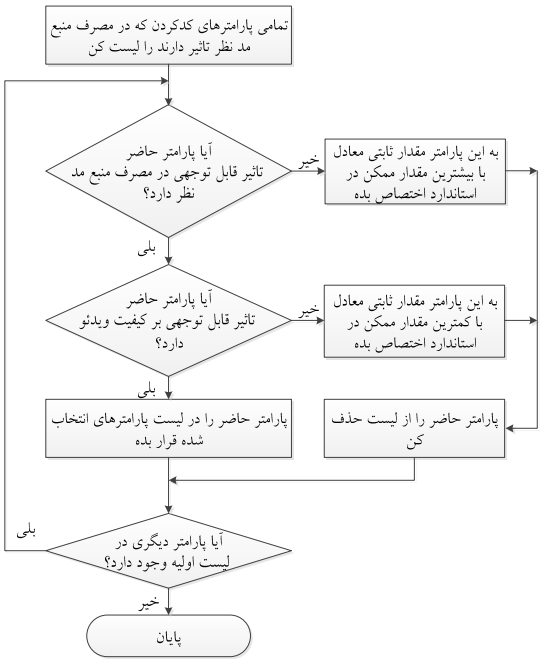
طراحی کنترل کننده مصرف منابع آخرین مرحله از طراحی کنترل کننده مطلع از محدودیت منابع می‌باشد. در این مرحله با اختصاص منابع در مراحل مختلف کدکردن ویدئو و با بکارگیری پارامترهای کنترل کننده، مصرف منابع مطابق با محدودیتهای تعریف شده صورت خواهد پذیرفت. در ادامه هر یک از مراحل مذکور را تشریح خواهیم کرد.



شکل ۳-۲ روند پیشنهادی برای ارائه روش انتزاعی طراحی کدکننده مطلع از محدودیتهای منابع

۳-۱-۱ انتخاب پارامترهای کدکننده

اولین مرحله برای حل مسئله آمده در رابطه (۳-۱) بدست آوردن پارامترهایی کنترلی است که بوسیله آنها بتوان میزان مصرف منابع مختلف را کنترل کرد. همچنین از این پارامترها در مدل کردن رابطه بین منابع نیز می‌توان سود جست. واضح است که چنانچه کنترل منابع و همچنین مدل کردن رابطه بین منابع مختلف با استفاده از تمامی پارامترهای کدکردن صورت پذیرد بهترین نتیجه را خواهد داشت، اما کنترل منابع و بدست آوردن چنین رابطه‌ای بین منابع مختلف غیر عملی می‌باشد، ضمن اینکه تمامی پارامترها مشارکت قابل توجهی در میزان مصرف منابع و نرخ اعوجاج ندارد. به همین دلیل، بهتر است روشی پیشنهاد کنیم که در دنیای واقعی قابل اجرا بوده، ضمن اینکه مهمترین پارامترهای تاثیر گذار روی منابع مختلف را به دست دهد. برای استخراج پارامترها یک روش شناسی پیشنهاد کرده‌ایم که در شکل ۳-۳ آمده است. در روش شناسی استخراج پارامترها، پارامترهایی را لحاظ کرده‌ایم که نه تنها در مصرف منابع محدود، بلکه در کیفیت ویدئو هم نقش دارند. به همین منظور پارامترهایی که تاثیر ناچیزی بر مصرف منابع دارند را از لیست پارامترهای انتخابی حذف کرده و برای اطمینان از عدم تاثیر منفی آنها بر کیفیت، مقدار آنها را برابر بیشترین مقدار قابل تعریف در استاندارد قرار می‌دهیم. همانطور که در شکل ۳-۳ آمده است، روش مشابهی برای پارامترهایی که تاثیر ناچیزی بر کیفیت دارند نیز اعمال شده است. به عبارت دیگر، اگر یک پارامتر تاثیر ناچیزی بر کیفیت دارد، برای جلوگیری از افزایش در مصرف منابع محدود، مقدار ثابتی معادل با کمترین مقدار قابل تعریف خواهد گرفت. واضح است که هر دو دسته پارامترهای مذکور از لیست پارامترهای انتخابی حذف خواهند شد.

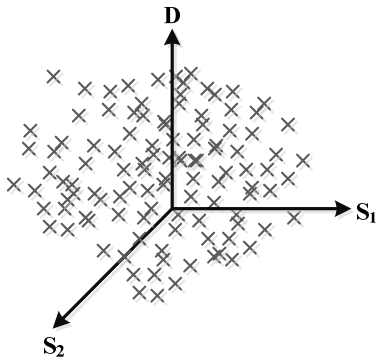


شکل ۳-۳ روش شناسی ارائه شده برای انتخاب پارامترها

۳-۱-۲ انتخاب ترکیبات بهینه

هدف این مرحله حذف ترکیبات نامناسب و انتخاب زیر مجموعه‌ای مناسب از ترکیبات می‌باشد. بعنوان مثال از بین دو ترکیب با مشخصات یکسان از نظر مصرف منابع، ترکیبی ترجیح دارد که اعوجاج کمتری داشته باشد و ترکیب دیگر را بایستی حذف کنیم.

پس از انتخاب پارامترهای موثر، به ازای ترکیبات مختلف از پارامترهای کدکردن، میزان مصرف منابع مربوط به هر محدودیت و اعوجاج حاصله را استخراج می‌نماییم. برای اینکار تعدادی شبیه‌سازی انجام می‌دهیم که در هر شبیه‌سازی کدکننده به یکی از ترکیبات ممکن پارامترهای مختلف، تنظیم شده است. برای تبیین بیشتر مسئله یک نمودار انتزاعی در شکل ۳-۴ آورده‌ایم. در این شکل، دو محدودیت به همراه اعوجاج در نظر گرفته شده‌اند. هر یک از نقاط واقع بر روی نمودار نشانگر یکی از ترکیبات ممکن برای پارامترهای مختلف $(P_1 = p_1, P_2 = p_2, \ldots, P_n = p_n)$ می‌باشد، بطوریکه P_j پارامتر کدکننده و P_j مقدار آن خواهد بود.



شکل ۳-۴ نمودار انتزاعی برای اعوجاج و دو محدودیت

برای بدست آوردن ترکیبات بهینه برای هر منبع j از m منبع موجود، از بین l نقطه مختلف که مصرف تمامی منابع دیگر آنها یکسان می‌باشد ($\forall k: S_k^1 = p_1, S_k^2 = p_2, \ldots, S_k^{j-1} = p_{j-1}, S_k^{j+1} = p_{j+1}, \ldots, S_k^m = p_m, k = 1..l$)، نقطه‌ای را انتخاب می‌کنیم که مصرف منبع j آن کمترین باشد. این کار را برای تمامی منابع تکرار می‌کنیم و به این ترتیب مجموعه ترکیبات بهینه را بدست می‌آوریم.

اگر بخواهیم این مسئله را برای شکل ۳-۴ تشریح نماییم، در صفحه S_1 و D، از بین نقاط با S_1 مساوی نقطه‌ای انتخاب می‌شود که D کمتری داشته باشد. همچنین از بین دو نقطه با D یکسان نقطه‌ای انتخاب می‌شود که S_1 کمتری داشته باشد. این عملیات برای ترکیبات دیگر نیز تکرار خواهد شد. برای این مثال خاص، عملا در نهایت صفحه‌ای خواهیم داشت که تمامی نقاط بهینه را در بر گرفته و نقاط دیگر در بالای آن واقع خواهند بود.

لازم به ذکر است که مقدار مصرف از یک منبع خاص برای دو ترکیب مستقل لزوما برابر نخواهند بود، لذا می‌توان هر منبع را به چند ناحیه تقسیم کرد و با نقاط واقع در هر ناحیه یکسان رفتار نمود.

۳-۱-۳ بدست آوردن مدل هر منبع

با استفاده از ترکیبات بهینه می‌توان رفتار هر منبع را نسبت به منابع دیگر و به ازای تغییر پارامترهای مختلف بدست آورد و نهایتا با استفاده از ابزار برازش منحنی، مدل هر منبع را بصورت تابعی از منابع دیگر استخراج نمود. به عبارت دیگر خواهیم داشت: $S^j = f(S^1, S^2, \ldots, S^{j-1}, S^{j+1}, \ldots, S^m)$.

۳-۱-۴ حل مسئله بهینه‌سازی و استخراج پارامترهای لاگرانژ

با داشتن روابط منابع مختلف می‌توان مسئله بهینه‌سازی آمده در رابطه (۳-۱) را حل کرد. به این ترتیب که رابطه کلی (۳-۲) را کمینه خواهیم کرد.

$$J = \sum_{i=1}^{n} D_i + \sum_{i=1}^{n}\sum_{j=1}^{m} \lambda_{i,j} \times S^{i,j} \qquad (۳-۲)$$

با بدست آوردن $m \times n$ رابطه ناشی از محاسبه $\frac{\partial J}{\partial S^{i,j}}$ برای i و jهای مختلف، این مسئله حل شده و $m \times n$ پارامتر لاگرانژ ($\lambda_{1,1}, \ldots, \lambda_{n,m}$) بدست خواهد آمد. با جایگذاری این روابط در رابطه (۳-۲) و ارزیابی آن در کنترل کننده می‌توان بهترین ترکیبات از پارامترهای کدکردن را بدست آورد.

۳-۱-۵ طراحی کنترل کننده

برای طراحی کدکننده‌ای که با توجه به شرایط مصرف منابع در سرویس گیرنده‌ها، اقدام به عملیات فشرده‌سازی نماید، نیاز به طراحی یک کنترل کننده می‌باشد. این کنترل کننده با در نظر گرفتن میزان منابع باقیمانده در سطوح مختلف، اقدام به تنظیم کردن پارامترهای کدکردن می‌نماید. شبه‌کد انتزاعی مربوط به کنترل کننده و روند اجرای آن در لیست ۳-۱ آمده است. همانطور که در لیست نشان داده شده است، در ابتدای عملیات کدکردن هر فریم، منابع موجود در سطح فریم برای سرویس گیرنده‌های مختلف به متغییر مربوطه ($S_{Frame}^{i,j}$) مقدار دهی خواهند شد. سپس با توجه به میزان منابع مصرف شده در سرویس

گیرنده‌های مختلف ($S_{Frame}^{i,j\ Consumed}$) و منابع موجود، پارامترهای سطح فریم را تنظیم خواهیم کرد. در ادامه عملیات در سطح ماکروبلوک را خواهیم داشت. در این سطح ابتدا منابع موجود مربوطه برای تمامی سرویس گیرنده‌ها در سطح ماکروبلوک بدست آمده و در یکسری متغییر ($S_{MB}^{i,j}$) قرار خواهد گرفت. سپس مشابه با سطح فریم، بایستی پارامترهای این سطح نیز با توجه به منابع مصرف شده و منابع موجود تنظیم گردند. سپس عملیات انتخاب مد بایستی انجام پذیرد که در آن مصالحه بین منابع مختلف در سرویس گیرنده‌های متنوع و مجموع اعوجاج لحاظ می‌گردد.

در انتهای کد کردن هر ماکروبلوک، میزان منابع باقیمانده با توجه به میزان منابع مصرف شده برای سرویس گیرنده‌های مختلف در ماکروبلوک فعلی به روز می‌شود. به طور مشابه پس از کد کردن هر فریم نیز، میزان منابع موجود در این سطح به روز خواهد شد.

For (each frame)
{
$Assign\ available\ frame\ level\ Resources\ for\ all\ clients\ (i.e. S_{Frame}^{1,1}, \dots, S_{Frame}^{n,m})$
Adjust frame level controlling parameters for all clients, based on $S_{Frame}^{i,j\ Consumed}$ *and* $S_{Frame}^{i,j}$
For (each MB)
{
$Assign\ available\ MB\ level\ Resources\ for\ all\ clinets\ (i.e. S_{MB}^{1,1}, \dots, S_{MB}^{n,m})$
Adjust MB level controlling parameters for all clients
For (each mode)
{
$$J_k = \sum_{i=1}^{n} \mathrm{D_i(k)} + \sum_{i=1}^{n}\sum_{j=1}^{m} \lambda_{i,j} \times S^{i,j}(\mathrm{k})$$
Update J_k^{Min} *and Best mode*
}
$Update\ available\ MB\ level\ Resources\ for\ all\ clients\ (i.e. S_{MB}^{1,1}, \dots, S_{MB}^{n,m})$
}
$Update\ available\ frame\ level\ Resources\ for\ all\ clients\ (i.e. S_{Frame}^{1,1}, \dots, S_{Frame}^{n,m})$
}

لیست ۳-۱ شبه‌کد انتزاعی برای کنترل کننده مصرف منابع در کدکننده

با طی کردن روند انتزاعی آمده در این بخش، مراحل طراحی کدکننده مطلع از محدودیتهای منابع بصورت سطح بالا ارائه شد. در ارائه این روند سعی شد که از وابسته بودن به یک استاندارد فشرده سازی ویدئویی خاص، پیاده سازی خاص و یا منابع و محدودیتهای خاص، پرهیز شود تا این روند انتزاعی قابل سفارشی سازی برای کاربردهای متنوع باشد. در ادامه برای سه مسئله مورد نظر این رساله، مدل انتزاعی پیشنهادی را سفارشی سازی خواهیم کرد. بعدا از این روشهای سفارشی سازی شده برای پیاده سازی و ارزیابی تعدادی نمونه مطالعاتی سود خواهیم جست.

۳-۲ سفارشی سازی مدل انتزاعی پیشنهادی برای مسائل مطرح شده در این رساله

در بخش بیان مسئله از فصل مقدمه سه مسئله کلی قابل تعریف در فضای مورد نظر این رساله را معرفی کردیم. در این بخش مدل انتزاعی پیشنهاد شده در بخش قبل را با توجه به مشخصات هر یک از سه مسئله مذکور سفارشی سازی خواهیم کرد.

۳-۲-۱ سفارشی سازی مدل انتزاعی برای کدکننده مطلع از محدودیت‌های گیرنده

برای این مسئله تنها یک کدکننده در سمت سرویس دهنده و یک گیرنده در سمت سرویس گیرنده خواهیم داشت. به این ترتیب برای چنین مسئله‌ای رابطه‌ای بصورت (۳-۳) خواهیم داشت.

$$\begin{cases} \text{Min } (D), \\ \text{s.t. } \{S^j < S_c^j, j = 1..m\}, \end{cases} \quad (۳-۳)$$

در این رابطه، D نشانگر اعوجاج، j عبارتست از نمایه برای m محدودیت مختلف، S^j میزان مصرف نهایی منبع j ام گیرنده، S_c^j مقدار هدف برای محدودیت j ام می‌باشند.

بدست آوردن مدل هر منبع در این مسئله نیز دقیقا با آنچه در روند انتزاعی کلی اشاره کردیم یکسان است و از تکرار این بخش خودداری می‌کنیم.

۳-۲-۱-۱ حل مسئله بهینه‌سازی و استخراج پارامترهای لاگرانژ

با داشتن روابط منابع مختلف می‌توان مسئله بهینه‌سازی آمده در رابطه (۳-۳) را حل کرد. به این ترتیب که رابطه کلی (۳-۴) بایستی کمینه گردد.

$$J = D + \sum_{j=1}^{m} \lambda_j \times S^j \quad (۳-۴)$$

با بدست آوردن m رابطه ناشی از محاسبه $\frac{\partial J}{\partial S^j}$ برای jهای مختلف که هر یک متناظر یک منبع از گیرنده می‌باشند، این مسئله حل شده و m پارامتر لاگرانژ $(\lambda_1, \dots, \lambda_m)$ بدست خواهد آمد. با جایگذاری این پارامترها در رابطه (۳-۴) و ارزیابی آن در حین عملیات انتخاب مد در کدکننده سمت سرویس دهنده می‌توان بهترین مد را از نظر مصالحه منابع-اعوجاج بدست آورد.

۳-۲-۱-۲ طراحی کنترل کننده

کنترل کننده برای کدکننده مطلع از محدودیتهای گیرنده عملا یک ساختار ساده شده از کنترل کننده انتزاعی می‌باشد. شبه‌کد مربوط به این کنترل کننده و روند اجرای آن در لیست ۳-۲ آمده است. همانطور که در شکل نشان داده شده است در ابتدای عملیات کد کردن هر فریم، منابع موجود در سطح فریم به متغیر مربوطه (S_{Frame}^j) مقدار دهی خواهند شد. سپس با توجه به میزان منابع مصرف شده $(S_{Frame}^{j\ Consumed})$ و منابع موجود، پارامترهای سطح فریم را تنظیم خواهیم کرد. در ادامه عملیات در سطح ماکروبلوک را خواهیم داشت. در این سطح ابتدا منابع موجود مربوطه در سطح ماکروبلوک بدست آمده و در یکسری متغیر (S_{MB}^j) قرار خواهد گرفت. سپس مشابه با سطح فریم، پارامترهای این سطح نیز با توجه به منابع مصرف شده و منابع موجود تنظیم می‌گردند. سپس عملیات انتخاب مد انجام می‌پذیرد با این تفاوت که در این کنترل کننده مصالحه بین منابع مختلف و اعوجاج یک گیرنده لحاظ می‌گردد.

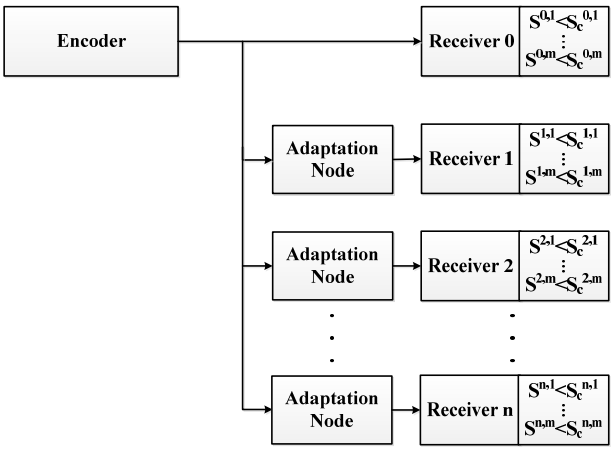
در انتهای کد کردن هر ماکروبلوک، میزان منابع باقیمانده با توجه به میزان منابع مصرف شده در ماکروبلوک فعلی به روز می‌شود. به طور مشابه پس از کد کردن هر فریم نیز، میزان منابع موجود در این سطح به روز خواهد شد.

```
For (each frame)
{
   Assign available frame level Resources (i.e. S^1_Frame, ..., S^m_Frame)
   Adjust frame level controlling parameters based on S^j_Frame^Consumed and S^j_Frame
   For (each MB)
   {
            Assign available MB level Resources (i.e. S^1_MB, ..., S^m_MB)
            Adjust MB level controlling parameters
             For (each mode)
             {
                J_k = D(k) + Σ_{j=1}^{m} λ_j × R^j (k)
                Update J_k^Min and Best mode
             }
                  Update available MB level Resources (i.e. S^1_MB, ..., S^m_MB)
   }
   Update available frame level Resources (i.e. S^1_Frame, ..., S^m_Frame)
}
```

لیست ۳-۲ شبه‌کد انتزاعی برای کنترل کننده منابع گیرنده در کدکننده

۳-۲-۲ سفارشی سازی مدل انتزاعی برای کدکننده مطلع از محدودیت‌های گیرنده در کاربردهای تطبیق

همانطور که در شکل ۱-۸ نشان دادیم، در این ساختار علاوه بر وجود تعداد متنوعی از گیرنده‌ها، عملیات تطبیق نیز بر روی رشته بیت تولید شده صورت می‌پذیرد. برای بیان مسئله بصورت ریاضی، بلوک دیاگرام کلی این ساختار را در شکل ۳-۵ نشان داده ایم. همانطور که در شکل نشان داده شده است، در چنین ساختاری هر گیرنده دارای m منبع خواهد بود که هر منبع محدودیت خاص خود را داراست.



شکل ۳-۵ ساختار کلی برای ارسال یک ویدئو و دریافت آن توسط چند گیرنده پس از عملیات تطبیق

از آنجا که در کدکننده مطلع از محدودیت‌های گیرنده برای کاربردهای تطبیق (RAEA[2]) محدودیت‌های همه گیرنده‌ها و همچنین رفتار نقطه تطبیق را لحاظ می‌کنیم مسئله بهینه‌سازی بصورت رابطه آمده در (۳-۵) خواهد بود.

$$\begin{cases} \text{Min}D = \left(\sum_{i=1}^{n} \alpha_{\text{i}}\, D_i\right), \\ \text{s.t. } \{ S_{\text{T}_\text{i}}^{\text{i,j}} < S_{\text{c}}^{\text{i,j}}, i = 1..n, j = 1..m\}, \end{cases} \qquad (۳-۵)$$

در این رابطه، D_i اعوجاج گیرنده iام و α_{i} ضریب مشارکت گیرنده‌های متنوع می‌باشد و D نشانگر اعوجاج کل می‌باشد. در این رابطه فرض شده است که n گیرنده متمایز داریم که هر یک دارای m محدودیت مختلف می‌باشد. $S_{\text{T}_\text{i}}^{\text{i,j}}$ میزان مصرف نهایی منبع jام از گیرنده iام در حالی که عملیات تطبیق T_i به کار گرفته شده باشد و بالاخره $\text{S}_\text{c}^{\text{i,}j}$ مقدار هدف برای محدودیت jام برای گیرنده iام می‌باشد.

۳-۲-۲-۱ بدست آوردن مدل هر منبع با در نظر گرفتن رفتار نقطه تطبیق

در بدست آوردن رابطه منابع مختلف نحوه تاثیر نقطه تطبیق بر روی منابع مختلف نیز لازم است لحاظ گردد. به عنوان مثال اگر عملیات MV-Reuse را برای تطبیق مد نظر داشته باشیم، بایستی نحوه تاثیر آن بر محدودیت‌های گیرنده نظیر نرخ بیت و یا پیچیدگی مد نظر قرار گیرد. به این ترتیب کدکننده میزان مصرف منابع مختلف را با توجه به نحوه عملیات تطبیق استخراج خواهد کرد. به عبارت دیگر خواهیم داشت: $S_{\text{i}}^{\text{T}} = \text{f}(S_1, S_2, \dots, S_{\text{i}-1}, S_{\text{i}+1}, \dots, S_{\text{n}}, \text{T})$. در این رابطه مدل منبع iام با توجه به عملیات تطبیق T نسبت به منابع دیگر معرفی شده است.

۳-۲-۲-۲ حل مسئله بهینه‌سازی و استخراج پارامترهای لاگرانژ

با داشتن روابط منابع مختلف می‌توان مسئله بهینه‌سازی آمده در رابطه (۳-۵) را حل کرد. به این ترتیب که رابطه کلی (۳-۶) بایستی کمینه گردد.

$$J = \sum_{i=1}^{n} \alpha_{\text{i}} \left(D_i + \sum_{j=1}^{m} \left(\lambda_{i,j} \times S_{\text{T}_\text{i}}^{\text{i,j}} \right) \right) \qquad (۳-۶)$$

لازم به ذکر است که در این رابطه α_{i} میزان مشارکت هر نوع گیرنده متمایز، D_i میزان اعوجاج برای گیرنده iام، $\lambda_{\text{i,j}}$ ضریب لاگرانژ برای منبع jام از گیرنده iام و نهایتا $S_{\text{T}_\text{i}}^{\text{i,j}}$ میزان مصرف منبع jام برای گیرنده iام می‌باشد.

با بدست آوردن $m \times n$ رابطه ناشی از محاسبه $\frac{\partial \text{J}}{\partial S_{\text{T}_\text{i}}^{\text{i,j}}}$ برای i و jهای مختلف، این مسئله حل شده و $m \times n$ پارامتر لاگرانژ $(\lambda_{1,1}, \dots, \lambda_{n,m})$ بدست خواهد آمد. با جایگذاری این پارامترها در رابطه (۳-۶) و ارزیابی آن در حین عملیات انتخاب مد در کنترل کننده می‌توان بهترین مد را از نظر مصالحه منابع–مجموع اعوجاج بدست آورد.

۳-۲-۲-۱ طراحی کنترل کننده

شبه‌کد مربوط به کنترل کننده و روند اجرای آن در لیست ۳-۳ آمده است. همانطور که در لیست نشان داده شده است، در ابتدای عملیات کد کردن هر فریم، منابع موجود در سطح فریم برای گیرنده‌های مختلف به متغیر مربوطه $(S_{Frame}^{i,j})$ مقدار دهی خواهند شد. سپس با توجه به میزان منابع مصرف شده در گیرنده‌های مختلف $(S_{Frame}^{i,j\ \ Consumed})$ و منابع موجود، پارامترهای سطح فریم را تنظیم خواهیم کرد. در ادامه عملیات در سطح ماکروبلوک را خواهیم داشت. در این سطح ابتدا منابع موجود

مربوطه برای تمامی‌گیرنده‌ها در سطح ماکروبلوک بدست آمده و در یکسری متغیر ($S_{MB}^{i,j}$) قرار خواهد گرفت. سپس مشابه با سطح فریم، پارامترهای این سطح نیز با توجه به منابع مصرف شده و منابع موجود تنظیم می‌گردند. سپس عملیات انتخاب مد انجام می‌پذیرد با این تفاوت که در این کنترل کننده مصالحه بین منابع مختلف در گیرنده‌های متنوع و مجموع اعوجاج لحاظ می‌گردد. واضح است که مدل بدست آمده برای منابع مختلف در گیرنده‌های متنوع در محاسبه روابط این مصالحه استفاده خواهد شد.

در انتهای کد کردن هر ماکروبلوک، میزان منابع باقیمانده با توجه به میزان منابع مصرف شده برای گیرنده‌های مختلف در ماکروبلوک فعلی به روز می‌شود. به طور مشابه پس از کد کردن هر فریم نیز، میزان منابع موجود در این سطح به روز خواهد شد.

For (each frame)

{

$Assign\ available\ frame\ level\ Resources\ for\ all\ receivers\ (i.e. S_{Frame}^{1,1}, \dots, S_{Frame}^{n,m})$

Adjust frame level controlling parameters for all receivers, based on $S_{Frame}^{i,j\ \ Consumed}$ *and* $S_{Frame}^{i,j}$

For (each MB)

{

$Assign\ available\ MB\ level\ Resources\ for\ all\ receivers\ (i.e. S_{MB}^{1,1}, \dots, S_{MB}^{n,m})$

Adjust MB level controlling parameters for all receivers

For (each mode)

{

$$J_k = \sum_{i=1}^{n} \alpha_i \left(D_i(k) + \sum_{j=1}^{m} \left(\lambda_{i,j} \times S_{T_i}^{i,j}(k) \right) \right)$$

Update J_k^{Min} *and Best mode*

}

$Update\ available\ MB\ level\ Resources\ for\ all\ receivers\ (i.e. S_{MB}^{1,1}, \dots, S_{MB}^{n,m})$

}

$Update\ available\ frame\ level\ Resources\ for\ all\ receivers\ (i.e. S_{Frame}^{1,1}, \dots, S_{Frame}^{n,m})$

}

لیست ۳-۳ شبه‌کد انتزاعی برای کنترل کننده منابع گیرنده در کدکننده مطلع از محدودیت‌های گیرنده برای کاربرد تطبیق

۳-۲-۳ سفارشی سازی مدل انتزاعی برای کدکننده مطلع از محدودیت‌های منابع خویش

در دو مسئله پیش بر روی کدکننده سمت سرویس دهنده تمرکز داشتیم و محدودیتهای منابع از طرف گیرنده‌های سمت سرویس گیرنده تحمیل می‌شد. اما همانگونه که در شکل ۱-۹ اشاره کردیم، کدکننده می‌تواند بر روی دستگاهی با محدودیت منابع، مانند یک کدکننده ویدئو در تلفن هوشمند، نیز واقع باشد. چنین کدکننده‌ای معمولا بر روی دستگاههای قابل حمل واقع است و می‌تواند از محدودیتهای دستگاه خود مطلع باشد. این مسئله از دید یک مسئله بهینه سازی کاملا با مسئله کدکننده مطلع از محدودیتهای گیرنده، یکسان می‌باشد. به این ترتیب روندی که برای کدکننده مطلع از محدودیتهای گیرنده

سفارشی سازی شد، برای حل این مسئله نیز معتبر خواهد بود و از تکرار آن خودداری می‌کنیم. بعدا در فصل ۷ یک نمونه مطالعاتی را در این زمینه ارائه کرده و روند پیشنهادی را به تفصیل دنبال خواهیم نمود.

۳-۳ جمع بندی

در روشهای موجود برای طراحی کدکننده مطلع از محدودیتهای منابع، برخی از ضرورتهای طراحی کدکننده رعایت نشده است. از جمله این کاستیها می‌توان به انتخاب سلیقه‌ای پارامترهای کدکردن و عدم امکان گسترش روشهای پیشنهادی اشاره کرد. این در حالی است که با توجه به امکان ارسال و دریافت محتوای ویدئویی بر روی ابزارهای مختلف و متنوع، لزوم در نظر گرفتن محدودیتهای گوناگونی در حین تولید رشته بیت در سمت سرویس دهنده، احساس می‌شود. با در نظر گرفتن این مسئله، در این فصل یک روش انتزاعی برای طراحی کدکننده مطلع از محدودیت منابع ارائه کردیم. با ارائه این روند انتزاعی تلاش شده است که کاستی‌های روشهای موجود برطرف گردد. به عبارت دیگر، انتخاب پارامترهای کدکردن بر اساس میزان تاثیر آنها بر میزان مصرف منابع و اعوجاج تحمیلی صورت پذیرد، از بین ترکیبات مختلف پارامترها، ترکیبات با میزان مصرف منابع و اعوجاج تحمیلی معقول انتخاب شوند و اختصاص منابع در سطوح مختلف کدکردن انجام پذیرد. به علاوه روش انتزاعی پیشنهادی برای تعداد نامحدودی منبع و بصورت عام پیشنهاد شده است تا عیب عدم امکان گسترش یافتن روشهای موجود را برطرف سازد.

در ادامه این فصل، برای سه مسئله بیان شده در مقدمه رساله، این روش انتزاعی را سفارشی سازی نمودیم. بعدا در فصول ۵، ۶ و ۷ این روش‌های سفارشی شده را برای چند نمونه مطالعاتی پیاده سازی و ارزیابی خواهیم نمود. از آنجا که در پیاده سازی نمونه‌های مطالعاتی مختلف نیاز به تخمین دقیق میزان مصرف منابع داریم، نمی‌توان از مدلهای کلی که در بخش ۳-۱-۳ در روش انتزاعی پیشنهادی بهره برد. به عبارت دیگر مدلهای آمده در بخش ۳-۱-۳، یک مدلهای سطح بالا و بر اساس تعداد پارامترهای کمی بدست می‌آیند تا بتوان از آنها در بدست آوردن پارامتر لاگرانژ استفاده کرد. این در حالی است که در یک پیاده سازی واقعی نیاز به تخمین دقیق میزان مصرف منابع مختلف خواهیم داشت. در فصل آینده پس از معرفی یک روش شناسی برای تخمین مصرف منابع، دو نمونه مطالعاتی که بعدا در پیاده سازیها از آنها استفاده خواهد شد را معرفی و ارزیابی خواهیم کرد.

فصل چهارم

روش شناسی تخمین میزان مصرف

منابع و نمونه‌های مطالعاتی مربوطه

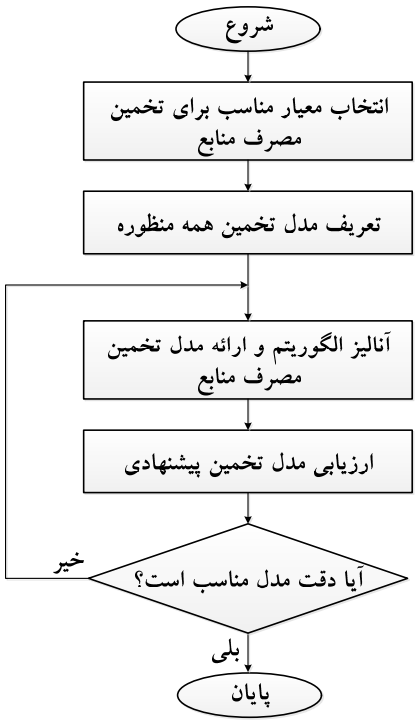
در بسیاری از پیاده سازی‌های عملی نیاز به محاسبه میزان مصرف منابع مختلف داریم. به عنوان نمونه، برای کنترل دینامیک ولتاژ بر اساس توان موجود[1]، نیاز به محاسبه میزان توان مصرفی در هر لحظه وجود دارد. همین طور در کدکننده ویدئو، نیاز به محاسبه میزان نرخ بیت مصرفی و میزان اعوجاج حاصل از آن خواهد بود. در پیاده سازی‌های مربوط به مباحث این رساله نیز به محاسبه میزان مصرف منابع نیاز است. به عبارت دیگر نیاز داریم که بدانیم با توجه به تنظیمات پارامترهای کدکردن، مشخصات گیرنده، و محتوای ویدئو، هر یک از منابع به چه صورتی مصرف می‌شوند. این در حالی است که در مدلهایی که در بخش ۳-۱-۳ معرفی کردیم تنها به یک رابطه کلی بین منابع مختلف بسنده می‌کنیم تا بتوان از این مدلها در حل مسئله بهینه سازی سود جست. واضح است که این مدلها جزئیاتی نظیر مقدار هر یک پارامترهای کدکردن را در نظر نمی‌گیرند. این در حالی است که در حین پیاده سازی هر چه مدل دقیقتری داشته باشیم، تخصیص منابع به شیوه بهتری صورت خواهد پذیرفت.

در این فصل ابتدا یک روش شناسی سطح بالا پیشنهاد می‌کنیم که در آن روند تخمین میزان مصرف منابع مختلف را پیشنهاد می‌کنیم. سپس با در نظر گرفتن توان بعنوان یکی از مهمترین منابع مصرفی، میزان مصرف توان را برای دو الگوریتم در استاندارد H.264/AVC، تخمین خواهیم زد. بعدا در فصول ۵، ۶ و ۷، از این روشهای تخمین استفاده خواهیم کرد.

[1] Dynamic Voltage Scaling

۴-۱ روش شناسی پیشنهادی

برای تخمین میزان مصرف منابع، ملزومات مختلفی باید رعایت شود تا بتوان میزان مصرف منابع را با دقت قابل قبول و در شرایط مختلف تخمین زد. بعنوان نمونه، مدل تخمینی که تنها برای یک پیاده سازی یا بستر خاص معتبر باشد و در بستر یا پیاده سازی دیگر عملیات تخمین مصرف منابع را با دقت قابل قبولی انجام ندهد، مدل مناسبی نخواهد بود. به همین ترتیب معیاری که بر اساس آن عملیات تخمین صورت پذیرفته شده است، بایستی رابطه مستقیمی با میزان واقعی مصرف داشته باشد. با توجه به این ملزومات – همانطور که در شکل ۴-۱ نشان داده شده است – در روش شناسی پیشنهادی، اولین مرحله عبارتست از تعریف یک معیار مناسب برای تخمین مصرف منابع که قابل تبدیل به مقدار واقعی باشد. در مرحله دوم، تعریف مدل تخمینی همه منظوره را خواهیم داشت. در این مرحله، روش تخمین به نحوی تعریف می‌شود که قابل بکارگیری در بسترها و پیاده سازی‌های مختلف باشد. برای تخمین هر منبع، نیاز به بدست آوردن پارامترهایی داریم که در میزان مصرف منابع تاثیر دارند. بنابراین مرحله بعدی عبارتست از بدست آوردن این پارامترها با آنالیز الگوریتم مربوطه و بررسی میزان تاثیر هر یک بر میزان مصرف منابع. با داشتن پارامترهای موثر در مصرف منابع، می‌توان میزان مصرف منابع را تخمین زد. برای اطمینان از دقت قابل قبول تخمین پیشنهادی در یک مرحله جداگانه، مدل تخمین بدست آمده را برای ترکیبات مختلف از سکو و پیاده سازی ارزیابی می‌کنیم. چنانچه دقت مدل تخمین پیشنهادی مناسب باشد، مدل تخمین را نهایی می‌کنیم، در غیر اینصورت به مرحله آنالیز الگوریتم برگشته و در انتخاب پارامترها و ارائه مدل تخمین بازنگری خواهیم کرد. این روند تا رسیدن به دقت قابل قبول تکرار خواهد شد. در ادامه، هر یک از مراحل مذکور را تشریح خواهیم کرد.



شکل ۴-۱ روند پیشنهادی برای ارائه روش شناسی تخمین میزان مصرف منابع

۴-۱-۱ معیار تخمین مصرف منابع

در اولین مرحله معیار مصرف منبعی که می‌خواهیم میزان مصرف آن را تخمین بزنیم تعیین می‌کنیم. از آنجا که در بسیاری از تخمینها محاسبه میزان مصرف منبع با معیار واقعی ممکن نمی‌باشد، از معیارهای دیگری که نزدیک به معیار واقعی باشند استفاده می‌کنند. بعنوان مثال برای توان مصرفی از آنجا که تخمین بر اساس معیار وات میسر نمی‌باشد از معیارهایی نظیر زمان اجرا بعنوان نماینده‌ای از میزان مصرف توان استفاده می‌شود. مهمترین مشخصه معیار مصرف، متناسب بودن آن با معیار واقعی مصرف می‌باشد. به عبارت دیگر امکان تبدیل معیار مصرف منابع به معیار واقعی با خطای کمی قابل انجام باشد. برای مثال مذکور، همانگونه که در ادامه این فصل تشریح خواهیم کرد، زمان اجرا معیار مناسبی نمی‌باشد و معیاری نظیر تعداد پالس ساعت معیار مناسبتری خواهد بود.

۴-۱-۲ تخمین مصرف منابع بصورت همه منظوره

مشخصه دیگری که در تخمین میزان مصرف منابع در نظر گرفته‌ایم، امکان تعریف یک مدل همه منظوره برای تخمین مصرف منابع می‌باشد. مدل همه منظوره، عبارتست از مدلی که برای سکوها و پیاده‌سازیهای مختلف با دقت مناسبی عمل کند. به عبارت دیگر، این مدلی بایستی با تنظیم یکسری پارامتر خاص برای هر سکو یا پیاده‌سازی مد نظر سفارشی گردد. برای داشتن یک مدل همه منظوره، مشابه با [61]، جمع وزن دار تعدادی عملیات پایه در هر الگوریتم را بعنوان پارامترهای مصرف منابع تعریف می‌کنیم. به عبارت دیگر، برای هر الگوریتم مجموعه‌ای از پارامترها تعریف می‌شود که عموما مهمترین عملیات پایه در آن الگوریتم می‌باشند. به ازای هر پارامتر، یک پارامتر وزن نیز خواهیم داشت که مقدار آن با توجه به نوع سکو و پیاده‌سازی تعیین خواهند شد. به این ترتیب مدل همه منظوره برای تخمین میزان مصرف منابع طبق رابطه (۴-۱) خواهد بود.

$$Resource\ Consumption = \sum_{i=1}^{n} P_i \times W_i \qquad (۴-۱)$$

در این رابطه P_i پارامتر مصرف منابع و W_i پارامتر وزن می‌باشد. همانطور که در این رابطه نشان داده شده است، مصرف کلی مجموع وزن دار n پارامتر می‌باشد که n بر اساس الگوریتم تعیین خواهد شد.

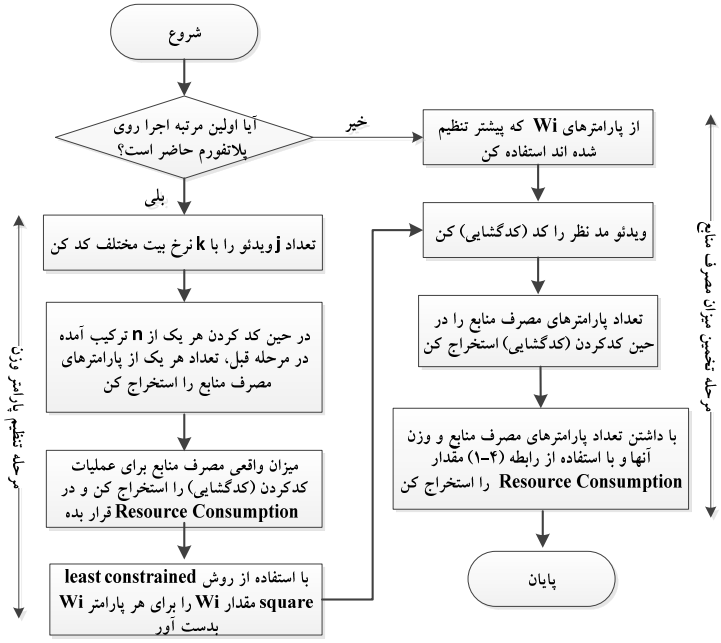
لازم به ذکر است که پارامترهای مذکور، با آنالیز و بررسی الگوریتم تعیین می‌شوند و پارامترهای وزن برای هر سکو یا پیاده‌سازی در یک مرحله اولیه تنظیم می‌شوند که در ادامه به آن اشاره خواهیم کرد.

۴-۱-۲-۱ تنظیم پارامترهای وزن

برای هر سکوی جدید در یک مرحله اولیه عملیات تنظیم وزن صورت می‌پذیرد و برای تمامی شبیه‌سازیهای بعدی از وزنهای استخراج شده در این مرحله استفاده خواهد شد. مجموعه عملیات تنظیم وزن و استخراج مصرف منبع در شکل ۴-۲ آمده است. پیش از توضیح این شکل، لازم به ذکر است که عملیات استخراج مصرف منبع نه تنها در سمت کدکننده، بلکه در سمت کدگشا نیز استفاده دارد و در هر دو سمت، مدل آمده در رابطه (۴-۱) قابل استفاده است. به همین دلیل در شکل ۴-۲ سعی شده است استخراج مصرف منابع برای هر دو سمت معتبر باشد.

در ابتدا تعداد j ویدئو، هر یک با k نرخ بیت مختلف فشرده می‌شوند ($n = j \times k$). در حین کدکردن، تعداد هر یک از پارامترهای مصرف منابع (P_i) برای هر یک از n ترکیب ممکن استخراج می‌شود. سپس، با استفاده از ابزار (نرم افزار) مرتبط میزان واقعی مصرف منابع برای انجام عملیات کدکردن (کدگشایی) تعیین می‌شود که عملا مقدار $Resource\ Consumption$ در رابطه (۴-۱) خواهد بود. با داشتن مجموعه‌ای از پارامترهای پیچیدگی (P_i) برای هر ترکیب و مصرف واقعی معادل آن ($Resource\ Consumption$) تنها مجهول رابطه (۴-۱) وزنهای مربوط به هر پارامتر (W_i) می‌باشد که این وزنها با حل معادله خطی استخراج خواهند شد.

واضح است که برای استخراج میزان مصرف منابع هر ویدئو نمونه جدیدی، از وزنهای استخراج شده در مرحله قبل استفاده خواهد شد، همانگونه که در مرحله تخمین میزان مصرف منابع در شکل ۴-۲ آمده است.



شکل ۴-۲ نحوه تنظیم پارامتر وزن و استخراج میزان مصرف منابع

۴-۱-۳ آنالیز الگوریتم و تخمین مصرف منابع

مهمترین مجهول موجود در رابطه (۴-۱)، پارامترهای مصرف منابع می‌باشد. هدف از این مرحله اولا استخراج این پارامترها و ثانیا نحوه تغییر مصرف منبع با تغییر مقدار این پارامترها می‌باشد که هر دو با آنالیز الگوریتم میسر می‌شود. به عبارت دیگر در این مرحله با آنالیز الگوریتم کدکننده/کدگشا، پارامترهای موثر در مصرف منبع مد نظر را استخراج می‌کنیم. سپس با آنالیز الگوریتم تعیین می‌کنیم که مصرف منابع چگونه بر اساس تغییر این پارامترها، تغییر می‌کند. لازم به ذکر است که آنالیز الگوریتم عموما بر اساس روند پیشنهادی در استاندارد صورت می‌پذیرد و از آنالیز الگوریتم بر اساس یک پیاده سازی خاص اجتناب می‌شود تا تخمین مصرف منابع از پیاده سازی مستقل گردد.

۴-۱-۴ ارزیابی مدل تخمین پیشنهادی

با دنبال کردن مراحل قبل، یک مدل برای تخمین مصرف منابع بدست خواهد آمد. برای استفاده از این تخمین در شرایط مختلف نیاز به اطمینان از دقت آن داریم. برای این منظور، مدل تخمین پیشنهادی را بر روی سکوها و پیاده سازی‌های متنوعی ارزیابی خواهیم کرد. برای انجام ارزیابی، طبق روند آمده در شکل ۴-۲ عمل می‌کنیم. به عبارت دیگر، برای هر پیاده سازی و سکوی جدید، در ابتدا با انجام تعدادی شبیه سازی پارامترهای وزن را تنظیم می‌کنیم. در ادامه برای نمونه ویدئوهای مختلفی، میزان مصرف منابع را با روش پیشنهادی تخمین زده و مقدار حاصل را با مقدار واقعی مصرف منابع مقایسه می‌کنیم. چنانچه میزان خطای تخمین در محدوده قابل قبولی باشد - محدوده قابل قبول ممکن است برای کاربردها و منابع مختلف متفاوت باشد – مدل تخمین پیشنهادی را نهایی فرض می‌کنیم. در غیر اینصورت، به مرحله آنالیز الگوریتم برگشته، پارامترهای استخراج شده را مجددا بررسی کرده و مدل تخمین را بهبود خواهیم داد. به بیان دیگر، ممکن است در تخمین مدل، تعداد مناسبی از پارامترها در نظر گرفته نشده باشد و یا ساده سازیهای اعمال شده در حین عملیات تخمین بیش از حد باشد. با رفع این کاستیها و به روز کردن مدل تخمین بر اساس آنها، عملیات ارزیابی تکرار خواهد شد. این روند تا رسیدن به یک مدل تخمین با دقت مناسب تکرار خواهد شد.

۴-۲ نمونه‌های مطالعاتی - طراحی مدل پیچیدگی

در این بخش با دنبال کردن روش شناسی پیشنهادی، پیچیدگی را برای دو نمونه مطالعاتی تخمین خواهیم زد. از آنجا که در بسیاری از کاربردها با ابزار متحرک سر و کار داریم، مصرف توان اهمیت قابل توجهی پیدا کرده است، لذا مصرف توان یا پیچیدگی بعنوان منبع مصرفی در این نمونه‌های مطالعاتی تخمین زده خواهد شد. در اولین نمونه مطالعاتی پیچیدگی کدگشای H.264/AVC را تخمین خواهیم زد و در نمونه مطالعاتی دیگر، میزان پیچیدگی واحد تخمین حرکت برای کدکننده استاندارد H.264/AVC را تخمین می‌زنیم. از این دو نمونه مطالعاتی بعدا در فصول ۵، ۶ و ۷ استفاده خواهیم کرد.

۴-۲-۱ معیار مصرف پیچیدگی

محققین مختلف معیارهای گوناگونی را برای پیچیدگی معرفی کرده‌اند. در بعضی از مقالات زمان اجرا بعنوان معیار پیچیدگی معرفی شده است. این معیار برای پیاده‌سازی‌هایی که کاملا سخت افزاری هستند معیار مناسبی می‌باشد، اگرچه برای پیاده‌سازیهای مبتنی بر پردازنده یا به عبارت دیگر پیاده‌سازیهای نرم افزاری مناسب نخواهد بود. در برخی دیگر نیز تعداد محاسبات بعنوان معیاری برای پیچیدگی استفاده شده است. این معیار نیز تخمین دقیقی از پیچیدگی در تمامی سکوهای موجود نخواهد بود، چرا که هیچ کدام از این دو معیار ارتباط مستقیمی با رابطه مصرف توان ندارد. برای داشتن یک معیار با دقت بالا در این رساله تعداد پالسهای ساعت پردازنده را بعنوان معیار پیچیدگی در نظر خواهیم گرفت. برتری چنین روشی، علاوه بر دقت آن در پیاده‌سازی‌های سخت افزاری و نرم افزاری، تناسب این معیار با توان مصرفی سیستم می‌باشد. در ادامه این مطلب را به تفصیل تشریح خواهیم کرد.

مصرف توان در مدارهای دیجیتال CMOS عموما نتیجه مصرف استاتیک و دینامیک می‌باشد، این در حالیست که توان دینامیک خود تابعی از فرکانس کاری مدار و میزان سوئیچینگ آن است ([66] [31]. رابطه مصرف توان دینامیک در (۴-۲) آمده است.

$$P_{dyn} \cong V_{DD}^2 \times f_{Clk} \times C_{EFF} \quad (۴-۲)$$

در اینجا، f_{Clk} نمایشگر فرکانس کاری مدار، C_{EFF} حاصلضرب ظرفیت خازنی C_L و فاکتور فعالیت (α) می‌باشد.

توان استاتیک ناشی از I_{sub}، I_j و I_g می‌باشد. رابطه مربوط به توان استاتیک در (۴-۳) آمده است.

$$P_{stc} = L_g\left(V_{dd}I_{sub} + |V_{bs}|I_j + V_{dd}I_g\right), \quad (۴-۳)$$

در این رابطه I_{sub} و I_g به صورت زیر تعیین می‌شوند.

$$I_{sub} = K_3 e^{K_4 V_{dd}} e^{K_5 V_{bs}} \quad (۴-۴)$$

$$I_g = K_6 e^{K_7 V_{dd}} \quad (۴-۵)$$

در اینجا L_g، V_{bs}، I_j، K_3، K_4، K_5، K_6 و K_7 مقادیر ثابت می‌باشند [46].

با در نظر گرفتن یک رابطه خطی بین V_{dd} و f_{Clk}، میزان مصرف توان بعنوان تابعی از V_{dd}، قابل تعریف است. به این ترتیب رابطه (۴-۶) برای توان و ولتاژ قابل تعریف است [46].

$$Power(V_{dd}) = p_1 V_{dd}^{\gamma_1} + p_2 V_{dd} e^{\gamma_2 V_{dd}} + p_3 \quad (۴-۶)$$

در این رابطه $1 < \gamma_1 < 2$، $\gamma_2 > 0$ و p_1، p_2 و p_3 مقادیر ثابت هستند که بر اساس نوع پردازنده تعیین می‌شوند.

با داشتن تعداد پالسهای ساعت مورد نیاز به علاوه زمان کل انجام یک پروسه، می‌توان حداقل فرکانس کاری را تعیین نمود.

با داشتن این فرکانس کاری و ولتاژ مدار و با توجه به رابطه (۴-۶)، توان مدار قابل محاسبه خواهد بود.

لازم به ذکر است که تعداد پالسهای ساعت می‌تواند بعنوان ورودی در روش‌های تغییر دینامیک ولتاژ و فرکانس[1] جهت کنترل توان مصرفی مورد استفاده قرار گیرد [46].

۴-۲-۲ مدل همه منظوره برای پیچیدگی

بر اساس مطالب بیان شده در بخش ۴-۱-۲، یک مدل همه منظوره برای پیچیدگی تعریف خواهیم کرد که بر اساس رابطه (۴-۱) خواهد بود. تنظیم پارامترهای وزن نیز با دنبال کردن روند آمده در شکل ۴-۲ صورت خواهد پذیرفت.

۴-۲-۱ آنالیز الگوریتم و ارائه مدل پیچیدگی

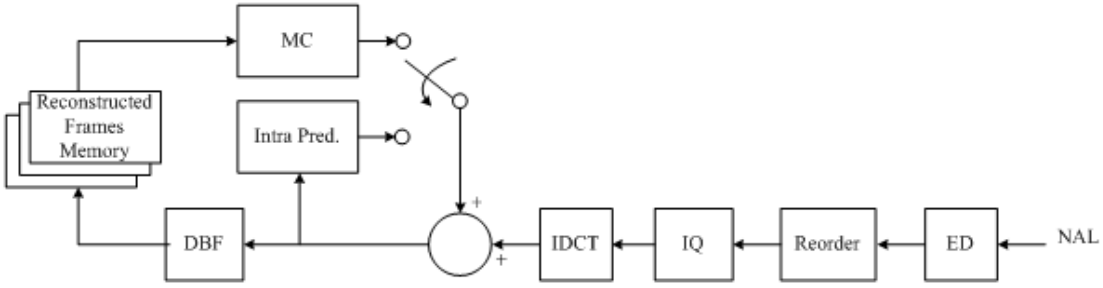
در این بخش هر یک از دو روش مذکور را مستقلا ارزیابی کرده و پارامترهای پیچیدگی هر یک را استخراج می‌کنیم و سپس رابطه کلی تخمین پیچیدگی را ارائه خواهیم داد.

[1] Dynamic Voltage and Frequency Scaling (DVFS)

۴-۲-۱-۱ آنالیز الگوریتم کدگشا H.264/AVC و ارائه مدل پیچیدگی

از آنجا که پیچیدگی از محدودیت‌های مهم گیرنده می‌باشد لذا در این بخش روند تخمین پیچیدگی کدگشا را ارائه می‌کنیم. روش ارائه شده، یک روش همه منظوره است که در آن نه تنها پیاده‌سازی و سکو مشخصی در نظر گرفته نمی‌شود، بلکه هیچ گونه تغییری بر روی پیاده‌سازی کدگشا (به منظور استخراج مدل) اعمال نمی‌گردد. در این بخش، الگوریتم مهمترین واحدهای کدگشای H.264/AVC تحلیل شده و برای هریک از آنها یک مدل پیچیدگی استخراج می‌گردد. بعدا از این مدل در طراحی کدکننده مطلع از محدودیت‌های گیرنده استفاده خواهیم کرد.

بلوک دیاگرام کلی کدگشای H.264/AVC در شکل ۴-۳ نشان داده شده است.



شکل ۴-۳ بلوک دیاگرام کلی کدگشای H.264/AVC

همانطور که در شکل مشاهده می‌گردد، رشته بیت فشرده شده که بصورت بسته‌های NAL بسته بندی شده است، ابتدا به واحد کدگشای آنتروپی (ED) ارسال می‌گردد. سپس ضرایب کدگشایی شده ضبط و جمع آوری گردیده و بعد عملیات پیمانش معکوس و تبدیل DCT معکوس (IDCT) به ترتیب بر روی آنها اعمال می‌گردد تا مانده‌های بازسازی شده و مقداری داده سرآیند مانند مد ماکروبلوک و بردارهای حرکت تولید گردد. در ادامه عملیات کدگشایی، بر اساس مد ماکروبلوک، یکی از عملیات پیش بینی intra یا فرایند جبران حرکت (MC) اعمال شده و سپس ماکروبلوک بازسازی شده به واحد فیلتر بلوک زدایی (DBF) ارسال می‌گردد. در پایان، ماکروبلوک فیلتر شده در یک حافظه ذخیره می‌گردد تا به عنوان مرجع فریمهای بعدی مورد استفاده قرار گیرد.

بر اساس نتایج پروفایلینگ گزارش شده در[58] و [64]، بیشترین بار محاسباتی کدگشا مربوط به واحدهای ED، IDCT، MC و DBF می‌باشد. در این بخش، پیچیدگی این چهار واحد با دنبال کردن شیوه شناسی پیشنهادی این فصل تخمین زده می‌شود. در این بخش، ابتدا الگوریتم هر واحد بر اساس استاندارد H.264/AVC تحلیل می‌گردد. عملیات هر واحد در سطح ماکروبلوک بررسی و کاوش می‌شود و مهمترین زیر واحدهای آن مشخص می‌گردند. این فرایند کاوش تا جایی که زیرواحد تنها شامل عملگرهای پایه مانند افزودن، مجموع، ضرب و تقسیم باشد، ادامه می‌یابد و سپس پارامترهای پیچیدگی واحد مذکور استخراج خواهند شد. برخلاف بیشتر تحقیقات اخیر در این زمینه، پارامترهای ارتباطی نیز همانند پارامترهای محاسباتی هرجا که قابل اعمال باشند در نظر گرفته شده‌اند. در برخی واحدها مانند کدگشایی آنتروپی عملگرهای محاسباتی عملگرهای غالب می‌باشند در حالی که در برخی دیگر مانند جبران حرکت، عملگرهای ارتباطی میزان بیشتری از توان مصرفی کدگشایی را استفاده می‌کنند و همانند عملگرهای محاسباتی باید در نظر گرفته شوند. در جدول ۴-۱، این پارامترها

برای واحدهای مختلف دسته بندی و خلاصه شده‌اند. جزئیات مربوط به هر پارامتر پیچیدگی در بخش مربوطه ارائه خواهد شد.

جدول ۴-۱ پارامترهای پیچیدگی برای واحدهای مختلف

واحد	پارامترهای پیچیدگی
کدگشای آنتروپی	# of Ex_Golomb (N_{Exp_Golomb}) # of CoeffToken ($N_{CoeffToken}$) # of Level (N_{Level}) # of Run_Before (N_{Run_Before})
پیمانش و تبدیل DCT معکوس	# of IDCT operations (N_{IDCT})
جبران حرکت	# of half interpolation (N_{Half}^{MB}) # of quad interpolation (N_{Quad}^{MB}) # of memory access (N_{Mem}^{MB})
فیلتر بلوک زدائی	# of BS4 filters ($N_{F_{Strong}}^{MB}$) # of BS123 filters ($N_{F_{Normal}}^{MB}$) # of memory access (N_{Mem}^{MB})

همانطور که در بخش ۴-۱-۲ اشاره شد، به منظور تخمین مصرف منابع در حالت کلی، لازم است که یک مدل عام منظوره ارائه شود. به همین دلیل رابطه (۴-۷) را به عنوان یک مدل پیچیدگی انتزاعی تعریف می‌کنیم که در آن P_iها پارامترهای پیچیدگی و w_iها وزن‌های مربوط به آنها می‌باشند. همچنین مقدار $Complexity_{Module\,(m)}$ نمایانگر تخمین پیچیدگی مربوط به هر واحد می‌باشد. همانطور که پیشتر اشاره کردیم، در اینجا نیز معیار پیچیدگی تعداد پالسهای ساعت می‌باشد.

$$Complexity_{Module\,(m)} = \sum_{i=1}^{n} P_i \times W_i \qquad (۴-۷)$$

پارامترهای P_i را به راحتی می‌توان در حین فاز کدکردن استخراج نمود. به منظور در نظر گرفتن نحوه پیاده‌سازی کدگشا و سکو طراحی آن، پارامترهای W_i مورد استفاده قرار می‌گیرند. به منظور تنظیم پارامترهای وزنی، برای هر دستگاه خاص، تعدادی شبیه‌سازی اولیه اجرا شده و بر اساس آنها زمان کدگشایی هر رشته بیت استخراج می‌گردد. با داشتن پارامترهای P_i در سمت کدگشا، پارامترهای وزنی برای هر رشته بیت محاسبه می‌گردند. سپس، بهترین ترکیب از W_iها به عنوان پارامتر وزنی آن کدگشای خاص در نظر گرفته خواهد شد. از این لحظه به بعد، برای هر رشته بیت دیگر نیز وزن‌های استخراج شده برای مدلسازی زمان کدگشایی مورد استفاده قرار خواهند گرفت.

استاندارد H.264/AVC چهار گام اصلی را برای کدگشایی یک رشته بیت فشرده شده اجرا می‌کند. در بخش‌های زیر، با توجه به استاندارد H.264/AVC الگوریتم این چهار واحد از نقطه نظر پیچیدگی مورد بررسی قرار خواهد گرفت. در تجزیه و تحلیل هر واحد، از زیرواحدهایی که تاثیر ناچیزی بر روی پیچیدگی دارند چشم پوشی خواهد شد. علاوه بر این، به منظور ساده‌سازی تخمین پیچیدگی، بعضی ساده‌سازی‌ها نیز بر روی واحدها صورت گرفته است که تاثیر این ساده‌سازی‌ها بر روی کاهش دقت تخمین ناچیز است.

۴-۲-۱-۱-۱ تخمین پیچیدگی کدگشای آنتروپی

واحد کدگشای آنتروپی از دو بخش تشکیل شده است که عبارتند از CAVLC و UVLC. بخش UVLC فرایند کدگشایی داده‌های سرآیند مانند CBP، مد ماکروبلوک، بردارهای حرکت، فریم‌های مرجع و مدهای پیش بینی Intra را انجام می‌دهد. از آنجا که این فرایند روش کدکننده Golomb را برای کدگشایی داده سرآیند مورد استفاده قرار می‌دهد، این بخش با نام Exp_Golomb نامیده می‌شود.

بعد از اجرای کدگشای UVLC، واحد CAVLC فرایند کدگشایی ضرایب تبدیل پیمایش شده را در چهار مرحله اصلی انجام می‌دهد. در ادامه، هرکدام از بخش‌های این روند از نظر پیچیدگی توضیح داده خواهد شد و پارامتر پیچیدگی هریک استخراج می‌گردد.

۴-۲-۱-۱-۱-۱ کدگشایی UVLC

رشته بیت فشرده شده در اولین گام به زیرواحد کدگشایی UVLC ارسال می‌گردد. هرکدام از داده‌های سرآیند کد شده دارای ساختار زیر می‌باشد:

[M Zeros][1][INFO]

به عبارت دیگر در هر داده سرآیند، M بیت صفر وجود دارد که توسط یک بیت '1' و داده‌های INFO دنبال می‌شود. عملیات کدگشایی هر سرآیند بصورت زیر خواهد بود:

$$Code\_num = 2^M + INFO - 1$$

پیچیدگی این عملیات به تعداد فراخوانی‌های رویه UVLC (N_{UVLC}) بستگی دارد که تابعی از مد ماکروبلوک، تعداد بردارهای حرکت و تعداد فریم‌های مرجع می‌باشد. تعداد فراخوانی‌های کدگشای UVLC بر اساس ساختار سرآیند ماکروبلوک محاسبه گردیده و نتایج آن در جدول ۴-۲ نشان داده شده است.

برای یک ماکروبلوک از نوع P8x8، ممکن است از ۴ تا ۱۶ بردار حرکت تفاضل (MVD) وجود داشته باشد که بر اساس مدهای زیرماکروبلوک استخراج می‌شوند. همه بلوکهای داخل یک بلوک 8x8 دارای فریم مرجع یکسان می‌باشند. بنابراین، برای یک ماکروبلوک از نوع P8x8، حداکثر چهار فریم مرجع ممکن است وجود داشته باشد. بطور مشابه حداکثر تعداد فریمهای مرجع برای مدهای P16x8 و P8x16 برابر با ۲ می‌باشد. در اینجا N_{UVLC} به عنوان پارامتر پیچیدگی این زیرواحد تعریف می‌گردد که بر اساس معادله (۴-۸) و با استفاده از جدول مذکور محاسبه خواهد شد.

$$N_{UVLC} = f(Mode, MVD, \#\ of\ Refs) \qquad (۴-۸)$$

جدول ۴-۲ تعداد فراخوانی رویه UVLC بر اساس مد ماکروبلوک

مد	توضیح اطلاعات سرآیند	N_{UVLC}
Intra16×16	Joint CBP and Prediction Mode	۱
Intra4×4	Mode + CBP + Prediction mode for Each block	۱+۱+۱۶
Skip	Mode	۱
P16×16	Mode + CBP + MVD[1]x,y + # of Refs[2]	۱+۱+۲+۱
P16×8 P8×16		۱+۱+۴+ # of Refs
P8×8		۵ + ۱ + (# of MVDs) × ۲ + # of Ref

۴-۲-۱-۱-۱-۲ کدگشایی CAVLC

در این زیرواحد، ضرایب تبدیل پیمایش شده با استفاده از تفسیر اطلاعات سرآیند استخراج شده در بخش قبل کدگشایی خواهند شد. در فرمت ویدئوی ۴:۲:۰ هر ماکروبلوک دارای شش بلوک ۸×۸ می‌باشد (چهار بلوک برای مولفه روشنایی و دو بلوک برای مولفه ی رنگ). بنابراین، یک پارامتر CBP شش بیتی تعریف می‌شود که هر بیت آن نشان می‌دهد که آیا یک بلوک ۸×۸ دارای ضریب غیرصفر می‌باشد یا خیر. زیرواحدهای CAVLC زیر روی آن دسته از بلوکهای ۴×۴ اجرا می‌شوند که بلوک ۸×۸ شامل آنها دارای مقدار CBP غیرصفر می‌باشد.

۴-۲-۱-۱-۱-۳ کدگشایی CoeffToken

برای هر بلوک ۴×۴، تعداد کل ضرایب غیرصفر (TotalCoeffs) و تعداد ضرایب Trailing 1's (T1s) با استفاده از یکی از چهار جدول جستجوی موجود استخراج خواهد شد. در هر مرحله کدگشایی، با توجه به تعداد ضرایب غیرصفر بلوکهای ۴×۴ مجاور، جدول جستجوی مناسب انتخاب می‌شود.

اگرچه پیچیدگی کدگشایی این زیرواحد به جدول جستجوی انتخاب شده و پهنای بیت پارامتر CoeffToken بستگی دارد، اما در اینجا بصورت خیلی ساده تعداد اجراهای فرایند کدگشایی CoeffToken را به عنوان پیچیدگی این زیرواحد در نظر می‌گیریم. زیرا، تعداد اجراهای CoeffToken هر ماکروبلوک بطور مستقیم به CBPهایش بستگی دارد. بنابراین، $N_{CoeffToken}$ را به عنوان پارامتر پیچیدگی بصورت زیر تعریف می‌کنیم:

$$N_{CoeffToken} = \sum_{i=1}^{6} 4 \times CBP(b_{8\times8}^{i}) \qquad (۹-۴)$$

۴-۲-۱-۱-۱-۴ کدگشایی یک‌های انتهایی یا trailing 1's

بر اساس استاندارد H.264/AVC، حداکثر سه ضریب با مقدار یک می‌توانند به صورت trailing 1's در نظر گرفته شوند. بنابراین، سایر ضرایب با مقدار یک بصورت یک ضریب عادی کدگشایی خواهند گردید. در این مرحله، با توجه به تعداد

[1] Motion Vector Difference
[2] Number of reference frames

trailing 1's که در بخش قبل کدگشایی شدند، بعضی بیت‌ها از رشته بیت خوانده می‌شوند تا بتوان از طریق آنها علامت trailing 1's را تشخیص داد. از آنجا که این فرایند بسیار ساده می‌باشد، در مدل پیشنهادی از تاثیر آن بر روی پیچیدگی واحد کدگشای آنتروپی چشم پوشی می‌کنیم.

۴-۲-۱-۱-۱-۵ کدگشایی سطح

در این مرحله با استفاده از کدگشایی یک داده پسوند و یک داده پیشوند، عملیات کدگشایی ضرایب غیرصفر انجام می‌گردد. تعداد اجراهای این فرایند برابر است با تعداد کل ضرایب غیرصفر منهای ضرایب یک trailing 1's که در بخش قبلی کدگشایی شدند. بنابراین، پارامتر پیچیدگی فرایند کدگشایی سطح هر ماکروبلوک بصورت نشان داده شده در معادله (۴-۱۰) تعریف می‌شود:

$$N_{Level} = \sum_{i=1}^{6}\sum_{j=1}^{4}\left[TotalCoeffs(b8\times 8^{i}_{b4\times 4^{j}}) - T1s(b8\times 8^{i}_{b4\times 4^{j}})\right] \qquad (۴-۱۰)$$

۴-۲-۱-۱-۱-۶ کدگشایی Run_Before

بعد از کدگشایی تمام مقادیر سطح، تعداد تمام صفرهایی که بلافاصله بعد از بزرگترین ضریب غیرصفر ظاهر شده‌اند، کدگشایی می‌گردند. در مرحله کدگشایی Run_Before بعضی صفرها بین ضرایب غیرصفر قرار می‌گیرند. به عبارت دیگر، در هر اجرای کدگشایی Run_Before تعداد صفرهای بین دو ضریب غیرصفر متوالی محاسبه خواهد شد.

برای تعریف پارامتر کدگشایی Run_Before هفت جدول جستجو تعریف می‌شوند. با توجه به مقادیر صفر باقیمانده جدول مناسبی از بین جداول مذکور انتخاب می‌گردد. پارامتر پیچیدگی این فرایند در سطح ماکروبلوک در معادله (۴-۱۱) نشان داده شده است. همانطور که این رابطه نشان می‌دهد، تعداد اجراهای این فرایند با تعداد کل ضرایب غیرصفر و تعداد کل صفرهای قبل از بزرگترین ضریب غیرصفر (Totzero ها) متناسب می‌باشد.

$$N_{Run\_Before} = \sum_{i=1}^{6}\sum_{j=1}^{4} f(TotalCoeffs(b8\times 8^{i}_{b4\times 4^{j}}), \mathrm{Totzeros}(b8\times 8^{i}_{b4\times 4^{j}})) \qquad (۴-۱۱)$$

مقدار پارامترهای پیچیدگی فوق (یعنی N_{UVLC}، $N_{CoeffToken}$ ، N_{Level} و N_{Run_Before}) را می‌توان به سادگی در طی مرحله کدگشایی استخراج نمود. حال، با داشتن مقادیر این پارامترها می‌توان یک مدل پیچیدگی برای تمام کدگشایی‌های آنتروپی ارائه نمود. به این ترتیب، یک مدل خطی پیچیدگی برای واحد کدگشایی آنتروپی بصورت زیر ارائه می‌شود:

$$\mathrm{C_{Entropy\ Decoder}} = \mathrm{W}_{UVLC}\times N_{UVLC} + \mathrm{W}_{CoeffToken}\times N_{CoeffToken} + \mathrm{W}_{Level}\times N_{Level} + \mathrm{W}_{Run\_Before}\times N_{Run\_Before} \qquad (۴-۱۲)$$

متغیرهای W_{UVLC}، $\mathrm{W}_{CoeffToken}$، W_{Level} و W_{Run_Before} در معادله (۴-۱۲) پارامترهای پیچیدگی وزنی هستند که می‌توان آنها را بر اساس مشخصات سکو طراحی و نحوه پیاده‌سازی تنظیم نمود. همانطور که در بخش ۴-۱-۲-۱ توضیح داده شده است، به ازای هر سکو طراحی و پیاده‌سازی ابتدا تعدادی شبیه‌سازی اولیه اجرا می‌شود تا وزن هر پارامتر پیچیدگی استخراج گردد و سپس از وزن‌های بدست آمده برای دیگر رشته بیتها نیز استفاده خواهد شد.

۴-۲-۱-۱-۲ تخمین پیچیدگی پیمانش و تبدیل DCT معکوس

بعد از اجرای فرایند کدگشایی آنتروپی بر روی هر ماکروبلوک، عملیات IDCT اجرا خواهد شد که طی آن برای هر بلوک ۴×۴ یک عملیات IDCT صحیح و سپس یک فرایند re-scaling اعمال می‌گردد. عملیات IDCT روی بلوک‌هایی که دارای غیرصفر می‌باشند با توجه به پارامتر CBP اجرا می‌گردد. به منظور مدلسازی پیچیدگی IDCT می‌توانیم تعداد IDCTهای ۴×۴ و re-scaling برای هر ماکروبلوک را محاسبه کنیم. بنابراین، پارامتر پیچیدگی برای واحد IDCT بصورت نشان داده شده در معادله (۴-۱۳) خواهد بود:

$$N_{IDCT} = \sum_{i=1}^{6} 4 \times CBP(b_{8\times8}^{i}) \qquad (۴-۱۳)$$

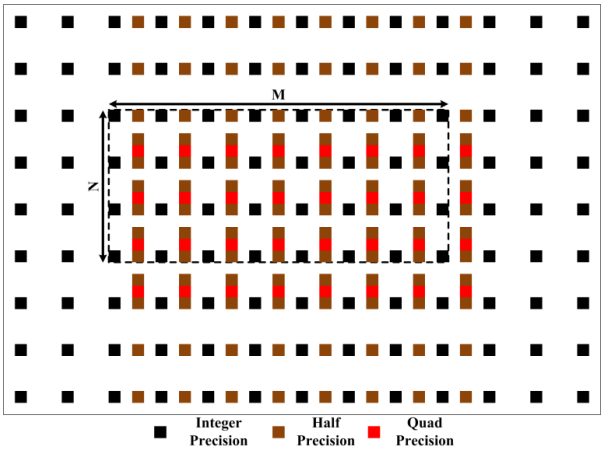
حال با استخراج وزن پارامتر پیچیدگی فوق، می‌توان یک مدل پیچیدگی برای عملیات IDCT ارائه کرد که بصورت نشان داده شده در معادله (۴-۱۴) می‌باشد.

$$C_{IDCT} = W_{IDCT} \times N_{IDCT} \qquad (۴-۱۴)$$

۴-۲-۱-۱-۳ تخمین پیچیدگی جبران حرکت

برای هر ماکربلوکی که بصورت بین فریمی کد شده است، فرایند جبران حرکت اجرا می‌گردد. فرایند MC بر روی تمام بلوک‌های داخل یک ماکروبلوک اعمال می‌گردد. فرایند MC برای هر بلوک MxN با استخراج مقدار دقیق بردار حرکت MV (MV_x و MV_y) برای آن بلوک از بردار حرکت پیش بینی شده و بردار حرکت تفاضل شروع می‌شود. سپس، بر اساس اندازه بلوک و دقت MV یک پیکسل مناسب از حافظه فریم قبلی خوانده می‌شود. با توجه به دقت MV_x و MV_y یک فیلتر شش مرحله‌ای درونیابی نیم پیکسل و/یا یک فیلتر دو مرحله‌ای درونیابی ربع پیکسل ممکن است اعمال گردد. سرانجام، با توجه به مقدار MV_x و MV_y یک بلوک MxN از ناحیه درونیابی شده به عنوان بلوک جبران حرکت استخراج خواهد شد. وقتی که هر دوی MV_x و MV_y دارای دقت در سطح عدد صحیح می‌باشند، از عملیات درونیابی، صرفنظر شده و عملیات MC به سادگی بصورت ذخیره یک ناحیه MxN از حافظه فریم با آدرس متناظر با MV_x و MV_y در می‌آید.

از منظر پیچیدگی، مراحل اصلی واحد جبران حرکت (MC) عبارتند از دو فرایند درونیابی و دستیابی به حافظه. با توجه به اینکه عملیات درونیابی در سطوح نیم پیکسل و ربع پیکسل متفاوت است، این دو حالت بصورت جداگانه بررسی و تحلیل خواهند شد. بنابراین، پارامترهای پیچیدگی واحد MC عبارتند از: تعداد عملیات درونیابی نیم پیکسل (N_{Half}^{MB})، تعداد درونیابی‌های ربع پیکسل (N_{Quad}^{MB}) و تعداد دستیابی‌های داخلی و خارجی حافظه ($N_{Int-Mem}^{MB}$ و $N_{Ext-Mem}^{MB}$). به منظور ارائه یک مدل پیچیدگی برای فرآیند جبران حرکت، لازم است که پارامترهای پیچیدگی فوق ابتدا در سطح بلوک MxN محاسبه شوند (N_{Half}^{b}، N_{Quad}^{b} و $N_{Int-Mem}^{b}$). در اینجا، با استفاده از تصویر نشان داده شده در شکل ۴-۴ سعی می‌شود که تعداد درونیابی‌های سطح نیم پیکسل و ربع پیکسل و همچنین تعداد دستیابی‌های داخلی و خارجی به حافظه، برای یک بلوک ۴×۸ در حالتی که بردار حرکت کسری (۰,۷۵ و ۰,۵) است، بررسی و تجزیه و تحلیل شود.



شکل ۴-۴ تعداد درون یابی‌های نیم پیکسل و ربع پیکسل و دسترسی به حافظه برای جبران حرکت یک بلوک ۸×۴

اگرچه در شکل ۴-۴ یک بلوک ۸×۴ و پیکسلهای مورد نیاز اطراف آن نشان داده شده‌اند، ولی در اینجا یک رابطه ی کلی برای هر بلوک M×N استخراج خواهد شد. در این مثال هر ربع پیکسل با اعمال یک فیلتر دو مرحله‌ای بر روی دو نیم پیکسل بالا و پایین آن تولید می‌شود. بنابراین تعداد درونیابی‌های در سطح ربع پیکسل برابر خواهد بود با M×N. نقاط با دقت نیم پیکسل که در بالای هر ربع پیکسل قرار دارند، با اعمال یک فیلتر شش مرحله‌ای عمودی بر روی نیم پیکسل‌هایی حاصل می‌شوند که خودشان با اعمال فیلترهای شش مرحله‌ای افقی بر روی نقاط صحیح حاصل شده‌اند. همانطور که در شکل ۴-۴ مشاهده می‌شود، برای تولید تمام ربع پیکسل‌های مورد نیاز، لازم است که به تعداد M×(2×N+5) نقطه با دقت نیم پیکسل تولید شوند. علاوه بر موارد فوق، پیکسل‌های صحیح نمایانگر تعداد دستیابی‌ها به حافظه برای این عملیات درونیابی می‌باشد که برای یک بردار حرکت با دقت نیم پیکسل در راستای افقی و دقت ربع پیکسل در راستای عمودی، تعداد آنها برابر است با (M+5)×(N+5).

پارامترهای پیچیدگی مورد نیاز برای جبران حرکت یک بلوک با اندازه دلخواه M×N به ازای تمام ترکیبات ممکن MV_x و MV_y استخراج شده و در جدول ۴-۳ نشان داده شده‌اند. در این جدول حروف I، H و Q به ترتیب نمایانگر سطوح درونیابی صحیح، نیم پیکسل و ربع پیکسل می‌باشند.

جدول ۴-۳ تعداد عملیات درون یابی با دقت نیم پیکسل، ربع پیکسل و تعداد دسترسی به حافظه برای تمامی ترکیبات بردار حرکت

$N^b_{Int-Mem}$	N^b_{Quad}	N^b_{Half}	MV_y	MV_x
M×N	0	0	I_y	I_x
M×(N+5)	0	M×N	H_y	I_x
M×(N+5)	M×N	M×N	Q_y	I_x
N×(M+5)	0	M×N	I_y	H_x
(N+5)×(M+5)	0	(2×N+5)×M	H_y	H_x
(N+5)×(M+5)	M×N	(2×N+5)×M	Q_y	H_x
N×(M+5)	M×N	M×N	I_y	Q_x
(N+5)×(M+5)	M×N	(2×M+5)×N	H_y	Q_x
M× (N+5)+5N	M×N	2×M×N	Q_y	Q_x

اجازه دهید که برای بردار حرکت افقی MV_x یک مجموعه $P_x = \{I_x, H_x, Q_x\}$ و برای بردار حرکت عمودی MV_y یک مجموعه $P_y = \{I_y, H_y, Q_y\}$ تعریف کنیم. بسته به دقت بردارهای حرکت MV_x و MV_y، اندیس متناظر در مجموعه P_x یا P_y برابر با یک و سایر داده‌ها صفر خواهند بود. به عنوان مثال، اگر بردار حرکت افقی MV_x دارای دقت نیم پیکسل و

بردار حرکت عمودی MV_y دارای ربع پیکسل باشد، مجموعه‌های مربوطه برابر خواهند بود با $P_x = \{0, 1, 0\}$ و $P_y = \{0, 0, 1\}$. با تعریف این دو مجموعه می‌توان پارامترهای پیچیدگی نشان داده شده در جدول ۴-۳ را بصورت پارامترهای محاسبه شده در معادلات (۴-۱۵) تا (۴-۱۷) به شکلی فشرده بیان کرد:

$$N_{Half}^{b} = \left(I'_x + I'_y + I'_x.I'_y\right) \times (M \times N) + \left(H_x.I'_y\right) \times (M \times N) + (Q_x.H_y) \times (N \times 5) \quad (۱۵-۴)$$

$$N_{Quad}^{b} = min\left[\left(Q_x + Q_y\right), 1\right] \times (M \times N) \quad (۱۶-۴)$$

$$N_{Int-Mem}^{b} = [M + 5 \times (H_x + Q_x)] \times \left[N + 5 \times \left(H_y + Q_y\right)\right] - 25 \times Q_x.Q_y \quad (۱۷-۴)$$

با تعریف روابط فوق می‌توان مجموعه‌های P_x و P_y را برای هر بلوک استخراج نمود و در نتیجه پارامترهای پیچیدگی مربوطه در سطح بلوک به سادگی قابل استخراج و محاسبه خواهند بود. همانطور که در جدول ۴-۴ نشان داده شده است، بر اساس مد ماکروبلوک، فرآیند MC ممکن است از یک تا ۱۶ بار برای هر ماکروبلوک اجرا گردد. با در نظر گرفتن مد ماکروبلوک در معادلات فوق، می‌توان متغیرهای M و N را با هرکدام از اندازه‌های مربوط به زیربلوک‌ها (یعنی ۴ و ۸ و ۱۶) به سادگی تعویض نمود.

جدول ۴-۴ تعداد عملیات جبران حرکت بر اساس مد ماکروبلوک

مد ماکروبلوک	مد زیر ماکروبلوک	تعداد عملیات جبران حرکت در سطح زیر بلوک	تعداد عملیات جبران حرکت در سطح ماکروبلوک
Skip، ۱۶×۱۶	۱۶×۱۶	۱	۱
۱۶×۸	۱۶×۸	۱	۲
۸×۱۶	۸×۱۶	۱	۲
P8×8	۸×۸	۱	بین ۴ تا ۱۶ (براساس مد زیر بلوک)
	۸×۴، ۴×۸	۲	
	۴×۴	۴	

با توجه به مد ماکروبلوک و مد زیربلوک، می‌توان پارامترهای پیچیدگی در سطح ماکروبلوک را با استفاده از روابط معادله (۴-۱۸) استخراج نمود. توجه به این نکته ضروری است که در این روابط مقدار پارامتر n که از یک تا ۱۶ متغیر است، به مد ماکروبلوک یا زیربلوک بستگی دارد.

$$\begin{cases} N_{Half}^{MB} = \sum_{i=1}^{n} N_{Half}^{b} \\ N_{Quad}^{MB} = \sum_{i=1}^{n} N_{Quad}^{b} \\ N_{Int-Mem}^{MB} = \sum_{i=1}^{n} N_{Mem}^{b} \end{cases} \quad (۱۸-۴)$$

همانطور که پیشتر اشاره شد، پارامتر پیچیدگی دیگری که برای این واحد داریم تعداد دسترسی به حافظه خارجی ($N_{Ext-Mem}^{MB}$) می‌باشد که بایستی مشابه با تعداد دسترسی به حافظه داخلی ($N_{Int-Mem}^{MB}$) نیز در نظر گرفته شده محاسبه گردد. لازم به ذکر است که ساختار حافظه بصورت آرایه یک بعدی می‌باشد اما در اینجا و برای تشریح آسانتر تعداد دسترسی‌ها به حافظه، آن را بصورت دو بعدی در نظر خواهیم گرفت. به این ترتیب طول و عرض حافظه داخلی را به ترتیب x و y فرض کرده‌ایم.

در ابتدا حافظه داخلی بایستی بر اساس آدرس بردار حرکت اولین ماکروبلوک پر شود. پس از آن، بر اساس اندازه حافظه داخلی و توزیع بردارهای حرکت، ممکن است اطلاعات مورد نیاز ماکروبلوکهای مجاور در حافظه داخلی موجود باشد، در غیر اینصورت یک cache miss اتفاق افتاده و اطلاعات مورد نظر بایستی از حافظه خارجی به درون حافظه داخلی منتقل گردد. تعداد دسترسیها به حافظه بستگی به ساختار حافظه داخلی و خارجی خواهد داشت. بر اساس این ساختار است که ممکن است تعدای پیکسل به دفعات از حافظه خارجی خوانده شود که به این محدوده پیکسلها، محدوده همپوشانی خواهیم گفت.

با در نظر گرفتن محدوده همپوشانی، تعداد دفعاتی که حافظه داخلی (در جهت افقی فریم مرجع) بایستی پر شود عبارتست از:

$$Number\ of\ horizontal\ cache\ fillings = \frac{Frm_x}{x - v} \quad (۱۹-۴)$$

در اینجا، v نشانگر عرض محدوده همپوشانی و Frm_x نیز عرض فریم را تعیین می‌کند. این عملیات بایستی برای تمامی ردیفهای ماکروبلوک در هر فریم تکرار گردد، به این ترتیب تعداد دفعات پر کردن حافظه داخلی (در جهت عمودی فریم مرجع) بصورت زیر خواهد بود.

$$Number\ of\ vertical\ cache\ fillings = \frac{Frm_x}{16} \quad (۲۰-۴)$$

نهایتا به ازای هر بار پر شدن حافظه داخلی، تعداد $x \times y$ پیکسل بایستی از حافظه خارجی خوانده شود. لذا تعداد کل دسترسیها به حافظه خارجی برای انجام عملیات تخمین حرکت برای N فریم مرجع (N_{Ref}) عبارتست از:

$$N_{Ext-Mem} = (\frac{Frm_x}{x - v} \times \frac{Frm_y}{16} \times x \times y) \times N_{Ref} \quad (۲۱-۴)$$

لازم به ذکر است که رابطه (۴-۲۱) تعداد دسترسی‌ها را برای کل فریم استخراج کرده است، از آنجا که روابط پیچیدگی ارائه شده در این فصل در سطح ماکروبلوک ارائه می‌شوند، لذا تعداد دسترسیها به حافظه خارجی در سطح ماکروبلوک بصورت زیر خواهد بود.

$$N^{MB}_{Ext-Mem} = (\frac{16 \times x \times y}{x - v}) \times N_{Ref} \quad (۲۲-۴)$$

حال با داشتن تمامی پارامترهای پیچیدگی در سطح ماکروبلوک یک رابطه خطی برای مدلسازی پیچیدگی واحد جبران حرکت (MC) ارائه می‌شود که بصورت بیان شده در معادله (۴-۲۳) خواهد بود:

$$C_{Motion\ Compensation} = W_{Half} \times N^{MB}_{Half} + W_{Quad} \times N^{MB}_{Quad} + W_{Int-Mem} \times N^{MB}_{Int-Mem} + W_{Ext-Mem} \times N^{MB}_{Ext-Mem} \quad (۲۳-۴)$$

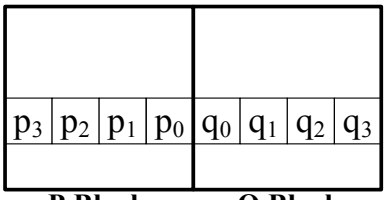
در معادله فوق، پارامترهای وزنی مربوط به پارامترهای پیچیدگی عبارتند از W_{Half}، W_{Quad}، $W_{Int-Mem}$ و $W_{Ext-Mem}$. همانطور که قبلا توضیح داده شد، این پارامترها با توجه به سکو طراحی کدگشا و نحوه پیاده‌سازی آن تنظیم می‌شوند.

۴-۲-۱-۱-۴ تخمین پیچیدگی فیلتر بلوک زدایی

به منظور کاهش آثار بلوکه شدن در لبه بلوک‌ها که نتیجه عملیات چندی سازی و تبدیل DCT است، در هر دو سمت کدکننده و کدگشای H.264/AVC، الگوریتم فیلتر درون حلقه‌ای اجرا می‌گردد. از آنجا که فیلتر بلوک زدایی در استاندارد H.264/AVC یک عملیات اختیاری می‌باشد، بنابراین ممکن است این عملیات در سمت کدکننده غیرفعال شود و در نتیجه،

در سمت کدگشا از این عملیات صرفنظر شود. با داشتن مدل محاسباتی واحد فیلتر بلوک زدایی (DBF) در سمت کدکننده، نه تنها می‌توان میزان مصرف توان در سطح ماکروبلوک را در طی فرآیند تصمیم گیری مد مدیریت کرد، بلکه می‌توان زمانی که توان کدگشا رو به پایان است، عملیات DBF را بطور کامل غیرفعال و حذف نمود.

همانند بخش‌های پیشین در اینجا نیز ابتدا الگوریتم DBF از منظر پیچیدگی تجزیه و تحلیل و پارامترهای پیچیدگی آن استخراج خواهد شد. سپس با توجه به پارامترهای استخراج شده، یک مدل پیچیدگی برای آن ارائه خواهد شد. عملیات فیلتر بلوک زدایی (DBF) بر روی لبه‌های تمام بلوک‌های ۴×۴ اعمال می‌گردد. برای هر بلوک ۴×۴، یک قدرت مرزی یا شدت مرزی BS تعریف شده و بر روی تمام لبه‌های آن بلوک اعمال می‌گردد. در شکل ۴-۵ دو بلوک مجاور P و Q و یک سطر از پیکسل‌های هرکدام نشان داده شده است در حالی که لبه عمودی در حال پردازش می‌باشد.



شکل ۴-۵ یک ردیف از دو بلوک ۴×۴ مجاور

برای مقادیر غیرصفر BS در صورتی که شرایط خاصی برقرار باشد، یک یا تعدادی از مقادیر موجود در مجموعه {p0, p1, p2, q0, q1, q2} فیلتر شده و اصلاح می‌گردند. زمانی که مقدار BS برابر با ۴ می‌باشد، یک فیلتر قوی بر روی لبه‌های بلوک اعمال می‌گردد و در غیر اینصورت برای مقادیر BS بین صفر و چهار (یعنی ۱و۲و۳) یک فیلتر نرمال یا معمولی بر روی پیکسل‌ها اعمال می‌گردد. بر اساس استاندارد H.264/AVC زمانی که مقدار BS برابر با ۱، ۲ یا ۳ است، فیلترهای یکسان بر روی تمام لبه‌های ماکروبلوک اعمال می‌گردد و تنها تفاوت این فیلترها در مقدار پارامتر آستانه می‌باشد. بنابراین، دو متغیر را برای بیان مقادیر BS تعریف می‌کنیم؛ متغیر BS123 که زمانی مقدار یک می‌گیرد که مقدار BS برابر با ۱، ۲ یا ۳ باشد و متغیر BS4 که زمانی مقدار یک می‌گیرد که قدرت مرزی برابر با چهار باشد. به منظور تعیین مقدار متغیرهای فوق، بعضی شرایط باید برآورده شوند. در اینجا، برای ساده سازی بیشتر فرمول بندی متغیرهای قدرت مرزی، تعدادی پارامتر تعریف می‌شوند که بصورت دودویی می‌باشند و در جدول ۴-۵ نشان داده شده‌اند. این پارامترها زمانی یک می‌شوند که شرایط مربوطه برقرار و برآورده شود.

جدول ۴-۵ شرایط مختلف و پارامترهای باینری معادل آنها که برای تعیین قدرت فیلتر استفاده می‌شوند

شرایط	پارامتر باینری
P|Q are intra	Intra
The edge of P and Q is a macroblock edge	MB_Edge
P or Q have non-zero coefficients	NZ_Res
$(MV_x^P - MV_x^Q \geq 1)$ && $(MV_y^P - MV_y^Q \geq 1)$	Large_MV
P and Q have different reference frames	Diff_Ref

در این جدول، P و Q دو بلوک همسایه افقی یا عمودی هستند و تمام شرایط جدول در سطح بلوک ارزیابی می‌شوند تا مقادیر BS برای دو بلوک همسایه مشخص شوند. با استفاده از پارامترهای جدول ۴-۵ هرکدام از متغیرهای قدرت مرزی با استفاده از روابط زیر تنظیم می‌شود:

$$BS_4 = \text{Intra} \,\&\&\, MB\_Edge \quad (۲۴-۴)$$

$$BS_{123} = !\,BS_4 \&\& \,(NZ\_Res \| Intra \,\| Large\_MV \| Diff\_Res) \quad (۲۵-۴)$$

$$BS_0 = !\,BS_{123} \&\& !\,BS_4 \quad (۲۶-۴)$$

همانطور که قبلا اشاره شد، برای بلوک‌هایی که مقدار BS آنها برابر با صفر است، یعنی (BS_0=1) عملیات فیلتر بلـوک زدایـی اعمال نخواهد شد. برای دو حالت دیگر، قبل از فیلتر کردن هر سطر شرایط زیر بررسی می‌شوند:

$$\begin{cases} C_1: |p_0 - q_0| < \alpha(Index_A) \\ C_2: |p_1 - p_0| < \beta(Index_B) \\ C_3: |q_1 - q_0| < \beta(Index_B) \end{cases} \quad (۲۷-۴)$$

در حالیکه:

$$\begin{cases} \alpha(x) = 0.8 \times (2^{\frac{x}{6}} - 1) \\ \beta(x) = 0.5 \times x - 7 \end{cases} \quad (۲۸-۴)$$

پارامترهای $Index_A$ و $Index_B$ با توجه به مقادیر QP تنظیم می‌گردند. سه شرط موجود در معادله (۴-۲۷) بـه منظـور تمـایز بین لبه‌های واقعی و لبه‌های حاصل از کدکردن بررسی می‌شوند. اگر تمام شرایط موجود در معادله (۴-۲۷) برآورده شوند، بـا توجه به مقادیر BS_{123} و BS_4 فیلترهای مناسبی بر روی سطری از پیکسل‌ها اعمال می‌گردنـد. هرچنـد کـه در هـر دو حالـت BS_{123} و BS_4 تعدادی شرایط دیگر نیز ارزیابی می‌شوند تا عملیات فیلتر کردن در گام دوم اجرا گردد، اما این مراحـل جزئـی در مدل پیچیدگی ما در نظر گرفته نخواهند شد.

بر اساس توضیح مختصر فوق، فرآیند DBF در چهار مرحله بخش بندی مـی‌شـود: پیـدا کـردن طبقـه بنـدی BS و خوانـدن ردیف‌های پیکسل از حافظه، ارزیابی شرایط C_1، C_2 و C_3، اجرای یک فیلتر قوی یـا فیلتـر معمـولی روی هـر سـطر. از بـین مراحل گفته شده، عملیات فیلتر کردن و دستیابی به حافظه بیشترین پیچیدگی محاسباتی را دارند. علاوه بر ایـن، از آنجـا کـه پیچیدگی فیلترهای معمولی و قوی متفاوت است، بنـابراین، بـرای هرکـدام از حـالات BS_{123} و BS_4 پارامترهـای پیچیـدگی جداگانه ای در نظر گرفته می‌شود. به عبارت دیگر، پارامترهای پیچیدگی بصورت تعـداد دسـتیابی‌هـای بـه حافظـه (N_{Mem}^{MB})، تعداد فیلترهای معمولی ($N_{F_{Normal}}^{MB}$) و تعداد فیلترهای قوی ($N_{F_{Strong}}^{MB}$) تعریـف مـی‌شـود. قبـل از توضـیح مـدل پیچیـدگی پیشنهادی، بهتر است بار دیگر، نگاهی به معادله (۴-۲۷) انداخته شود. همانطور که قبلا گفته شد، بـرای مقـادیر BS غیرصـفر، این شرایط قبل از اعمال عملیات فیلتر کردن بر روی هر سطر از پیکسل‌ها ارزیابی می‌شود. نویسـندگان مرجـع [64] بررسـی این شرایط را نادیده گرفته‌اند. به عبارت دیگر، فرض کرده اند که زمانی که مقدار BS مخالف صفر است، عملیات فیلتر بلوک زدایی بر روی تمام لبه‌ها اعمال می‌شود. بر اساس تحلیل‌ها و شبیه سازی‌های انجام شده در این رسـاله، ایـن فـرض درسـت نیست و می‌تواند به شدت بر مدل پیچیدگی تاثیر منفی بگذارد. در شبیه‌سازیهایی که انجام داده‌ایم، درصد لبه‌های فیلتر نشـده به خاطر برآورده نشدن شرایط در حالتی که مقدار BS غیرصفر است، استخراج شده است. این نتایج در ستون چهارم جـدول ۴-۶ برای تعدادی از ویدئوهای نمونه گزارش شده‌اند.

جدول ۴-۶ درصد مواقعی که C_1، C_2 و C_3 برقرار نمی‌شوند وقتی BS != 0

اندازه ویدئو	ویدئو نمونه	نرخ بیت (Kbps)	درصد لبه‌های فیلتر نشده، هنگامیکه BS !=0	لبه‌های فیلتر شده		
				تعداد لبه‌های واقعی	تعداد لبه‌های تخمین زده شده	درصد خطا (٪)
QCIF	Container	۴۶,۰۲	۳۶,۳	۶۷۹۰۴	۶۷۵۴۴	۰,۵۳
	Football	۷۸۰,۹۶	۷۶,۹۷	۱۸۹۷۱۵	۱۸۶۵۰۴	۱,۶۹
	Foreman	۶۶,۷۲	۳۲,۷۱	۱۱۹۳۹۶	۱۱۹۱۳۶	۰,۲۲
	Garden	۹۴۷,۰۵	۸۸,۸۶	۱۰۶۷۳۱	۱۰۹۴۰۰	۲,۵۰
	Mobile	۱۱۱۰,۷۶	۸۵,۹۷	۱۴۲۵۹۳	۱۴۵۳۸۰	۱,۹۵
	Stefan	۷۴۵,۰۳	۸۰,۱۲	۱۵۳۴۹۴	۱۵۳۵۷۶	۰,۰۵
CIF	Coastguard	۱۴۵۵,۷۱	۶۵,۰۶	۱۰۵۸۲۰۴	۱۰۳۰۴۶۴	۲,۶۲
	Container	۱۸۱,۵۹	۳۷,۰۵	۳۰۸۲۷۴	۳۰۷۲۲۴	۰,۳۴
	Foreman	۲۸۲,۷۹	۲۱,۵۷	۸۲۵۴۶۳	۸۲۴۹۱۲	۰,۰۷
	Mobile	۲۲۸۸,۱۱	۸۲,۲۲	۵۳۵۰۱۲	۵۳۶۴۳۲	۰,۲۷
	Paris	۶۱۹,۶۸	۵۶,۹۲	۴۴۱۰۷۳	۴۳۸۲۷۲	۰,۶۴
	Stefan	۱۳۸۱,۷۶	۷۰,۶۶	۶۸۸۰۴۹	۶۷۹۹۵۲	۱,۱۸
	Student	۱۹۳,۶۵	۲۷,۰۳	۳۸۴۲۳۷	۳۸۲۵۵۶	۰,۴۴

همانطور که در جدول مشاهده می‌شود، برای بیشتر ویدئوها تعداد قابل توجهی از لبه‌ها فیلتر نشده‌اند. بنابراین، برخلاف مرجع [64]، در مدل پیچیدگی پیشنهادی ما شرایط C_1، C_2 و C_3 کنترل خواهند شد. مشکل اصلی ارزیابی این شرایط در مدل پیچیدگی این است که مقداری پیچیدگی اضافی به کدکننده تحمیل می‌شود. به منظور پرهیز از این پیچیدگی، فرض می‌کنیم که اولین سطر از دو بلوک همسایه نماینده تمام سطرهای دیگر می‌باشد. به عبارت دیگر، اگر سطر اول بلوکها شرایط معادله (۴-۲۷) را برآورده کند، فرض می‌شود که تمام سطرهای دیگر نیز آنها را برآورده می‌کنند. دقت مدل پیشنهادی بعد از این ساده سازی با استفاده از شمردن تعداد لبه‌های فیلتر شده در شبیه‌سازی‌های ما محاسبه شده است که نتایج آن برای لبه‌های واقعی و مدل تخمینی پیشنهادی در ستون‌ها پنجم و ششم جدول ۴-۶ قابل مشاهده است. همانطور که مشاهده می‌شود، درصد خطای تخمین صورت گرفته بسیار ناچیز است (ستون آخر جدول ۴-۶) و این مشاهدات استفاده از این ساده سازی در مدل پیچیدگی پیشنهادی را تایید می‌کند. با استفاده از ساده سازی فوق و با فرض $C = C_1 \times C_2 \times C_3$ پارامترهای پیچیدگی در سطح بلوک به صورت زیر خواهند بود:

$$N^b_{F_{Normal}} = 4 \times BS_{123} \times C \quad (۴-۲۹)$$

$$N^b_{F_{Strong}} = 4 \times BS_4 \times C \quad (۴-۳۰)$$

$$N^b_{Mem} = 4 \times [6 \times (BS_{123} + BS_4) + (BS_{123} \times 3 + BS_4 \times 5) \times C] \quad (۴-۳۱)$$

با توجه به اینکه عملیات DBF در راستاهای عمودی و افقی برای تمام ۱۶ بلوک ۴×۴ درون هر ماکروبلوک اجرا می‌شود، پارامترهای پیچیدگی در سطح ماکروبلوک بصورت نشان داده شده در معادله (۴-۳۲) خواهند بود و سرانجام مدل خطی پیچیدگی پیشنهادی ما برای فیلتر بلوک زدایی بصورت نشان داده شده در معادله (۴-۳۳) خواهد بود.

$$\begin{cases} N_{F_{Normal}}^{MB} = \sum_{i=0}^{1}\sum_{j=1}^{16} N_{F_{Normal}}^{b} \\ N_{F_{Strong}}^{MB} = \sum_{i=0}^{1}\sum_{j=1}^{16} N_{F_{Strong}}^{b} \\ N_{Mem}^{MB} = \sum_{i=0}^{1}\sum_{j=1}^{16} N_{Mem}^{b} \end{cases} \quad (۳۲-۴)$$

$$C_{DBF} = W_{F_{Normal}} \times N_{F_{Normal}}^{MB} + W_{F_{Strong}} \times N_{F_{Strong}}^{MB} + W_{Mem} \times N_{Mem}^{MB} \quad (۳۳-۴)$$

به این ترتیب، پیچیدگی کدگشا با استفاده از روابط (۴-۱۲)، (۴-۱۴)، (۴-۲۳) و (۴-۳۳) قابل محاسبه خواهد بود. بعدا نشان خواهیم داد که این مدل پیچیدگی در طراحی کدکننده مطلع از محدودیت گیرنده به کار خواهد آمد.

۴-۲-۱-۲ آنالیز الگوریتم واحد تخمین حرکت در کدکننده H.264/AVC

در این بخش پیچیدگی واحد تخمین حرکت را تخمین خواهیم زد. با توجه به اینکه مدل همه منظوره پیچیدگی برای استاندارد H.264/AVC مدنظر است، لذا می‌توان زیر مجموعه‌ای از عملیات پایه که تاثیر بیشتری در پیچیدگی دارند را لحاظ نمود. به این منظور در مقاله [47] واحدهای مختلف استاندارد H.264/AVC بررسی شده است و تعداد هر یک از عملیات پایه بدست آمده است. جدول ۴-۷ تعداد این عملیات را به همراه درصد هر یک نشان می‌دهد.

جدول ۴-۷ تعداد عملیات پایه برای واحدهای مختلف و درصد کلی آنها [47]

واحد	زیر واحد	&	?	Store	Load	Shift	Branch	Mult.	Add
DCT4x4	Pre	–	–	–	۳	–	–	۲	۵
	DCT	–	–	–	۱۶	۱۶	–	–	۶۴
	IDCT	–	۳۲	۱۶	۱۶	۴۸	–	–	۱۰۰
	Quant.	–	۱۶	–	۳۲	۱۶	۱۶	۱۶	۳۲
	RLC	–	۳	۲	۱	۳	–	۲	۶
IntraPred	–	۳۷	–	۱۸۴	۲۲	۸۴	۱۸,۵	۳۳	۱۶۹
MC	–	–	–	۱۶	۴	–	–	–	–
Interpolation	Pre	–	–	–	۲۶۲	–	–	–	–
	1/2 - pel	–	۴	–	–	۸	–	۹	۱۹
	1/4 - pel	–	۲	–	–	۱	–	–	۲
	Post	–	–	۲۵۶	–	–	–	–	–
MVPred	–	۱۶	۹	۱۶	۸	۲۶	۱۹	۲۶	۲۴
Fast IME	Pre	–	۱	–	۲,۶	۲	۲	۳	۷
	ME	۶۴	۴۵۴	–	۲۸۸۰	۱۳۴	۲۹۴	۳۷۱	۴۱۰۶
	Post	–	۲	–	–	–	–	–	۲
Fast FME	–	۱۰	۲۲۲	۲	۲۹۲	۶۸,۵	۱۴	۲۱	۱۷۱۳
جمع	–	۹۰	۶۹۴	۲۷۴	۳۴۴,۶	۲۳۹,۵	۳۲۹	۴۳۰	۵۸۷۳
درصد	–	۰,۷۹	۶,۱۰	۲,۴۱	۳۰,۲۸	۲,۱۱	۲,۸۹	۳,۷۸	۵۱,۶۳

همانطور که در این جدول آمده است بیشترین عملیات پایه مربوط به جمع و دسترسی به حافظه می‌باشد. از آنجا که میزان مصرف توان و تعداد دسترسی به ثبات، حافظه داخلی و خارجی متفاوت است، برای دسترسی به حافظه سه پارامتر مستقل در نظر می‌گیریم. به این ترتیب می‌توان عملیات جمع، دسترسی به ثبات، دسترسی به حافظه داخلی و دسترسی به حافظه خارجی را بعنوان پارامترهای پیچیدگی در نظر گرفت و سپس پیچیدگی را بر اساس این پارامترها بدست آورد. به این ترتیب مدل پیچیدگی آمده در (۴-۱) بصورت زیر در خواهد آمد:

$$Complexity = N_{\mathrm{Sum}} \times W_{\mathrm{Sum}} + N_{Register} \times W_{Register} + N_{Int-Mem} \times W_{\mathrm{Int-Mem}} + N_{\mathrm{Ext-Mem}} \times W_{\mathrm{Ext-Mem}} \quad (۳۴-۴)$$

در این رابطه، N_{Sum}، $N_{Register}$، $N_{\mathrm{Int-Mem}}$ و $N_{\mathrm{Ext-Mem}}$ به ترتیب نمایانگر تعداد جمعها، دسترسی به ثبات، دسترسی به حافظه داخلی و دسترسی به حافظه خارجی می‌باشند. این عملیات پایه بر اساس الگوریتم قابل محاسبه می‌باشند. همچنین W_{Sum}، $W_{Register}$، $W_{\mathrm{Int-Mem}}$ و $W_{\mathrm{Ext-Mem}}$ وزن متناظر با هر یک از پارامترهای مذکور را نشان می‌دهند.

با استخراج پارامترهای پیچیدگی برای هر یک از روش‌های تخمین حرکت با جستجوی کامل و جستجوی سریع می‌توان پیچیدگی این روش‌ها را استخراج کرد. برای محاسبه هزینه عملیات تخمین حرکت، ابتدا هزینه روش تخمین حرکت صحیح (IME[1]) برای هر دو روش جستجوی کامل و جستجوی سریع را بدست می‌آوریم. سپس هزینه روش تخمین حرکت اعشاری (FME[2]) (دقت ½ پیکسل و دقت ¼ پیکسل) محاسبه می‌شود.

۴-۲-۱-۲-۱ تخمین حرکت صحیح

برای روش تخمین حرکت صحیح عملا دو روش معروف جستجوی کامل و جستجوی سریع معرفی شده است. برای پیاده‌سازیهای سخت افزاری عموما از روش جستجوی کامل استفاده می‌شود، چرا که این روش دارای ساختار دسترسی منظم تری به حافظه است. روش تخمین حرکت سریع، معمولا در پیاده‌سازیهای نرم افزاری به کار می‌رود، چرا که این روش سریعتر بوده، پیچیدگی محاسباتی کمتری دارد. برای اینکه مدلسازی پیشنهادی برای تمامی پیاده‌سازیهای ممکن قابل استفاده باشد، در این بخش پیچیدگی هر دو این روش‌ها را استخراج خواهیم کرد.

۴-۲-۱-۲-۱-۱ روش جستجوی کامل

اولین عملیات در تخمین حرکت عملیات محاسبه قدر مطلق تفاضل SAD[3] می‌باشد، اگر اندازه یک بلوک را n×n در نظر بگیریم، آنگاه به منظور محاسبه SAD برای آن نیاز به انجام n^2 عمل قدرمطلق و $n^2 - 1$ عمل جمع می‌باشد. با تبدیل هر عمل قدرمطلق به دو جمع تعداد عملیات جمع برای انجام عملیات SAD عبارتست از:

$$N_{\mathrm{Sum}}^{\mathrm{SAD}} \approx 3n^2 \quad (۳۵-۴)$$

[1] Integer Motion Estimation
[2] Fractional Motion Estimation
[3] Sum of Absolute Differences

در روش جستجوی کامل، عملیات SAD بایستی برای تمامی $(2SR+1)^2$ نقاط جستجو موجود در پنجره جستجو تکرار شود. لازم به ذکر است که SR معرف محدوده جستجو می‌باشد. به این ترتیب کل تعداد جمعهای لازم برای انجام عملیات تخمین حرکت صحیح عبارتست از:

$$N_{\text{Sum}}^{\text{Full ME-Integer}} = N_{\text{Sum}}^{\text{SAD}}(2SR+1)^2 NRef \quad (۳۶-۴)$$

همانطور که پیشتر اشاره شد، سه سطح برای دسترسی به حافظه تعریف کرده‌ایم که عبارتند از ثبات، حافظه داخلی و حافظه خارجی. برای اولین ماکروبلوک در هر ردیف، محدوده‌ای معادل $(16+2SR)^2$ بایستی از حافظه خارجی خوانده شود. برای سایر ماکروبلوکها، با توجه به همپوشانی پنجره جستجو بین ماکروبلوکهای متوالی، تعداد $16(16+2SR)$ پیکسل بایستی از حافظه خارجی خوانده شود. با در نظر گرفتن فریمی به ابعاد $(\text{Frm}_\text{x} \times \text{Frm}_\text{y})$ این عملیات بایستی به تعداد $\frac{\text{Frm}_\text{y}}{16}$ مرتبه تکرار شود، لذا تعداد کلی دسترسی به حافظه خارجی طبق رابطه (۴-۳۷) قابل تعریف است.

$$\text{Number of external memory accesses for the whole frame} = \frac{\text{Frm}_\text{y}}{16}\left[(16+2SR)^2 + 16(16+2SR)\left(\frac{\text{Frm}_\text{x}-16}{16}\right)\right] \quad (۳۷-۴)$$

در این رابطه، $\frac{\text{Frm}_\text{x}-16}{16}$ نشانگر تعداد کل ماکروبلوکهای واقع در هر ردیف بجز ماکروبلوک اول می‌باشد. از آنجایی که تمامی رابطه‌ها را در سطح ماکروبلوک تعریف می‌کنیم لذا تعداد کل دسترسیها به حافظه خارجی برای هر ماکروبلوک برای $NRef$ فریم مرجع طبق رابطه (۴-۳۸) بدست خواهد آمد.

$$N_{\text{Ext-Mem}}^{\text{Full ME-Integer}} = \frac{16}{\text{Frm}_\text{x}}\left[(16+2SR)^2 + 16(16+2SR)\left(\frac{\text{Frm}_\text{x}-16}{16}\right)\right] NRef \quad (۳۸-۴)$$

برای کد کردن هر ماکروبلوک، کل محدوده جستجو بایستی خوانده شده و به ثباتهای داخلی منتقل گردد. به طور کلی، در ابتدای هر ردیف از محدوده جستجو، کل بلوک بایستی از حافظه داخلی به ثباتها منتقل شود که به این ترتیب $n \times n$ دسترسی خواهیم داشت. برای دیگر نقاط جستجوی هر ردیف بایستی یک ستون n پیکسلی از حافظه داخلی خوانده شود. با در نظر گرفتن این واقعیت که عملیات بالا بایستی برای تمام ردیفهای پنجره جستجو (به عبارت دیگر $2SR$ مرتبه) تکرار شود، تعداد دسترسی به حافظه داخلی وقتی $NRef$ فریم مرجع داریم، عبارتست از:

$$N_{\text{Int-Mem}}^{\text{Full ME-Integer}} = (n^2+2nSR)(2SR)\,NRef \quad (۳۹-۴)$$

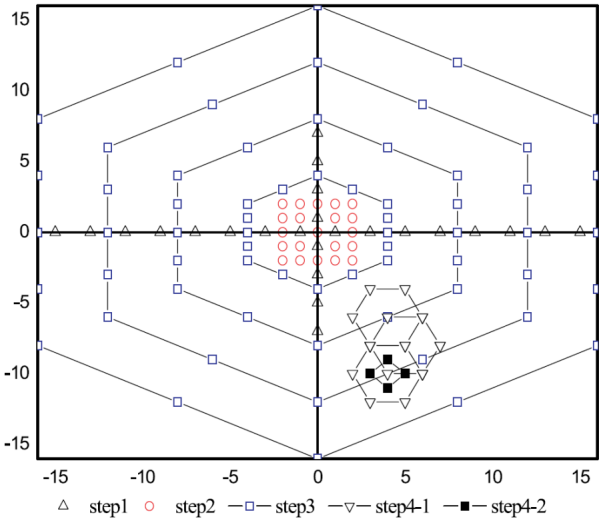
برای انجام عملیات تخمین حرکت، تمام پیکسلهای هر بلوک که در ثبات ذخیره شده‌اند، بایستی به واحدهای محاسباتی منتقل شوند، به این ترتیب $n \times n$ دسترسی به ثبات برای هر بلوک متصور خواهد بود. به این ترتیب کل ارجاعات به ثبات برای عملیات تخمین حرکت با جستجوی کامل و برای $NRef$ عبارت خواهد بود از:

$$N_{\text{Register}}^{\text{Full ME-Integer}} = (n \times n)\,NRef \quad (۴۰-۴)$$

۴-۲-۱-۲-۱-۲ روش جستجوی سریع

روش تخمین حرکت با جستجوی کامل، عملیات تخمین حرکت را با جستجوی کل محدوده پنجره جستجو انجام می‌دهد. اگرچه این روش منجر به یافتن بهترین بردار حرکت می‌گردد، اما دارای پیچیدگی بسیار زیادی می‌باشد. برای اجتناب از پیچیدگی بسیار زیاد این روش، روش‌های جستجوی سریع متنوعی پیشنهاد شده‌اند که هدف همه آنها کاهش میزان محاسبات ضمن رسیدن به کیفیتی نزدیک به روش جستجوی کامل می‌باشد.

با توجه به پیچیدگی بسیار کمتر روش جستجوی سریع، از این روش در بسیاری از پیاده‌سازیهای نرم افزاری استفاده شده است. در این بخش با تمرکز بر روی یکی از معروفترین روش‌های تخمین حرکت سریع، به نام UMHexagonS [13]، میزان پیچیدگی این دسته از روش‌های تخمین حرکت را مدل خواهیم کرد. عملیات جستجو برای روش UMHexagonS در شکل ۴-۶ نشان داده شده است.



شکل ۴-۶ عملیات جستجو در روش UMHexagonS، برای SR=۱۶

همانطور که در این شکل نشان داده شده است، الگوریتم UMHexagonS دارای چهار مرحله می‌باشد. در مرحله اول یک جستجوی غیرمتقارن در دو جهت افقی و عمودی صورت می‌گیرد. در این مرحله SR نقطه در جهت افقی و SR/2 نقطه در جهت عمودی جستجو خواهند شد. به این ترتیب در مجموع $(3/2 \times SR)$ نقطه در این مرحله ارزیابی خواهند شد.

در مرحله دوم، یک جستجوی کامل به شعاع کوچکی انجام می‌پذیرد. در این مرحله هنگامی که SR بزرگتر یا مساوی ۲ باشد، تعداد ۲۵ نقطه جستجو می‌شود. در حالیکه برای SR=۱ تعداد ۹ نقطه جستجو خواهد شد. به این ترتیب در مجموع برای مرحله دوم $min(25, (2 \times SR + 1)^2)$ نقطه پیمایش خواهد شد.

در مرحله سوم، با یک پترن ۶-ضلعی با شعاع‌های مختلف عملیات جستجو را ادامه می‌دهیم. شعاع این ۶-ضلعی از ۱ تا $SR/4$ تغییر می‌کند و هر پترن ۶-ضلعی شامل ۱۶ نقطه می‌باشد. به این ترتیب در مجموع در مرحله سوم، $16 \times (\frac{SR}{4})$ نقطه جستجو خواهد شد.

مرحله چهارم، شامل دو زیر مرحله است. زیر مرحله اول حاوی یک پترن ۶-ضلعی حاوی ۶ نقطه جستجو است که حول بهترین نقطه از مرحله ۳ قرار می‌گیرند. این زیر مرحله تا هنگامی ادامه پیدا می‌کند که بهترین نقطه جستجو، واقع در مرکز ۶-ضلعی باشد. در زیر مرحله دوم عملیات جستجو بصورت یک پترن لوزی ادامه می‌یابد. این مرحله از بهترین نقطه زیر مرحله اول شروع شده و تا هنگامی که بهترین نقطه جستجو در مرکز لوزی یافت نشود، ادامه می‌یابد. با توجه به اینکه تعداد نقاط جستجوی این مرحله برای ویدئوهای مختلف متفاوت خواهد بود، لذا به صورت میانگین تعداد ۲۱ نقطه جستجو را برای مرحله ۴ لحاظ کرده‌ایم.

همانطور که در رابطه (۴-۳۵) اشاره شد، برای هر نقطه جستجو $3n^2$ عملیات جمع نیاز است، به این ترتیب برای عملیات تخمین حرکت سریع تعداد عملیات جمع هنگامی که $NRef$ فریم مرجع و EnM مد فعال داشته باشیم عبارت خواهد بود از:

$$N_{\text{Sum}}^{\text{Fast ME-Integer}} = N_{\text{Sum}}^{\text{SAD}} \times \left[(3/2)SR + p_1\left(\text{Min}(25, (2\,SR+1)^2) + p_2(4\,SR + 21p_3)\right)\right] EnM \times NRef \quad (۴۱-۴)$$

لازم به ذکر است که پس از انجام هر مرحله از عملیات تخمین حرکت، یک مرحله پایان سریع[1] انجام می‌شود. برای در نظر گرفتن این مراحل در رابطه (۴-۴۱)، از پارامترهای p_1، p_2 و p_3 استفاده کرده‌ایم.

تعداد دسترسی به حافظه خارجی برای عملیات تخمین حرکت سریع و روش تخمین حرکت کامل یکسان خواهد بود. به این ترتیب رابطه (۴-۴۱) برای عملیات تخمین حرکت سریع نیز معتبر خواهد بود.

برای استخراج تعداد دسترسی به حافظه داخلی هر مرحله را جداگانه بررسی می‌کنیم. از آنجا که مراحل ۱ و ۲، ساختار منظمی دارند می‌توانیم مشابه با (۴-۳۹) از همپوشانی نقاط جستجوی همجوار استفاده کنیم، اما برای مراحل دیگر بایستی کل بلوک از حافظه داخلی خوانده شود. به این ترتیب تعداد عملیات دسترسی به حافظه داخلی هنگامی که $NRef$ فریم مرجع و EnM مد فعال داشته باشیم عبارت خواهد بود از:

$$N_{\text{Int-Mem}}^{\text{Fast ME-Integer}} = \left((2n^2 + (3/2)nSR) + p_1\left(5(n^2+4n) + p_2(4n^2SR + 21p_3n^2)\right)\right) EnM \times NRef \quad (۴۲-۴)$$

مشابه با روش جستجوی کامل، در روش جستجوی سریع نیز، برای هر بلوک محتویات کلیه ثبات‌ها بایستی به واحد محاسباتی منتقل شود. البته برای روش سریع این عملیات بایستی برای تمامی مدهای فعال (EnM) تکرار شود، بنابراین تعداد کل دسترسی به ثبات برای روش جستجوی سریع طبق رابطه (۴-۴۳) خواهد بود.

$$N_{\text{Register}}^{\text{Fast ME-Integer}} = (n \times n)\, EnM \times NRef \quad (۴۳-۴)$$

در ادامه این بخش، پارامترهای پیچیدگی برای تخمین حرکت اعشاری با دقت ½ و ¼ پیکسل، بصورت مستقل استخراج خواهند شد. بعدا پیچیدگی کلی برای هر یک از روش‌های تخمین حرکت با جستجوی کامل و تخمین حرکت با جستجوی سریع با در نظر گرفتن پیچیدگی مراحل تخمین حرکت صحیح و اعشاری ارائه خواهد شد.

۴-۲-۱-۲-۲ تخمین حرکت اعشاری با دقت ½ پیکسل

برای محاسبه بردار تخمین حرکت با دقت ½ پیکسل باید حول بهترین بلوک انتخاب شده در فریم مرجع توسط عملیات IME، درونیابی[2] انجام شود و ۸ بار دیگر SAD محاسبه شود و مکان با SAD کمینه انتخاب شود.

رابطه مربوط به درونیابی عبارتست از:

$$b = round\left(\frac{E - 5F + 20G + 20H + 5I + J}{32}\right) \quad (۴۴-۴)$$

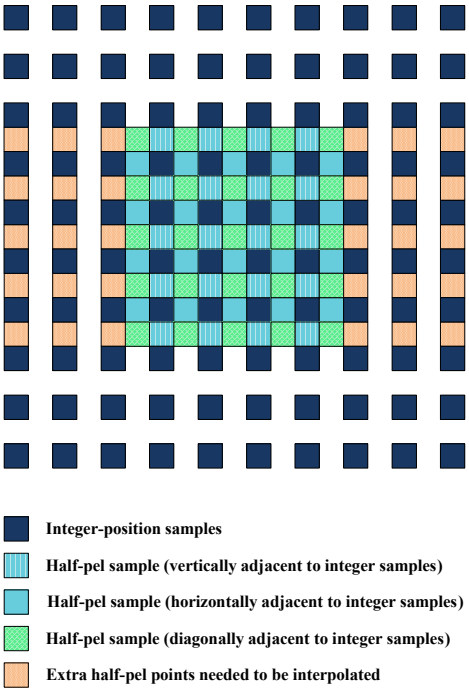
در رابطه (۴-۴۴)، متغیرهای E، F، G، H، I و J نشانگر مقدار پیکسلهای صحیح و متغیر b نشانگر پیکسل اعشاری حاصل می‌باشد. با تبدیل تمامی محاسبات به جمع، برای درون یابی هر نقطه جدید بایستی ۱۶ عمل جمع انجام داد.

$$N_{Sum}^{One-pel-half} = 16 \quad (۴۵-۴)$$

---

[1] Early termination
[2] Interpolation

برای یافتن تعداد محاسبات جمع و دسترسی به حافظه برای تخمین حرکت اعشاری با دقت ½ پیکسل، تعداد عملیات درون یابی مورد نیاز بایستی استخراج گردد. برای واضح شدن مسئله، عملیات درون یابی مربوطه در شکل ۴-۷ نشان داده شده است.



شکل ۴-۷ عملیات درون یابی با دقت ½ پیکسل برای یک بلوک ۴×۴

همانگونه که در شکل آمده است، نقاط با دقت ½ پیکسل به دو گروه قابل تقسیم بندی هستند. نقاطی که بین دو نقطه صحیح قرار گرفته‌اند و نقاطی که از روی نقاط با دقت ½ پیکسل تولید شده‌اند. پیکسلهای واقع در گروه اول شامل پیکسلهای افقی و عمودی می‌باشند که با استفاده از فیلتر ۶ مرحله‌ای تولید خواهند شد. پیکسلهای واقع در گروه دوم، با فیلتر کردن پیکسلهای افقی یا عمودی با دقت ½ پیکسل ایجاد می‌شوند. برای تولید پیکسلهای گروه دوم، تعدادی عملیات درون یابی اضافی نیاز است که در شکل نشان داده شده‌اند. با توجه به این دسته بندی و با توجه به شکل ۴-۷ تعداد فیلترهای ۶ مرحله‌ای برای یک بلوک $n \times n$ را می‌توان مانند آنچه در جدول ۴-۸ آمده است محاسبه کرد.

جدول ۴-۸ تعداد عملیات درون یابی با دقت ½ مورد نیاز برای یک بلوک $n \times n$

انواع پیکسلهای با دقت ½	تعداد فیلترهای ۶ مرحله‌ای مورد نیاز
پیکسلهای عمودی	$n \times (n+1)$
پیکسلهای افقی	$n \times (n+1)$
پیکسلهای قطری	$(n+1) \times (n+1)$
درون یابی اضافه	$2 \times (3 \times (n+1))$
جمع	$\mathbf{3n^2 + 10n + 7}$

با در نظر گرفتن تعداد کل پیکسلهایی که بایستی درون یابی شوند، تعداد عملیات جمع برای یک بلوک $n \times n$ عبارتست از:

$$N_{\text{Sum}}^{\text{Interpolation-half}} = (3n^2 + 10n + 7)N_{\text{Sum}}^{\text{One-pel-half}} \quad (۴۶-۴)$$

در نتیجه، تعداد کل عملیات جمع برای تخمین حرکت اعشاری با دقت ½ عبارتست از:

$$N_{\text{Sum}}^{\text{Half-ME}} = \left(8N_{\text{Sum}}^{\text{SAD}} + N_{\text{Sum}}^{\text{Interpolation-half}}\right)EnM \times NRef \quad (۴۷-۴)$$

با توجه به اینکه مقدار پارامتر EnM برای روش تخمین حرکت کامل، ثابت است لذا برای این پارامتر در رابطه (۴-۴۷) و بقیه روابط این بخش، مقدار ۷ را (تنها برای روش تخمین حرکت کامل) لحاظ می‌کنیم.

از آنجا که در مرحله تخمین حرکت صحیح، کلیه پیکسلهای لازم از حافظه خارجی خوانده شده است، لذا میزان دسترسی به حافظه خارجی در این مرحله برابر صفر ($N_{\text{Ext-Mem}}^{\text{Half-ME}} = 0$) در نظر گرفته می‌شود.

تعداد دسترسیهای به حافظه داخلی نیز از شکل ۴-۷ با در نظر گرفتن نقاط با دقت صحیح قابل استخراج است که برابر است با:

$$N_{\text{Int-Mem}}^{\text{Half-ME}} = (n^2 + 12n + 36)EnM \times NRef \quad (۴۸-۴)$$

تعداد دسترسی به ثبات برای هر بلوک برابر با $n \times n$ می‌باشد. به این ترتیب برای هر ماکروبلوک و برای EnM مد فعال و هنگامی که $NRef$ فریم مرجع داریم، تعداد کل دسترسی به ثباتها عبارت خواهد بود از:

$$N_{\text{Register}}^{\text{Half-ME}} = (n \times n)EnM \times NRef \quad (۴۹-۴)$$

۴-۲-۱-۲-۳ تخمین حرکت اعشاری با دقت ¼ پیکسل

در نهایت برای محاسبه بردار تخمین حرکت با دقت ¼ پیکسل باید حول بهترین بلوک انتخاب شده در فریم مرجع توسط عملیات تخمین حرکت با دقت ½ پیکسل، درونیابی انجام شود و ۸ بار دیگر SAD محاسبه شود و مکان با SAD کمینه انتخاب شود. برای درونیابی نقاط با دقت ¼ پیکسل از نقاط موجود قبلی میانگین گرفته می‌شود، پس هر درونیابی را می‌توان با یک عمل جمع مدل کرد. مشابه با تخمین حرکت با دقت ½، در اینجا نیز نقاط با دقت ¼ پیکسل را می‌توان به دو دسته تقسیم کرد. نقاطی که بصورت عمودی و افقی بین دو نقطه صحیح یا اعشاری با دقت ½ قرار دارند و همچنین نقاطی که بصورت قطری بین دو نقطه بدست آمده از مراحل قبل قرار دارند. با استفاده از این دسته بندی، می‌توان تعداد نقاط با دقت ¼ پیکسل را برای یک بلوک $n \times n$ طبق آنچه در جدول ۴-۹ آمده است محاسبه کرد.

جدول ۴-۹ تعداد عملیات درون یابی با دقت ¼ مورد نیاز برای یک بلوک $n \times n$

انواع پیکسلهای با دقت ¼	تعداد فیلترهای ۲ مرحله‌ای مورد نیاز
پیکسلهای عمودی	$2n \times (2n-1)$
پیکسلهای افقی	$2n \times (2n-1)$
پیکسلهای قطری	$(2n)^2$
جمع	$\mathbf{12n^2 - 4n}$

لازم به ذکر است که برای انجام عملیات تخمین حرکت اعشاری با دقت ¼ پیکسل نیاز به انجام $\frac{(6n+2)}{9}$ عملیات درون یابی با دقت ½ پیکسل می‌باشد به این ترتیب تعداد عملیات جمع برای درون یابی با دقت ¼ پیکسل عبارتست از:

$$N_{\text{Sum}}^{\text{Interpolation-quarter}} = (12n^2 - 4n) + \frac{(6n+2)}{9} N_{\text{Sum}}^{\text{One-pel-half}} \quad (۵۰-۴)$$

به این ترتیب کل عملیات جمع برای تخمین حرکت با دقت اعشاری بصورت زیر خواهد بود:

$$N_{\text{Sum}}^{\text{Quarter-ME}} = \left(8N_{\text{Sum}}^{\text{SAD}} + N_{\text{Sum}}^{\text{Interpolation-quarter}}\right) EnM \times NRef \quad (۵۱-۴)$$

نحوه دسترسی به حافظه در این بخش با مرحله تخمین حرکت با دقت ½ یکسان است. به عبارت دیگر از آنجاییکه تمام نقاط صحیح مورد نیاز در بخش تخمین حرکت صحیح حافظه خوانده شده‌اند، لذا میزان دسترسی به حافظه خارجی برای این مرحله صفر ($N_{\text{Ext-Mem}}^{\text{Quarter-ME}} = 0$) فرض خواهد شد. به ترتیب مشابه میزان دسترسی به حافظه داخلی عبارتست از:

$$N_{\text{Int-Mem}}^{\text{Quarter-ME}} = (n^2 + 12n + 36) EnM \times NRef \quad (۵۲-۴)$$

و نهایتا میزان دسترسی به ثباتها طبق رابطه (۴-۵۳) محاسبه خواهد شد.

$$N_{\text{Register}}^{\text{Quarter-ME}} = (n \times n) EnM \times NRef \quad (۵۳-۴)$$

همانطور که در روابط (۴-۵۳)–(۴-۳۶)، تعداد پارامترهای پایه عملیات تخمین حرکت تابعی از دقت بردار حرکت، محدوده جستجو و تعداد فریم‌های مرجع خواهد بود. با استفاده از متغییرهای باینری h و q می‌توان تمامی فرمولهای بالا را ادغام کرده و بصورت فرمولهای یکپارچه در آورد. در ادامه روابط (۴-۵۷)–(۴-۵۴) را برای روش تخمین حرکت با جستجوی کامل و روابط (۴-۶۱)–(۴-۵۸) را برای روش تخمین حرکت با جستجوی سریع تعریف خواهیم کرد.

$$N_{\text{Sum}}^{\text{Full ME}} \approx [3n^2(2SR+1)^2 + h(72n^2 + 160n + 112) EnM + q(36n^2 + 6n + 4) EnM] NRef \quad (۵۴-۴)$$

$$N_{\text{Ext-Mem}}^{\text{Full ME}} = \frac{16}{\text{Frm}_x}\left[(16+2SR)^2 + 16(16+2SR)\left(\frac{\text{Frm}_x - 16}{16}\right)\right] NRef \quad (۵۵-۴)$$

$$N_{\text{Int-Mem}}^{\text{Full ME}} = \left((n^2 + 2nSR)(2SR) + h(n^2 + 12n + 36) EnM + q(n^2 + 12n + 36) EnM\right) NRef \quad (۵۶-۴)$$

$$N_{\text{Register}}^{\text{Full ME}} = \left((n \times n) + h(n \times n) EnM + q(n \times n) EnM\right) NRef \quad (۵۷-۴)$$

$$N_{\text{Sum}}^{\text{Fast ME}} = \left(3n^2\left(\left(\frac{3}{2}\right)SR + p_1\left(\text{Min}(25, (2\,SR+1)^2) + p_2(4\,SR + 21p_3)\right)\right) + h(72n^2 + 160n + 112) + q(36n^2 + 6n + 4)\right) EnM \times NRef \quad (۵۸-۴)$$

$$N_{\text{Ext-Mem}}^{\text{Fast ME}} = \frac{16}{\text{Frm}_x}\left[(16+2SR)^2 + 16(16+2SR)\left(\frac{\text{Frm}_x - 16}{16}\right)\right] NRef \quad (۵۹-۴)$$

$$N_{\text{Int-Mem}}^{\text{Fast ME}} = \left(\left((2n^2 + (3/2)nSR) + p_1\left(5(n^2+4n) + p_2(4n^2 SR + 21p_3 n^2)\right)\right) + h(72n^2 + 160n + 112) + q(36n^2 + 6n + 4)\right) EnM \times NRef \quad (۶۰-۴)$$

$$N_{\text{Register}}^{\text{Fast ME}} = \left((n \times n) + h(n \times n) + q(n \times n)\right) EnM \times NRef \quad (۶۱-۴)$$

در فرمول بالا با قراردادن h = q = 0 می‌توان روابط مربوط به تخمین حرکت صحیح را بدست آورد، برای تخمین حرکت با دقت ½ پیکسل بایستی h = 1 و q = 0 قرار داده شود و بالاخره برای تخمین حرکت با دقت ¼ پیکسل h = q = 1 خواهد بود.

با داشتن N_{Sum}، N_{Register}، $N_{\text{Int-Mem}}$ و $N_{\text{Ext-Mem}}$ و پارامترهای وزن برای هر یک از روش‌های تخمین حرکت، پیچیدگی این واحد با استفاده از رابطه (۴-۳۴) و بر اساس تعداد سیکل ساعت قابل محاسبه خواهد بود.

۴-۲-۱-۲-۴الگوریتم نهایی برای استخراج پیچیدگی واحد تخمین حرکت

در این بخش تمامی مباحث قبلی را جمع بندی کرده و الگوریتمی برای استخراج پیچیدگی ارائه می‌کنیم. از آنجا که برای هر سکو جدیدی بایستی در ابتدا عملیات تنظیم پارامترهای وزن صورت گیرد، لذا این مرحله نیز در الگوریتم پیشنهادی ارائه کرده‌ایم. تنها یادآوری می‌کنیم که برای تنظیم پارامترهای وزن (شامل W_{Sum}، W_{Register}، $W_{\mathrm{Int-Mem}}$ و $W_{\mathrm{Ext-Mem}}$) برای هر پیاده‌سازی و سکو، یکسری شبیه‌سازی اولیه روی تعدادی ویدئو نمونه صورت خواهد گرفت. در حین کد کردن هر ویدئو، تعداد عملیات پایه (N_{Sum}، N_{Register}، $N_{\mathrm{Int-Mem}}$ و $N_{\mathrm{Ext-Mem}}$) برای هر ترکیبی از پارامترهای کدکردن استخراج می‌شود. همزمان، تعداد پالسهای ساعت مصرف شده برای کدکردن نیز استخراج می‌شود. با داشتن تعداد عملیات پایه و پیچیدگی معادل آنها که با تعداد پالس ساعت مشخص می‌شود، می‌توان وزن هر پارامتر را استخراج کرد. برای هر ویدئوی دیگری در این سکو و شبیه‌سازی، می‌توان از همین وزنها استفاده کرد.

الگوریتم پیشنهادی برای استخراج پیچدگی روش‌های تخمین حرکت صحیح و تخمین حرکت کامل در لیست ۴-۱ آمده است.

۱. اگر برای اولین بار است که عملیات استخراج پیچیدگی بر روی سکو حاضر اجرا می‌شود به مرحله ۲ برو، در غیر اینصورت به مرحله ۵ برو.
۲. برای زیر مجموعه‌ای از ترکیبات {NRef, SR, Res, EnM} تعدادی شبیه‌سازی انجام بده و تعداد پالس ساعت لازم برای اجرای عملیات تخمین حرکت را استخراج کن.
۳. مقدار پارامترهای N_{Sum}، N_{Register}، $N_{\mathrm{Int-Mem}}$ و $N_{\mathrm{Ext-Mem}}$ را برای هر یک از ترکیبات {NRef, SR, Res, EnM} استخراج کن. برای روش تخمین حرکت با جستجوی کامل از روابط (۴-۵۷)-(۴-۵۴) و برای روش تخمین حرکت با جستجوی سریع از روابط (۴-۶۱)-(۴-۵۸) استفاده کن.
۴. با استفاده از روش Least Square پارامترهای وزن (W_{Sum}، W_{Register}، $W_{\mathrm{Int-Mem}}$ و $W_{\mathrm{Ext-Mem}}$) را استخراج کن. پایان الگوریتم.
۵. با توجه به مقادیر NRef، SR، Res و EnM مقدار پارامترهای N_{Sum}، N_{Register}، $N_{\mathrm{Int-Mem}}$ و $N_{\mathrm{Ext-Mem}}$ را، با استفاده از روابط (۴-۵۷)-(۴-۵۴) برای روش تخمین حرکت با جستجوی کامل و (۴-۶۱)-(۴-۵۸) برای روش تخمین حرکت با جستجوی سریع، استخراج کن.
۶. با توجه به مقادیر پارامترهای وزن و با استفاده از رابطه (۴-۳۴)، پیچیدگی روش تخمین حرکت را بر اساس تعداد پالس ساعت گزارش کن. پایان الگوریتم.

لیست ۴-۱ الگوریتم استخراج پیچیدگی برای واحد تخمین حرکت

۴-۲-۲ ارزیابی مدل تخمین پیچیدگی پیشنهادی

در این بخش دو روش تخمین پیشنهادی را در بخشهای مجزا بر روی سکوها و پیاده سازیهای مختلف ارزیابی خواهیم کرد.

۴-۲-۲-۱ ارزیابی مدل پیچیدگی پیشنهادی برای کدگشا H.264/AVC

در این بخش، نرم افزار JM 16.2 [67] و همچنین X264 [68] به عنوان گزینه‌های پیاده‌سازی H.264/AVC انتخاب شده‌اند. علاوه بر این، به منظور نشان دادن دقت مدل برای سکوهای سخت افزاری، واحد جبران حرکت با استفاده از زبان توصیف سخت افزاری VHDL به صورت یک سخت افزار اختصاص یافته پیاده‌سازی شده است. این واحد سخت افزاری کاملا مطابق با استاندارد H.264/AVC بوده و قابلیت کار روی یک Virtex4 FPGA مدل XC4VLX200 با فرکانس کاری 75 مگاهرتز را دارد. برای سکوهای مختلف، از یک سیستم PC با پردازنده چهار هسته‌ای اینتل با ساعت ۳ گیگاهرتزی، ۸ گیگابایت حافظه و همچنین یک سیستم نت بوک با پردازنده Atom با ساعت ۱,۶۶ گیگاهرتزی و ۱ گیگابایت حافظه استفاده کرده‌ایم. جدول ۴-۱۰ تمام ترکیبات شبیه‌سازی مورد استفاده برای هر واحد را نشان می‌دهد.

جدول ۴-۱۰ انواع ارزیابیها برای پیاده‌سازی و سکوهای مختلف

پیاده‌سازی	سکو	واحد			
		DBF	**MC**	**IDCT**	**ED**
JM 16.2	Intel	✓	✓	✓	✓
	Atom	✓	✓	✓	✓
X264 Decoder (FFMPEG)	Intel	✓	✓	✓	✓
	Atom	✓	✓	✓	✓
Dedicated Hardware		×	✓	×	×

برای انجام شبیه‌سازی روی سکوها و پیاده‌سازیهای مذکور، ویدئوهای نمونه با اندازه‌های مختلفی از اندازه QCIF تا HD استفاده شده است. عموما در وضوح پیکسل بالاتر، کیفیت بیشتری مد نظر بیننده می‌باشد لذا برای هر دسته از اندازه ویدئوها یک PSNR جداگانه تعریف کرده‌ایم. این دسته بندی‌ها در جدول ۴-۱۱ آمده است.

جدول ۴-۱۱ کیفیت مد نظر برای هر اندازه ویدئو

اندازه ویدئو	کیفیت مد نظر PSNR (dB)
QCIF (176×144)	۳۲
CIF (352×288)	۳۴
4CIF (704×576)	۳۸
HD (1280×720)	۴۰
HD (1920×1080)	۴۲

برای هر ویدئو نمونه نرخ بیتی که منجر به کیفیت آمده در جدول ۴-۱۱ می‌شود را استخراج کرده و از آن نرخ بیت در شبیه‌سازیها استفاده خواهیم کرد. تنظیمات کلی برخی پارامترهای کدکننده نیز طبق جدول ۴-۱۲ صورت گرفته است.

جدول ۴-۱۲ تنظیمات پارامترهای کدکننده

Profile	Baseline
Level	۳
Number of Coded Frames	۱۰۰
Intra Period	۱۵
Number of Reference Frames	۲
Search Range	۳۲
Rate Distortion Optimization	روشن
Rate Control	روشن
ME Precision	۱/۴ پیکسل

در مرحله تنظیم پارامترهای وزن، دو ویدئو نمونه برای هر اندازه ویدئو استفاده شده است. هر یک از ویدئوها با ۶ پارامتر چندی سازی متفاوت (بین ۲۴ و ۳۴ و با پله ۲) کد شده است. تمامی رشته بیتهای تولید شده، کدگشایی شده و تعداد پالس ساعت واقعی مورد نیاز برای کدگشایی با استفاده از نرم افزار VTune [69] استخراج شده است. سپس با توجه به موجود بودن تعداد پارامترهای پیچیدگی و زمان واقعی کدگشایی، مقدار پارامترهای وزن استخراج شده است. لازم به ذکر است که عملیات تنظیم وزن برای هر سکو و پیاده‌سازی تنها یک بار و بصورت offline انجام می‌شود. برای هر دسته از اندازه ویدئوها، مدت زمان مورد نیاز برای انجام عملیات تنظیم پارامترهای وزن تنها برابر با کدکردن و کدگشایی ۱۲ ویدئو نمونه می‌باشد. لازم به ذکر است که هیچ یک از ویدئوهای استفاده شده در مرحله تنظیم پارامترها، در مرحله ارزیابی مورد استفاده قرار نگرفته است.

پس از استخراج پارامترهای وزن، ویدئوهای نمونه مختلفی کدشده و مقدار هر یک از پارامترهای پیچیدگی استخراج خواهد شد. نهایتا با استفاده از روابط (۴-۱۲)، (۴-۱۴)، (۴-۲۳) و (۴-۳۳) بترتیب پیچیدگی واحدهای ED، IDCT، MC و DBF تخمین زده می‌شود. پیچیدگی واقعی هر واحد نیز با کدگشایی هر رشته بیت و شمارش تعداد پالسهای ساعت با استفاده از نرم افزار VTune بدست خواهد آمد.

جدول ۴-۱۳ نرخ خطای مدل پیچیدگی پیشنهادی برای پیاده‌سازی JM 16.2 بر روی سکو PC را نشان می‌دهد، ضمن اینکه جدول ۴-۱۴ این نرخ را برای پیاده‌سازی x264 روی سکو PC ارائه می‌کند. برای هر واحد، پیچیدگی کدگشایی مدل شده به همراه تفاوت آن با مقدار پیچیدگی واقعی در جدولهای مذکور گزارش شده است. برای هر اندازه ویدئو و با در نظر گرفتن قدر مطلق خطای تخمین، میزان خطای تخمین میانگین نیز در این جداول گزارش شده است. آخرین ستون این جدولها، مجموع پیچیدگی مدل شده برای تمامی ۴ واحد به همراه خطای تخمین کلی را نشان می‌دهد.

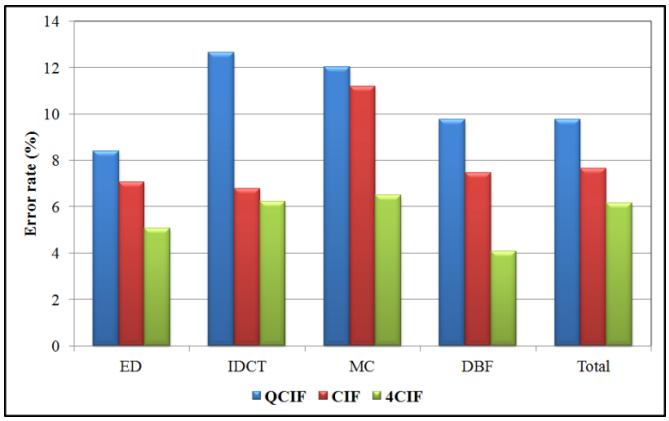
شبیه‌سازیهای مشابهی نیز برای سکو Atom انجام شده است و نتایج مشابهی بدست آمده است. نرخ متوسط خطا برای سکو Atom و پیاده‌سازی JM16.2 در شکل ۴-۸ نشان داده شده است.

جدول ۴-۱۳ نرخ خطای مدل پیچیدگی پیشنهادی برای پیاده‌سازی JM 16.2 بر روی سکو PC

ویدئو نمونه		نرخ بیت (Kbps)	ED (1000 Clks)		IDCT (1000 Clks)		MC (1000 Clks)		DBF (1000 Clks)		Total (1000 Clks)	
			مدل شده	خطای تخمین	مدل شده	خطای تخمین	مدل شده	خطای تخمین	مدل شده	خطای تخمین	مدل شده	خطای تخمین
QCIF	Football	۸۵۰	۲۱۸۳۴۳	۱۱۹۶۴	۵۳۸۳۸	-۳۰۰۵	۹۳۵۴۵	-۱۱۹۶۴	۵۶۸۲۹	-۵۹۵۴	۴۲۲۵۵۵	۳۲۸۸۷
	Garden	۹۵۲	۲۲۴۳۲۵	۸۹۷۳	۵۹۸۲۰	۲۷۸۹	۹۱۲۵۱	-۸۹۸۴	۵۹۸۲۰	۳۲۰۲	۴۳۵۲۱۶	۲۳۹۴۸
	Stefan	۹۰۰	۲۱۵۳۵۲	۱۱۸۷۵	۵۳۸۳۸	-۳۱۰۳	۸۸۹۴۱	-۲۴۷۸	۵۶۸۲۹	۴۰۱۲	۴۱۴۹۶۰	۲۱۴۶۸
	میزان خطای میانگین		٪۴,۹۹		٪۵,۳۳		٪۸,۴۷		٪۷,۶۳		٪۶,۱۵	
CIF	Flower	۱۷۰۰	۴۳۶۶۸۶	-۷۷۷۶۶	۱۸۲۴۵۱	۲۸۷۰	۲۸۱۱۵۴	-۳۲۹۰۱	۲۰۹۳۷۰	۱۷۹۴۶	۱۱۰۹۶۶۱	۱۳۱۴۸۳
	Mobile	۲۳۰۰	۵۴۱۳۷۱	۲۸۷۴	۲۷۲۱۸۱	۱۱۹۶۴	۴۶۸۶۰۵	۱۴۹۵۵	۲۵۱۲۴۴	-۳۱۵۴	۱۵۳۳۴۰۱	۳۲۹۴۷
	Stefan	۱۶۰۰	۴۶۰۶۱۴	-۵۳۸۳۸	۲۰۰۳۹۷	۵۹۸۸	۳۶۹۸۵۱	۱۱۹۶۴	۲۳۶۲۸۹	۲۸۷۴	۱۲۶۷۱۵۱	۷۴۶۶۴
	میزان خطای میانگین		٪۱۰,۰۱		٪۲,۹۹		٪۶,۰۴		٪۳,۶۸		٪۶,۶۳	
4CIF	City	۵۷۰۰	۲۰۷۲۷۶۳	-۸۶۷۳۹	۹۳۹۱۷۴	۶۱۴۰	۱۸۴۱۵۷۸	۱۵۵۵۳۲	۱۱۸۱۴۴۵	-۲۷۲۱	۶۰۳۴۹۶۰	۲۵۱۱۳۲
	Crew	۳۲۰۰	۱۴۲۶۷۰۷	-۱۱۶۶۴۹	۶۸۱۹۴۸	-۱۱۴۳۲۵	۱۴۲۹۶۹۸	۱۴۰۵۷۷	۱۵۰۷۴۶۴	۱۹۷۴۰۶	۵۰۴۵۸۱۷	۵۶۸۹۵۷
	Harbour	۷۰۰۰	۲۷۴۸۷۲۹	۴۱۸۷۴	۱۰۹۷۶۹۷	۳۰۱۴۵	۱۹۷۴۰۶۰	۱۵۲۵۴۱	۱۴۸۰۵۴۵	۳۸۸۸۳	۷۳۰۱۰۳۱	۲۶۳۴۴۳
	میزان خطای میانگین		٪۴,۶۳		٪۶,۷۲		٪۸,۶۷		٪۵,۳۲		٪۶,۳۵	
1280x720	In to three	۲۲۰۰۰	۷۸۳۹۴۱۱	۱۲۲۶۳۱	۲۷۰۳۸۶۴	-۱۳۱۶۰۴	۳۴۲۴۶۹۵	۳۰۸۰۷۳	۲۹۹۱۰۰۰	-۴۴۲۶۶۸	۱۶۹۵۸۹۷۰	۲۵۱۱۳۲
	Mobcal	۳۵۰۰۰	۱۱۲۵۲۱۴۲	۲۶۰۲۱۷	۲۸۵۳۴۱۴	-۵۲۳۴۲۵	۳۷۶۵۶۶۹	۱۹۴۴۱۵	۲۷۹۰۶۰۳	-۴۶۰۶۱۴	۲۰۶۶۱۸۲۸	۵۶۸۹۵۷
	Old town cross	۲۲۰۰۰	۷۸۶۶۳۳۰	۱۲۸۶۱۳	۲۷۵۴۷۱۱	-۵۶۸۲۹	۳۸۰۱۵۶۱	۴۱۸۷۴۰	۳۶۱۹۱۱۰	-۱۱۹۶۴	۱۸۰۴۱۷۱۲	۲۶۳۴۴۳
	Park joy	۴۵۰۰۰	۱۰۹۵۳۰۴۲	-۴۰۶۷۷۶	۲۵۲۴۴۰۴	-۲۸۴۱۴۵	۴۱۱۵۶۱۶	۱۳۴۵۹۵	۲۱۸۳۴۳۰	-۴۶۰۶۱۴	۱۹۷۷۶۴۹۲	۲۵۱۱۳۲
	میزان خطای میانگین		٪۲,۳۱		٪۹,۱۳		٪۷,۱۱		٪۱۳,۱۸		٪۵,۶۳	

جدول ۴-۱۴ نرخ خطای مدل پیچیدگی پیشنهادی برای پیاده‌سازی x264 بر روی سکو PC

ویدئو نمونه		نرخ بیت (Kbps)	ED (1000 Clks)		IDCT (1000 Clks)		MC (1000 Clks)		DBF (1000 Clks)		Total (1000 Clks)	
			مدل شده	خطای تخمین	مدل شده	خطای تخمین	مدل شده	خطای تخمین	مدل شده	خطای تخمین	مدل شده	خطای تخمین
QCIF	Football	۸۵۰	۵۶۸۲۹	۱۵۴۵	۱۲۴۵۸	-۷۴۸	۲۶۷۸۵	-۳۱۷۸	۲۸۴۱۵	۲۵۱۸	۱۲۴۴۸۷	۷۹۸۹
	Garden	۹۵۲	۵۹۸۲۰	۲۷۳۹	۱۳۹۶۴	۲۳۰	۲۵۴۲۴	-۲۸۲۶	۲۶۵۹۰	-۳۲۴۲	۱۲۵۷۹۸	۹۰۳۷
	Stefan	۹۰۰	۵۳۸۳۸	-۲۸۱۶	۱۱۴۹۶	-۷۱۸	۲۳۶۲۹	-۱۴۷۸	۳۰۶۵۸	۲۲۴۴	۱۱۹۶۲۱	۷۲۵۶
	میزان خطای میانگین		٪۴,۱۸		٪۴,۶۳		٪۹,۷۵		٪۹,۴۶		٪۶,۵۶	
CIF	Flower	۱۷۰۰	۱۳۷۵۸۶	-۱۳۴۶۰	۳۵۵۰۳	-۲۵۹۷	۸۲۴۳۲	-۶۸۶۹	۹۴۵۴۶	-۴۶۲۵	۳۵۰۰۶۷	۲۷۵۵۱
	Mobile	۲۳۰۰	۱۷۰۴۸۷	-۱۹۴۴۲	۵۰۸۴۷	۲۶۰۲	۱۲۱۵۸۴	۱۵۸۱	۱۱۲۰۴۳	-۶۴۳۴	۴۵۴۹۶۱	۳۰۰۵۹
	Stefan	۱۶۰۰	۱۴۰۵۷۷	-۶۲۸۱	۳۸۵۸۴	-۳۳۷۳	۱۰۰۴۹۸	-۴۸۵۲	۱۱۴۲۱۴	۱۱۹۲۰	۳۹۳۸۷۳	۲۶۳۹۹
	میزان خطای میانگین		٪۸,۵۵		٪۷,۰۶		٪۴,۸۱		٪۷,۰۲		٪۷,۰۶	
4CIF	City	۵۷۰۰	۵۶۲۳۰۸	-۲۹۹۱۰	۱۸۶۷۸۸	۳۱۵۴	۴۲۱۷۳۱	۴۹۴	۴۱۴۲۸۳	-۳۰۰۱۶	۱۵۸۵۱۱۰	۶۳۵۷۴
	Crew	۳۲۰۰	۴۰۹۷۶۷	-۳۲۹۰۱	۱۳۶۲۱۰	-۴۷۸۵۶	۳۲۶۰۱۹	-۳۶۳۰	۴۹۵۴۲۹	-۱۲۱۵۵	۱۳۶۷۴۲۵	۹۶۵۴۲
	Harbour	۷۰۰۰	۷۳۵۷۸۶	۵۰۸۴۷	۲۱۸۳۴۳	۱۹۵۶	۴۶۰۶۱۴	-۷۸۹۶	۵۱۵۳۸۵	-۱۵۰۹۳	۱۹۳۰۱۲۸	۷۵۷۹۲
	میزان خطای میانگین		٪۶,۷۵		٪۱۲,۵۷		٪۰,۹۸		٪۴,۲۱		٪۵,۰۰	
1280x720	In to three	۲۲۰۰۰	۲۰۳۹۸۶۲	۲۱۸۳۴۳	۵۸۶۲۳۶	-۸۸۶۹۵	۹۲۲۷۲۴	-۷۰۷۰۷	۱۱۷۸۴۵۴	-۹۵۸۶۲	۴۷۲۷۲۷۶	۴۷۳۶۰۷
	Mobcal	۳۵۰۰۰	۲۸۴۱۴۵۰	۳۷۳۸۷۵	۶۱۹۱۳۷	-۶۸۸۷۲	۱۰۴۴۰۰۹	-۵۵۷۸۵	۱۲۶۲۲۰۲	-۱۶۹۰۵	۵۷۶۶۷۹۸	۵۱۵۴۳۷
	Old town cross	۲۲۰۰۰	۲۰۴۲۸۵۳	۲۳۰۳۰۷	۵۹۸۲۰۰	-۳۱۸۶۶	۱۰۰۳۹۴۱	-۷۷۹۱۶	۱۳۳۰۹۹۵	-۱۸۱۴۲	۴۹۷۵۹۸۹	۳۵۸۲۳۱
	Park joy	۴۵۰۰۰	۲۶۱۱۱۴۳	-۱۱۹۶۴۰	۵۴۷۳۵۳	-۸۴۶۶۵	۱۱۵۳۴۴۹	-۶۳۰۰۵	۱۰۱۶۹۴۰	-۱۸۴۰۶۶	۵۳۲۸۸۸۵	۴۵۱۳۷۶
	میزان خطای میانگین		٪۹,۹۳		٪۱۱,۷۶		٪۶,۵۶		٪۷,۲۳		٪۸,۲۰	



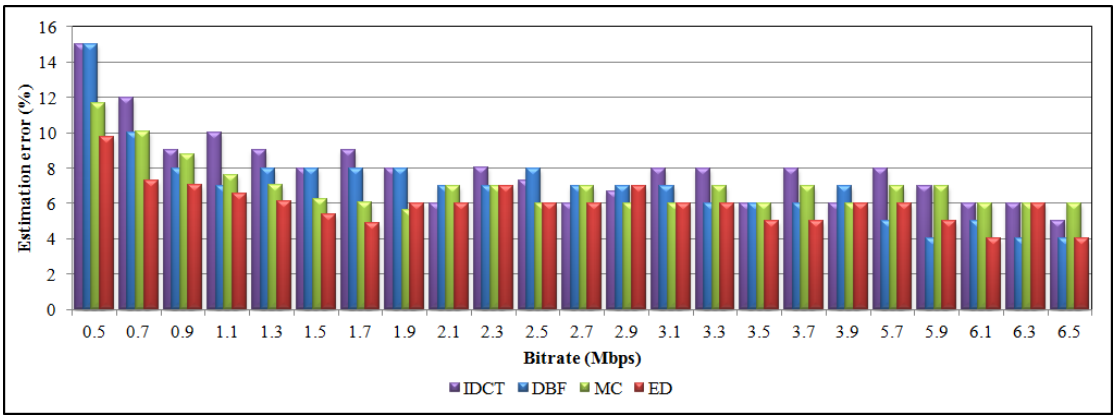
شکل ۴-۸ نرخ متوسط خطا برای مدل پیشنهادی پیچیدگی برای پیاده‌سازی JM 16.2 و روی سکو Atom

دقت مدل پیشنهادی برای پیاده‌سازی سخت افزاری واحد جبران حرکت را نیز در جدول ۴-۱۵ آورده‌ایم. در این جدول پیچیدگی این واحد بر اساس تعداد پالسهای ساعت تخمین زده شده و با تعداد واقعی پالسهای ساعت مقایسه شده است. همانطور که در جدول نشان داده شده است، به طور میانگین مدل پیشنهادی دارای خطای میانگینی برابر با ۱۰,۳۹٪ می‌باشد.

جدول ۴-۱۵ دقت مدل پیچیدگی پیشنهادی برای پیاده‌سازی سخت افزاری برای واحد جبران حرکت

ویدئو نمونه		نرخ بیت (Kbps)	ED (1000 Clks)	
			مدل شده	خطای تخمین
QCIF	Container	۶۷	۵۵۱۵۴۴	-۶۱۹۸۰
		۸۳	۵۲۱۱۰۷	-۵۳۵۵۲
	Football	۷۴۵	۱۴۷۴۷۰۵	۱۹۴۵۵۳
		۸۵۵	۱۴۳۳۲۰۴	۲۰۲۴۰۹
	Foreman	۱۰۹	۱۱۰۱۹۵۳	۱۹۴۲۱۰
		۱۲۵	۱۰۸۷۶۱۹	۱۹۸۴۴۱
	Garden	۸۵۰	۱۳۱۰۱۱۳	۵۵۶۶۹
		۹۷۸	۱۲۸۲۹۷۳	۶۱۵۹۱
	Stefan	۷۷۶	۱۲۴۰۷۱۰	۱۸۰۶۶
		۹۶۴	۱۲۰۰۶۳۲	۳۵۹۹۰
میزان خطای میانگین			٪۱۰,۳۹	

با توجه به اینکه در نرخ بیتهای پائینتر توزیع اطلاعات سرآیند و ضرایب نسبت به نرخ بیتهای بالا متفاوت است، شبیه‌سازی دیگری برای بررسی عملکرد مدل پیشنهادی در محدوده وسیعی از نرخ بیتها انجام داده‌ایم. در این شبیه‌سازیها ۴ ویدئو با اندازه 4CIF انتخاب شده‌اند، ضمن اینکه نرخ بیت از ۵۰۰ کیلوبیت بر ثانیه تا ۶,۷ مگابیت بر ثانیه با پله ۲۰۰ کیلوبیت بر ثانیه‌ای در تغییر است. تنظیمات دیگر کدکننده مشابه آنچه در جدول ۴-۱۲ آمده است، می‌باشد. میزان خطای میانگین برای این شبیه‌سازی در شکل ۴-۹ آمده است. براساس نتایج این شبیه‌سازی و همانطور که در شکل آمده است، متوسط خطای تخمین برای تمامی نرخ بیتها برابر با ۶,۶۴٪ می‌باشد، این در حالیست که میزان خطای تخمین برای نرخ بیتهای پایین (بعنوان نمونه ۰,۵ و ۰,۷ مگابیت بر ثانیه) حدود ۱۱,۳٪ می‌باشد و با افزایش نرخ بیت این خطا به مرور کاهش می‌یابد.



شکل ۴-۹ میزان خطا برای ویدئوهای مختلف 4CIF و روی محدوده وسیعی از نرخ بیتها

۴-۲-۲-۲ ارزیابی مدل پیچیدگی واحد تخمین حرکت در کدکننده H.264/AVC

در این بخش پیچیدگی هر ویدئو بر اساس پارامترهای کدکردن آن {NRef, SR, Res} با معیار تعداد پالسهای ساعت تخمین زده شده و با مقدار واقعی آن مقایسه می‌شود.

این شبیه‌سازیها بر روی بیش از ۱۰۰ ترکیب {NRef, SR, Res} و هنگامی که ۱۰۰ فریم از هر ویدئو کد شده است، انجام گرفته است. نتایج شبیه‌سازی در جدول ۴-۱۶ آمده است.

جدول ۴-۱۶ میزان خطا در تخمین پیچیدگی برای روش‌های تخمین حرکت سریع و تخمین حرکت کامل در سکوها و پیاده‌سازیهای مختلف

ویدئو نمونه		ضریب چندی سازی	تخمین حرکت کامل				تخمین حرکت سریع			
			سکو ۱		سکو ۲		سکو ۱		سکو ۲	
			JM	x.264	JM	x.264	JM	x.264	JM	x.264
CIF	Bus	۲۸	۵,۶۳	۱۰,۰۷	۶,۲۵	۱۴,۸۱	۲۱,۴۶	۲۰,۸۸	۲۲,۵۳	۲۰,۹۶
		۳۲	۶,۸۱	۱۱,۶۰	۷,۹۷	۱۵,۹۱	۱۷,۴۹	۲۳,۴۴	۱۹,۲۴	۲۹,۳۰
	City	۲۸	۵,۷۴	۱۲,۶۱	۵,۸۱	۱۳,۹۷	۱۶,۵۷	۱۸,۰۰	۱۶,۲۴	۱۷,۶۴
		۳۲	۴,۸۲	۱۳,۵۸	۷,۰۶	۱۶,۵۷	۱۳,۶۷	۱۳,۳۰	۱۶,۴۰	۱۹,۶۳
	Crew	۲۸	۶,۴۲	۷,۰۸	۵,۷۸	۹,۸۰	۲۴,۶۹	۲۲,۵۹	۲۶,۴۲	۲۱,۹۱
		۳۲	۵,۰۶	۹,۸۱	۸,۲۸	۱۳,۱۹	۲۵,۴۳	۲۴,۷۳	۲۳,۹۰	۲۵,۷۲
	Stefan	۲۸	۶,۴۹	۱۷,۰۵	۶,۸۴	۱۶,۹۸	۱۵,۷۳	۲۳,۱۳	۱۵,۴۲	۲۳,۸۲
		۳۲	۴,۰۱	۱۸,۵۰	۷,۵۵	۱۶,۹۹	۱۱,۵۲	۲۳,۶۸	۱۳,۸۲	۲۲,۶۰
میانگین خطا			**۵,۶۲**	**۱۲,۴۱**	**۶,۹۶**	**۱۴,۸۳**	**۱۸,۳۲**	**۲۱,۹۷**	**۱۹,۲۵**	**۲۲,۷۰**
4CIF	City	۲۸	۶,۲۲	۱۰,۶۱	۵,۲۷	۱۳,۹۷	۱۹,۸۸	۱۷,۱۶	۱۹,۴۸	۱۶,۸۵
		۳۲	۷,۱۲	۱۲,۵۸	۵,۰۸	۱۱,۲۱	۱۳,۷۰	۲۰,۱۴	۱۴,۳۹	۲۲,۱۵
	Crew	۲۸	۴,۵۸	۱۵,۹۹	۶,۲۷	۸,۴۴	۲۵,۴۳	۲۰,۳۵	۲۶,۱۹	۲۱,۷۷
		۳۲	۴,۰۵	۹,۰۶	۷,۹۴	۱۵,۹۲	۲۴,۶۵	۱۸,۲۵	۲۴,۱۶	۱۸,۰۷
	Harbour	۲۸	۶,۳۵	۱۷,۴۲	۵,۱۴	۱۳,۳۴	۲۰,۴۶	۲۶,۰۸	۲۰,۸۷	۲۴,۷۸
		۳۲	۵,۸۹	۱۵,۵۴	۷,۷۳	۹,۶۶	۲۳,۸۳	۲۳,۲۶	۲۳,۳۵	۲۲,۵۶
	Ice	۲۸	۴,۳۶	۲۱,۹۵	۶,۶۷	۱۷,۸۴	۱۴,۷۲	۲۸,۳۹	۱۵,۷۵	۲۹,۵۳
		۳۲	۵,۶۲	۱۵,۴۷	۷,۳۳	۱۵,۱۵	۱۱,۷۶	۳۰,۹۹	۱۲,۷۰	۲۸,۵۱
میانگین خطا			**۵,۵۲**	**۱۴,۸۳**	**۶,۴۴**	**۱۳,۲۷**	**۱۹,۳۰**	**۲۳,۰۷**	**۱۹,۶۱**	**۲۳,۰۲**

در این جدول نرم افزار مرجع JM [67] و نرم افزار x264 [68] بعنوان پیاده‌سازیهای مختلف استفاده شده‌اند. سکو ۱، یک سیستم چهار هسته ای با پردازنده ۳ گیگا هرتزی، ۴ گیگابایت حافظه و سیستم عامل ویندوز ۷ می‌باشد. سکو ۲، یک نت بوک با پردازنده ۱٫۶۶ گیگا هرتزی اتم، ۱ گیگابایت حافظه و سیستم عامل ویندوز ۷ می‌باشد. همانگونه که در این جدول مشاهده می‌شود، دقت روش پیشنهادی برای مدل کردن پیچیدگی دارای خطای معقول و مناسب می‌باشد. همانگونه که انتظار می‌رفت، کد بهینه شده x264 دارای دقت کمتری می‌باشد، چرا که علاوه بر کد ساده شده دارای مراحل اتمام زودرس و حذف نقاط جستجو نیز می‌باشد.

۳-۴ جمع بندی

در این فصل یک روش شناسی بعنوان روندی کلی برای تخمین مصرف منابع را پیشنهاد کردیم. سپس میزان مصرف توان یا پیچیدگی را برای الگوریتمهای تخمین حرکت در کدکننده و همچنین الگوریتم کدگشایی استاندارد H.264/AVC تخمین زدیم. پس از ارائه مدلی برای تخمین میزان پیچیدگی، هر یک از تخمینهای پیشنهاد شده را در سکوها و پیاده سازیهای مختلف ارزیابی نموده‌ایم. نتایج ارزیابی‌ها نشان می‌دهد که هر دو روش پیشنهادی با دقت مناسبی میزان مصرف پیچیدگی را تخمین می‌زنند. روش تخمین پیچیدگی کدگشا در پیاده سازی کدکننده مطلع از محدودیتهای گیرنده و کدکننده مطلع از محدودیتهای گیرنده برای کاربرد تطبیق، به ترتیب در فصول ۵ و ۶ به کار خواهد آمد. روش تخمین پیچیدگی تخمین حرکت نیز در طراحی کدکننده مطلع از محدودیتهای خویش در فصل ۷ به کار خواهد رفت.

فصل پنجم

طراحی کدکننده مطلع از محدودیت‌های گیرنده - نمونه مطالعاتی

در این فصل یک کدکننده مطلع از محدودیتهای گیرنده طراحی خواهد شد. برای طراحی چنین کدکننده‌ای روندی که در روش انتزاعی پیشنهاد شده در فصل ۳، معرفی کردیم را دنبال خواهیم کرد. از آنجا که در نظر گرفتن تعداد زیادی محدودیت عملی نمی‌باشد، لذا در این فصل، مهمترین محدودیت ابزارهای قابل حمل یعنی پیچیدگی را به همراه نرخ بیت بعنوان محدودیت‌های گیرنده لحاظ کرده‌ایم. با داشتن محدودیت پیچیدگی و اعوجاج، مسئله بهینه‌سازی بصورت رابطه ریاضی (۱-۵) در خواهد آمد.

$$\begin{cases} Min\ (D) \\ s.t.\ \ R < R_c, C < C_c \end{cases} \quad (۱-۵)$$

در این رابطه، D میزان اعوجاج، R_c نرخ بیت از پیش تعیین شده، R نرخ بیت ویدئو، C_c پیچیدگی از پیش تعیین شده و C میزان پیچیدگی نهایی ویدئو می‌باشد.

با داشتن رابطه D، R و C براساس پارامترهای یکسان می‌توان رابطه (۱-۵) را به رابطه مقید (۲-۵) تبدیل کرده و پارامترهای لاگرانژ λ_r و λ_c را استخراج نمود.

$$J = D + \lambda_r \times R + \lambda_c \times C \quad (۲-۵)$$

با داشتن پارامترهای λ_r و λ_c، می‌توان برای هر مد ماکروبلوک، میزان نرخ، اعوجاج و پیچیدگی را بدست آورد و مدی را انتخاب کرد که منجر به J کمینه می‌شود. نهایتا کنترل کننده پیچیدگی با توجه به توان موجود ابزارهای پیچیدگی را در سطوح مختلف تنظیم خواهد کرد تا رشته بیت تولید شده محدودیت‌های پیچیدگی را در هنگام کدگشایی برآورده سازد. در ادامه هر یک از این مراحل را با جزئیات آن تشریح خواهیم کرد.

۵-۱ پیاده سازی مدل انتزاعی برای کدکننده H.264/AVC مطلع از محدودیت پیچیدگی گیرنده

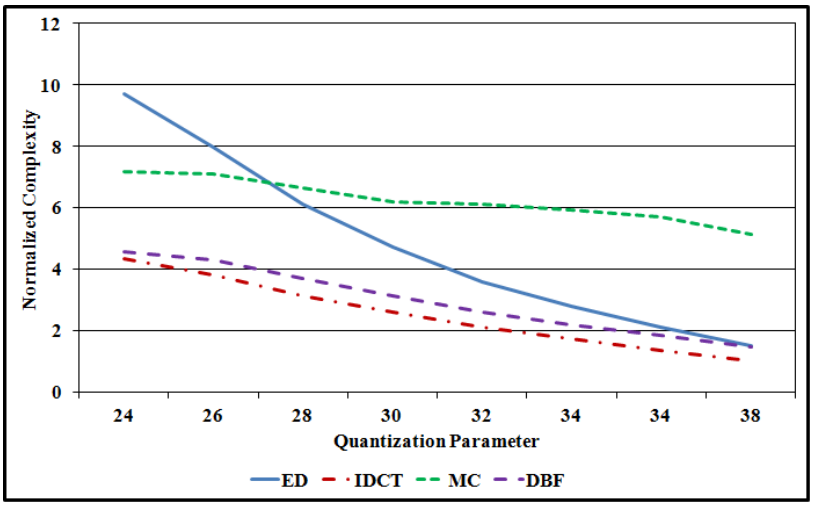
در این بخش با طی چند مرحله بر اساس روش انتزاعی پیشنهادی، اقدام به طراحی یک کنترل کننده پیچیدگی خواهیم کرد که بتواند محدودیت توان کدگشا را در حین کد کردن در نظر بگیرد. لازم به ذکر است که این کنترل کننده پیچیدگی در کنار کنترل کننده نرخ بیت عمل می‌کند به نحوی که علاوه بر برآورده شدن محدودیت توان گیرنده، محدودیت نرخ بیت آن نیز برآورده شده و کمترین اعوجاج ممکن تحمیل گردد.

۵-۱-۱ انتخاب پارامترهای کدکننده

با بررسی روابط پیچیدگی که در فصل قبل بدست آمد، می‌توان پارامترهای موثر بر پیچیدگی را استخراج کرد. همانطور که در رابطه (۴-۱۲) مربوط به کدگشای آنتروپی مشاهده شد، پارامترهای پیچیدگی به مقادیر مانده وابسته هستند. به عبارت دیگر با داشتن مانده بیشتر، پارامترهای پیچیدگی بزرگتری خواهیم داشت که خود به پیچیدگی بیشتر منجر خواهد شد. میزان اطلاعات مانده نیز خود به پارامتر چندی سازی و مد وابسته است. برای واحد IDCT نیز عملیات تبدیل تنها برای بلاکهایی انجام می‌شود که دارای CBP غیر صفر می‌باشد. این در حالی است که پارامتر CBP عملا نشانگر میزان مقادیر مانده می‌باشد. برای واحد بلوک زدائی نیز تعداد فیلترهای قوی و معمولی تابعی از مد ماکروبلوک و پارامترهای چندی سازی می‌باشد. برای واحد جبران حرکت، تعداد عملیات درون یابی و دسترسی به حافظه، تابعی از مد ماکروبلوک و دقت عمیات تخمین حرکت می‌باشند. به این ترتیب واضح است که پیچیدگی واحدهای کدگشایی آنتروپی، IDCT و فیلتر بلوک زدائی بصورت مستقیم تابعی از پارامتر چندی سازی می‌باشد، حال آنکه پیچیدگی واحد جبران حرکت بصورت غیر مستقیم تابعی از این پارامتر می‌باشد. چرا که وقتی یک ویدئو با پارامتر چندی سازی بزرگتری کد می‌شود، ماکروبلوکهای بیشتری بصورت Skip و ۱۶×۱۶ کد می‌شوند و به این ترتیب تعداد بردارهای حرکت با دقت ¼ کمتری نسبت به حالتی که پارامتر چندی سازی بزرگتر داریم، خواهیم داشت. به این ترتیب با افزایش پارامتر چندی سازی، پیچیدگی کاهش خواهد یافت.

برای بررسی شهودی تاثیر تغییر پارامتر چندی سازی بر پیچیدگی واحدهای مختلف، تعدادی شبیه‌سازی بر روی ویدئوهای مختلف انجام داده‌ایم و برای هر واحد پیچیدگی میانگین ویدئوهای مختلف را برای هر پارامتر پیچیدگی بدست آورده ایم. این کار را برای پارامترهای چندی سازی مختلف تکرار کرده‌ایم و نتایج این شبیه‌سازی را در شکل ۵-۱ آورده ایم.

همانطور که در شکل آمده است، پیچیدگی واحد کدگشایی آنتروپی به شدت به پارامتر چندی سازی وابسته است، این در حالی است که واحدهای IDCT و فیلتر بلوک زدائی رفتار خطی در مقابل پارامتر چندی سازی دارند. همانطور که انتظار داشتیم پیچیدگی واحد جبران حرکت کمترین وابستگی را به پارامتر چندی سازی دارد، هر چند که سیر نزولی کاهش پیچیدگی در ازای افزایش پارامتر چندی سازی در این شکل مشهود است.



شکل ۵-۱پیچیدگی واحدهای کدگشایی برای پارامترهای چندی سازی مختلف

به این ترتیب پارامتر چندی سازی را بعنوان مهمترین پارامتر کدکننده موثر در مصرف توان مد نظر قرار خواهیم داد.

۵-۱-۲ بدست آوردن مدل هر منبع

از آنجا که قبلا برای اعوجاج و نرخ بیت، مدلهایی بر اساس پارامتر چندی سازی پیشنهاد شده است، در این رساله نیز به همین روابط بسنده می‌کنیم. به این ترتیب D و R، بعنوان تابعی از پارامتر چندی سازی و طبق روابط زیر تعریف می‌شوند [70].

$$D(Q) = \frac{Q^2}{3} \quad (۳-۵)$$

$$R(Q) = a\log_2^{\frac{3b}{Q^2}} \quad (۴-۵)$$

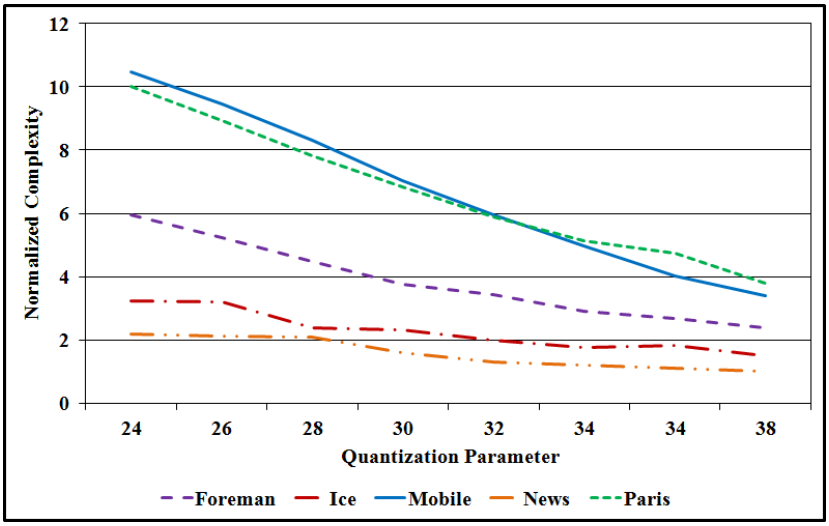
لازم به ذکر است که بین Q و QP رابطه آمده در (۵-۵) برقرار است. بنابراین در ادامه این رساله ممکن است در بعضی اوقات روابط و شبیه سازیها را با Q و برخی اوقات با QP نشان می‌دهیم که تبدیل این دو به یکدیگر طبق رابطه (۵-۵) صورت خواهد پذیرفت.

$$Q = 2^{(\frac{QP-12}{3})} \quad (۵-۵)$$

از آنجا که برای پیچیدگی نیز پارامتر چندی سازی را بعنوان مهمترین پارامتر در نظر گرفتیم لذا در ادامه مدلی برای پارامتر پیچیدگی بر اساس پارامتر چندی سازی پیشنهاد خواهیم کرد.

۵-۱-۲-۱ مدل پیچیدگی بعنوان تابعی از پارامتر چندی سازی

جهت مدل کردن پیچیدگی کدگشا بعنوان تابعی از پارامتر چندی سازی، پیچیدگی واقعی واحدهای مختلف بایستی استخراج شود. برای این کار تعدادی ویدئو نمونه، با استفاده از نرم افزار مرجع H264/AVC [67] کد و کدگشایی شده‌اند. در حین عملیات کدگشایی، پیچیدگی واحدهای مختلف استخراج شده و با هم جمع شده است. این عملیات به ازای پارامترهای پیچیدگی متنوعی تکرار شده است تا بتوان رفتار پیچیدگی کدگشا به ازای تغییر پارامتر چندی سازی را بررسی نمود. نتایج این آزمایش برای تعدادی ویدئو CIF در شکل ۵-۲ و برای پارامترهای چندی سازی بین ۲۴ تا ۳۸ آمده است.



شکل ۵-۲ پیچیدگی کدگشا برای ویدئوهای مختلف در ابعاد CIF و با پارامترهای چندی سازی متنوع

همانطور که در این شکل نیز مشاهده می‌شود، پیچیدگی کدگشا را می‌توان بصورت تابعی خطی از پارامتر چندی سازی تعریف کرد. همانطور که می‌دانیم، نرخ بیت و اعوجاج بصورت توابعی از Q تعریف شده‌اند، ما نیز همین پارامتر را استفاده خواهیم کرد. بنابراین با استفاده از ابزار برازش منحنی نرم افزار Matlab یک مدل خطی برای پیچیدگی بعنوان تابعی از Q ارائه خواهیم داد. این تابع در رابطه (۵-۶) آمده است.

$$C(Q) = \frac{m}{Q} + C_0 \qquad (۵-۶)$$

در رابطه (۵-۶) C_0 یک مقدار ثابت و m شیب مدل پیچیدگی می‌باشد. هر دو این پارامترها به محتوای ویدئو و نرخ بیت وابسته بوده و قابل تنظیم در یکسری شبیه‌سازی اولیه می‌باشند. به عبارت دیگر، برای تنظیم این پارامترها، پیچیدگی تعدادی از ماکروبلوکهای فریم‌های ابتدایی توسط مدل پیچیدگی پیشنهادی در فصل قبل استخراج شده و بعنوان $C(Q)$ در نظر گرفته خواهد شد. با داشتن مقدار Q، می‌توان مقدار پارامترهای مجهول C_0 و m را استخراج نمود.

به این ترتیب با داشتن یک مدل کلی برای پیچیدگی که بعنوان تابعی از پارامتر Q می‌باشد، می‌توان پارامترهای لاگرانژ را استخراج نمود. در بخش بعدی به نحوه استخراج پارامترهای لاگرانژ با داشتن روابط ریاضی پیچیدگی، اعوجاج و نرخ بیت خواهیم پرداخت.

۵-۱-۳ حل مسئله بهینه‌سازی و استخراج پارامترهای لاگرانژ

برای بدست آوردن روابط پارامترهای لاگرانژ، از روشی که در [70] برای بهینه‌سازی نرخ-اعوجاج در کدکننده متداول ارائه شده است کمک گرفته ایم، به همین دلیل در ادامه روش پیشنهادی در [70] را می‌آوریم و در ادامه روش پیشنهادی برای استخراج پارامترهای لاگرانژ را خواهیم داشت.

۵-۱-۳-۱ الگوریتم بهینه‌سازی نرخ-اعوجاج در کدکننده متداول

در کدکننده متداول کمینه اعوجاج به شرط برآوردن محدویت نرخ بیت مد نظر است. این مسئله بصورت روابط ریاضی در رابطه (۵-۷) آمده است.

$$\begin{cases} Min\,(D) \\ s.t.\;\; R < R_c \end{cases} \qquad (۷-۵)$$

در این رابطه، D میزان اعوجاج، R_c نرخ بیت از پیش تعیین شده، R نرخ بیت ویدئو می‌باشد.

با استفاده از روش لاگرانژ رابطه غیر مقید آمده در رابطه (۵-۷) به رابطه مقید زیر تبدیل خواهد شد.

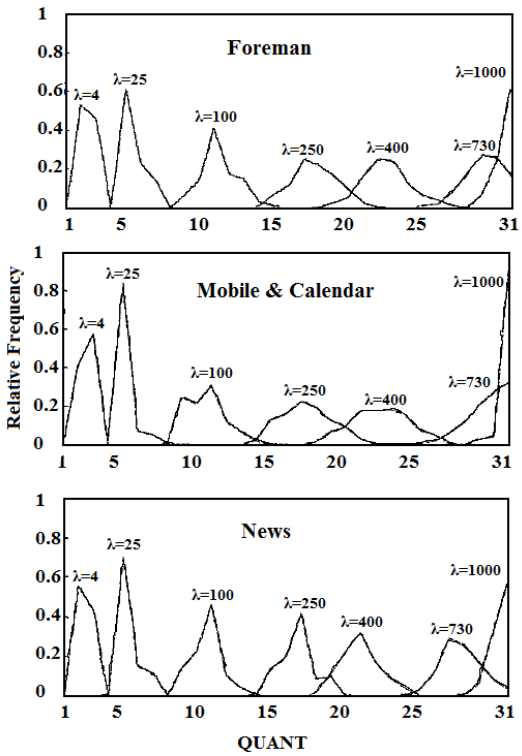
$$J = D + \lambda \times R \qquad (۸-۵)$$

با داشتن مدلهای نرخ بیت و اعوجاج طبق روابط (۵-۳) و (۵-۴) می‌توان رابطه (۵-۸) را حل کرده و پارامتر λ را استخراج نمود که این پارامتر در رابطه (۵-۹) آمده است [70].

$$\lambda = \frac{ln2}{3a} Q^2 \qquad (۹-۵)$$

مقدار پارامتر λ با انجام یکسری عملیات شبیه‌سازی و بصورت تجربی $0.85Q^2$ تعریف شده است.

مولفین [70]، با روش دیگری نیز مقدار λ را استخراج کرده‌اند. برای استخراج این پارامتر، به ازای هر Q، ضرایب λ متنوعی در نظر گرفته شده است. به عبارت دیگر، در حین عملیات انتخاب مد برای یک ماکروبلوک، عملیات بهینه‌سازی آمده در (۵-۸) به ازای λ مختلفی انجام می‌شود و مقدار λای که منجر به کمترین هزینه شود، ذخیره خواهد شد. به این ترتیب در انتهای عملیات فشرده‌سازی برای یک Q، می‌توان درصد ماکروبلوکهایی را بدست آورد که هزینه نهایی آنها، به ازای یک λ خاص، کمینه شده است. این عملیات برای تمامی ترکیبات Q و λ تکرار خواهد شد. به این ترتیب نموداری مشابه با شکل ۵-۳ بدست خواهد آمد [70].



شکل ۵-۳ فراوانی نسبی در برابر Q برای مقادیر مختلف λ [70]

با بررسی نمودارهای آمده در شکل ۵-۳ مشاهده می‌شود که به ازای هر Q یک λ دارای فراوانی قابل توجهی نسبت به λهای دیگر می‌باشد. این قله‌ها برای ویدئوهای مختلف نیز مشابه می‌باشد. به این ترتیب مولفین [70] مقدار $0.85Q^2$ را برای λ پیشنهاد کرده‌اند.

۵-۱-۳-۲ روش پیشنهادی برای حل مسئله بهینه‌سازی و استخراج پارامترهای لاگرانژ

با توجه به روند طی شده در [70]، می‌توان روند مشابهی برای حل مسئله مد نظر این بخش، که بهینه‌سازی توامان پیچیدگی و نرخ بیت می‌باشد، در پیش گرفت. همانطور که در ابتدای این فصل آوردیم، برای برآوردن توام محدودیت‌های پیچیدگی و نرخ بیت و همچنین رسیدن به کمترین اعوجاج بایستی رابطه (۵-۱۰) را کمینه نمود.

$$J = D + \lambda_r \times R + \lambda_c \times C \qquad (۱۰-۵)$$

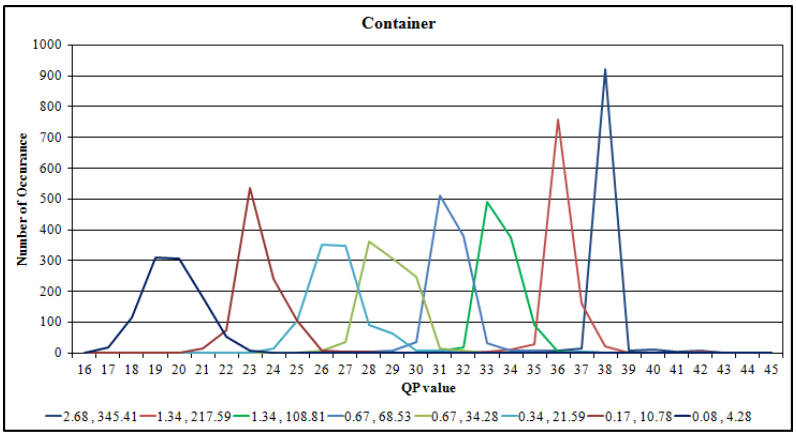
در این رابطه، D، R و C هر سه تابعی از Q می‌باشند، همانطور که به ترتیب در روابط (۵-۳)،و (۵-۴) و (۵-۶) آمده است. با جایگزینی این سه رابطه در رابطه (۵-۱۰) خواهیم داشت:

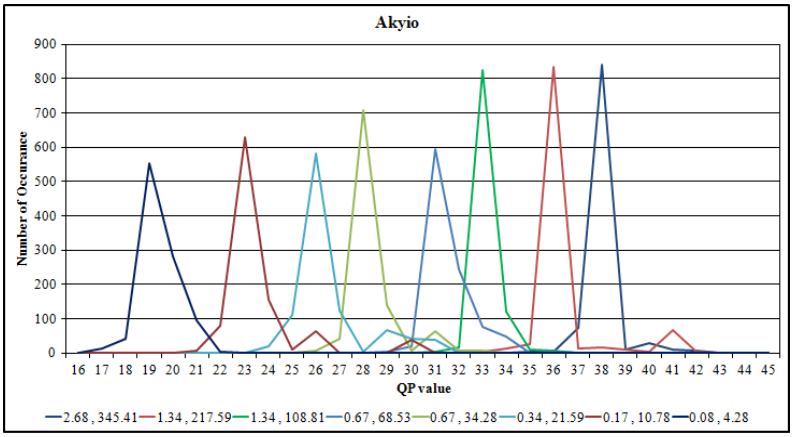
$$J(Q) = \frac{Q^2}{3} + \lambda_r \times \left(a log_2^{\frac{3b}{Q^2}} \right) + \lambda_c \times \left(\frac{m}{Q} + C_0 \right) \qquad (۱۱-۵)$$

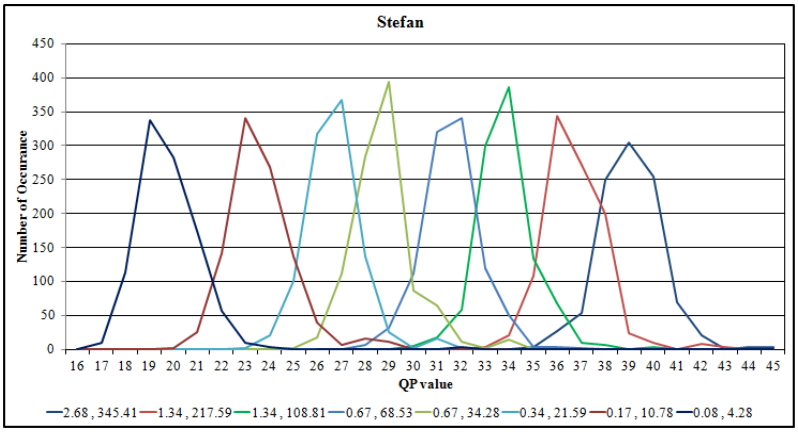
برای کمینه کردن رابطه (۵-۱۱) بایستی $\frac{\partial J(Q)}{\partial Q}$ را برابر صفر قرار دهیم و به این ترتیب به رابطه‌ای بین Q، λ_r و λ_c خواهیم رسید.

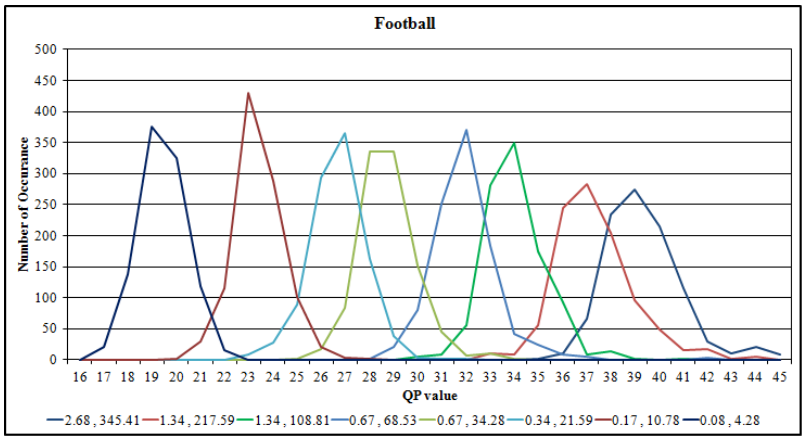
$$2\,log(2)\,Q^3 - 6aQ\lambda_r - 3\,log(2)\,m\lambda_c = 0 \qquad (۱۲-۵)$$

لازم به ذکر است که بر اساس شرایط روش لاگرانژ، مقادیر پارامترهای λ_c و λ_r بزرگتر یا مساوی صفر خواهد بود. همانطور که در رابطه (۵-۱۲) مشاهده می‌شود، هر کدام از پارامترهای λ_c و λ_r را می‌توان بصورت تابعی از پارامتر دیگر و Q بدست آورد. برای بدست آوردن هر کدام از این پارامترها بصورت مستقل، روشی مشابه با روش آمده در [70] را دنبال خواهیم کرد. به این ترتیب که به ازای ترکیبات مختلف λ_c و λ_r ماکروبلوکها را با مقادیر متنوعی از Qها کد کرده و مقدار Qای که منجر به کمترین اعوجاج خواهد شد را در نظر خواهیم گرفت. این عملیات را برای تمامی ماکروبلوکها تکرار کرده و نهایتا فراوانی هر Q را به ازای هر یک از ترکیبات λ_c و λ_r بدست آورده و Q با بیشترین فراوانی را بعنوان Q غالب ($Q_{Dominant}$) خواهیم شناخت. در شکل ۵-۴ فراوانی QPهای مختلف را برای تعدادی از ترکیبات λ_c و λ_r و برای ویدئوهای مختلف آورده ایم. همانطور که در این شکل مشاهده می‌شود، مشابه با نتایج [70]، QP غالب برای هر یک از ترکیبات λ_c و λ_r در ویدئوهای مختلف به همدیگر بسیار نزدیک است.









شکل ۵-۴ فراوانی QP برای برخی از ترکیبات λ_c و λ_r در ویدئوهای مختلف

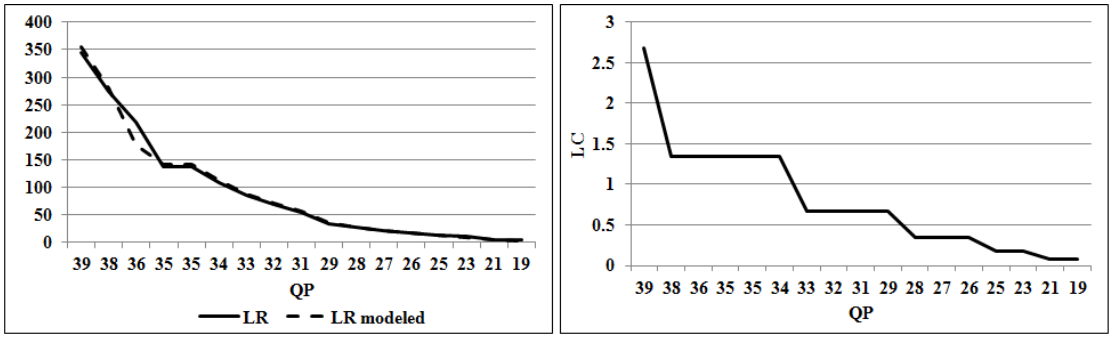
از آنجا که ممکن است برای ترکیبات متفاوتی از λ_c و λ_r، $Q_{Dominant}$ یکسانی داشته باشیم لذا از بین این ترکیبات، ترکیبی انتخاب می‌شود که منجر به کمینه شدن رابطه (۵-۱۲) گردد. با قرار دادن ترکیبات بهینه λ_c، λ_r و $Q_{Dominant}$، در یک جدول جستجو می‌توان در هنگام کد کردن هر ماکروبلوک، مقادیر λ_c و λ_r را با توجه به مقدار Q استخراج نمود. ترکیبات نهایی λ_c، λ_r و QP را در جدول ۵-۱ آورده ایم.

جدول ۵-۱ مقادیر λ_c و λ_r به ازای QPهای مختلف

QP		
39	345.41	2.68
38	274.16	1.34
36	217.59	1.34
35	137.09	1.34
35	137.09	1.34
34	108.81	1.34
33	86.34	0.67
32	68.53	0.67
31	54.41	0.67
29	34.28	0.67
28	27.19	0.34
27	21.59	0.34
26	17.13	0.34
25	13.59	0.17
23	10.78	0.17
21	5.41	0.08
19	4.28	0.08

رابطه پارامترهای λ_c و λ_r با پارامتر QP در شکل ۵-۵ آمده است. همانطور که در شکل نشان داده شده است، می‌توان رابطه بین λ_r با پارامتر QP را، با عملیات برازش منحنی و با دقت مناسبی، طبق رابطه (۵-۱۳) نیز بدست آورد.

$$\lambda_r = 0.6926 \times 2^{\frac{QP-12}{3}} \qquad (۱۳-۵)$$



شکل ۵-۵ رابطه پارامتر λ_c و QP، رابطه پارامتر λ_r و QP و مدل ریاضی آن

با داشتن پارامترهای λ_c و λ_r متناظر با هر QP در هنگام انجام عملیات انتخاب مد، می‌توان بر اساس رابطه (۵-۱۰) بهترین مد را از نظر نرخ بیت، اعوجاج و پیچیدگی بدست آورد. در ادامه روش پیشنهادی برای کنترل پیچیدگی را ارائه خواهیم کرد.

۵-۱-۴ طراحی کنترل کننده پیچیدگی

در بخش قبل نحوه بدست آوردن پارامترهای لاگرانژ مربوط به بهینه‌سازی توامان نرخ بیت و پیچیدگی را ارائه کردیم. این پارامترها در حین انجام عملیات انتخاب مد به کار گرفته می‌شوند تا از بین مدهای موجود مد بهینه – مدی که علاوه بر برآوردن محدودیت نرخ بیت، محدودیت پیچیدگی را نیز برآورده کرده و منجر به کمترین اعوجاج گردد – انتخاب شود. برای کنترل توان مصرفی، مشابه با کنترل کننده نرخ بیت به یک کنترل کننده پیچیدگی نیز نیاز خواهیم داشت. این کنترل کننده پیچیدگی با ارزیابی توان باقیمانده، عملکرد واحدهای مختلف را در سطوح متفاوت به نحوی تنظیم می‌کند که محدودیت پیچیدگی کدگشا لحاظ گردد. در این بخش، کنترل کننده پیچیدگی پیشنهادی را معرفی کرده و مراحل مختلف آن را مرور خواهیم کرد. در ابتدای عملیات کدکردن فرض بر این است که محدودیت پیچیدگی کدگشا بصورت یک پارامتر ورودی به کدکننده داده شده است. همچنین لازم به یادآوری می‌باشد که در بدست آوردن پیچیدگی واحدهای مختلف سمت کدگشا، از مدلهای پیچیدگی ارائه شده در فصل ۴ استفاده شده است.

شبه‌کد کلی مربوط به کنترل کننده پیچیدگی در لیست ۵-۱ آمده است. در ابتدای عملیات و در سطح فریم، کنترل کننده پیچیدگی توان قابل مصرف فریم جاری (C_{Frame}) را با توجه به میزان توان باقی مانده (C_{Total}) و تعداد فریم‌های کد نشده بدست می‌آورد. در همین مرحله، توان مصرف شده توسط آخرین فریم ($C_{Frame}^{Consumed}$) با توان موجود برای فریم فعلی (C_{Frame}) مقایسه می‌شود. در صورتیکه توان مصرف شده توسط آخرین فریم کد شده از توان موجود برای فریم فعلی بیشتر باشد، فیلتر بلوک زدائی در سطح فریم غیر فعال خواهد شد. به این ترتیب برای تمامی ماکروبلوکهای موجود در فریم جاری عملیات بلوک زدائی انجام نخواهد شد. در ادامه عملیات اختصاص نرخ بیت در سطح فریم صورت می‌پذیرد که مهمترین خروجی آن مقدار پارامتر QP خواهد بود. همانطور که در بخش ۵-۱-۲-۱ اشاره شد، توان مصرفی تابعی از پارامتر QP می‌باشد و عملا مهمترین پارامتری است که کنترل کننده پیچیدگی برای تنظیم میزان توان مصرفی از آن استفاده می‌کند. پس از تعیین مقدار پارامتر QP توسط کنترل کننده نرخ بیت، این پارامتر توسط پروسه *AdjustQP* تنظیم خواهد شد. پروسه *AdjustQP* عملا مقدار QP را با توجه به میزان توان مصرفی و توان موجود تغییر خواهد داد. جزئیات مربوط به این پروسه بعدا در همین بخش ارائه خواهد شد.

پس از اختصاص توان به فریم جاری، عملیات اختصاص توان به BU (C_{BU}) صورت می‌پذیرد. عملیات اختصاص توان به BUها به تعداد BUهای موجود در فریم – که یکی از پارامترهای ورودی کد کننده می‌باشد – تکرار می‌شود. پس از اختصاص توان به هر BU عملیات کنترل نرخ بیت در سطح BU صورت می‌گیرد. خروجی این مرحله عملا مقدار پارامتر QP خواهد بود. مشابه با بخش مربوط به اختصاص توان به فریم، پس از تعیین مقدار پارامتر QP توسط کنترل کننده نرخ بیت، این پارامتر توسط پروسه *AdjustQP* نیز تنظیم خواهد شد. با تنظیم پارامتر QP در سطح BU، عملیات کدکردن در سطح ماکروبلوک شروع خواهد شد. در این مرحله، عملیات بهینه‌سازی نرخ بیت و اعوجاج برای مدهای مختلف صورت خواهد پذیرفت. برای این کار پیچیدگی واحدهای مختلف – با استفاده از مدل آمده در فصل ۴– محاسبه شده و پیچیدگی مد مورد نظر بدست خواهد آمد. با توجه به پارامتر QP مقدار پارامتر λ_c نیز از روی یک جدول جستجو مشابه با جدول ۵-۱ استخراج

خواهد شد. با داشتن نرخ بیت و اعوجاج هر مد و همچنین محاسبه پارامتر λ_r طبق رابطه (۵-۱۳)، می‌توان هزینه توامان نرخ بیت، پیچیدگی و اعوجاج (*JRDC*) را محاسبه نمود. واضح است که از بین مدهای مختلف، مدی انتخاب می‌شود که منجر به *JRDC* کمینه گردد. پس از انتخاب مد بهینه و به روز کردن میزان توان مصرفی توسط ماکروبلوک حاضر ($C_{MB}^{Consumed}$)، میزان توان مصرفی BU ($C_{BU}^{Consumed}$) بروز می‌گردد. عملیات مذکور برای تمامی BUهای یک فریم تکرار خواهد شد. پس از اتمام کدکردن یک فریم، میزان توان باقیمانده (C_{Total}) با توجه به میزان توان مصرفی توسط فریم حاضر($C_{Frame}^{Consumed}$) به روز خواهد شد و عملیات کنترل پیچیدگی بر روی فریم بعدی ادامه خواهد یافت.

For (each frame)
{
$C_{Frame} = \frac{C_{Total}}{Number\ of\ remaining\ frames\ in\ sequence}$
If ($C_{Frame}^{Consumed} > C_{Frame}$)
Disable DBF for current frame
Perform frame level RateControl
AdjustQP
For (each BasicUnit)
{
$C_{BU} = \frac{C_{Frame}}{Number\ of\ Basic\ Units\ in\ frame}$
Perform BasicUnit level RateControl
AdjustQP
For (each MB)
{
For (each mode)
{
Extract modules complexity using decoder complexity model
$C_{MB}^{Mode} = C_{ED} + C_{IDCT} + C_{MC} + C_{DBF}$
Assign λ_c *based on QP value*
$JRDC = D + \lambda_c \times R_{MB}^{Mode} + \lambda_c \times C_{MB}^{Mode}$
}
Update best mode & $C_{MB}^{Consumed}$
$C_{BU}^{Consumed} += C_{MB}^{Consumed}$
}
$C_{Frame}^{Consumed} += C_{BU}^{Consumed}$
}
$C_{Total} -= C_{Frame}^{Consumed}$
}

لیست ۵-۱ شبه‌کد کنترل کننده پیچیدگی برای کدکننده مطلع از محدودیت‌های گیرنده

همانطور که اشاره شد، تابع *AdjustQP* مسئول تنظیم پارامتر QP با در نظر گرفتن توان مصرفی و توان موجود می‌باشد، شبه‌کد مربوط به این تابع را در لیست ۵-۲ آورده ایم. از آنجا که این تابع در دو سطح فریم و BU فراخوانی می‌شود، ورودی

عمومی α را برای آن لحاظ کرده‌ایم که می‌تواند فریم یا BU فعلی باشد که در توضیح این پروسه "واحد" شناخته خواهد شد. همانطور که در لیست ۵-۲ مشخص است، در ابتدای این پروسه میزان توان مصرفی برای واحد فعلی ($C_\alpha^{Consumed}$) با میزان توان مصرفی میانگین برای تمامی واحدهای کدشده ($C_\alpha^{Average\ Consumed}$) جمع شده و در پارامتر *Cons* قرار خواهد گرفت. به همین ترتیب میزان توان موجود برای واحد فعلی (C_α) و میانگین توان موجود برای تمامی واحدهای کدشده ($C_\alpha^{Average}$) جمع شده و در پارامتر *Avail* قرار خواهد گرفت. سپس پارامترهای *Cons* و *Avail* با هم مقایسه می‌شود. چنانچه پارامتر *Cons* از *Avail* بزرگتر باشد، این نشان می‌دهد که توان مصرف شده بیشتر از توان موجود بوده است و لذا بایستی پارامتر QP افزایش یابد، البته این افزایش با در نظر گرفتن محدودیت‌های استاندارد انجام خواهد شد. در صورتیکه توان موجود از توان مصرفی بیشتر باشد، می‌توان پارامتر QP را کاهش داد تا ویدئو با کیفیت بهتری کد شود ضمن اینکه لازم است محدودیت نرخ بیت نیز لحاظ شود تا مصرف نرخ بیت، بیشتر از میزان تعیین شده نگردد.

```
AdjustQP(α)
{
        Cons = C_α^Consumed + C_α^Average Consumed
        Avail = C_α + C_α^Average
        If (Cons > Avail)
                If (Avail > 0)
                        QP+=min((Cons/Avail),4)
                Else
                        QP+=4
        Else
                If R_α^Consumed < (bitrate / (Number of total frames × Number of α per frame)) × number of coded α
                        QP=Max(0,QP-Min((Avail/Cons),4))
}
```

لیست ۵-۲ شبه‌کد مربوط به تابع AdjustQP()

با پیاده‌سازی این کنترل کننده پیچیدگی درون نرم افزار H.264/AVC، می‌توان یک کدکننده مطلع از محدودیت‌های گیرنده داشت. در بخش بعدی به مطالب مرتبط با پیاده‌سازی و نتایج شبیه‌سازی مربوطه خواهیم پرداخت.

۵-۲ نتایج شبیه‌سازی

در بخشهای پیشین نحوه طراحی کدکننده مطلع از محدودیت‌های گیرنده را تشریح کردیم، در این قسمت کارائی روش پیشنهادی را ارزیابی خواهیم کرد. برای ارزیابی روش پیشنهادی، عملکرد آن را در مقایسه با کدکننده معمول بررسی خواهیم کرد. در شبیه‌سازیهای انجام شده تنظیمات کدکننده مطابق با جدول ۵-۲ می‌باشد.

جدول ۵-۲ تنظیمات کدکننده H.264/AVC

پروفایل	Baseline
سطح	۳
تعداد فریم‌های کد شده	۱۰۰
تعداد فریم‌های مرجع	۱
محدوده جستجو	۱۶
بهینه ساز نرخ – اعوجاج (RDO)	روشن
کنترل کننده نرخ بیت	روشن
دقت تخمین حرکت	¼ پیکسل

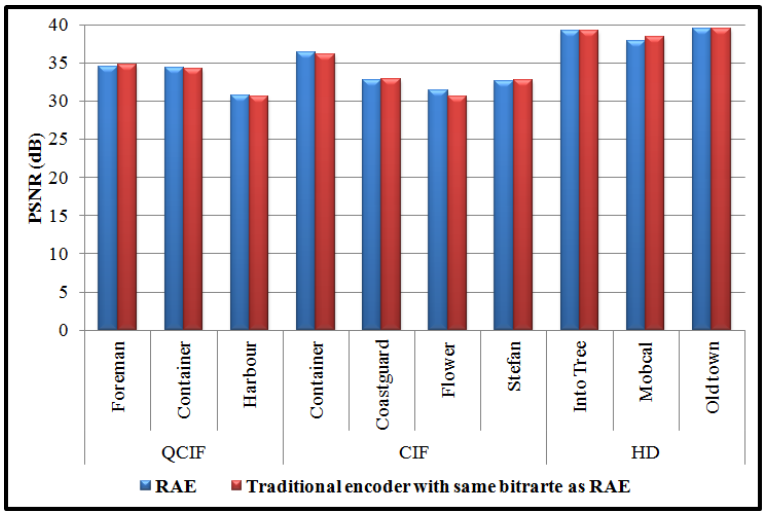
جهت ارزیابی کارائی کدکننده مطلع از محدودیت‌های گیرنده، دو مجموعه شبیه‌سازی صورت پذیرفته است. در شبیه‌سازی اول، ویدئو نمونه بدون هیچگونه محدودیت پیچیدگی و با استفاده از کدکننده متداول H.264/AVC فشرده شده است. سپس رشته بیت تولید شده کدگشایی شده و مشخصات کیفیتی نظیر نرخ بیت و PSNR و همچنین میزان مصرف توان کدگشایی آن استخراج شده است. در شبیه‌سازی دوم، ویدئو نمونه با استفاده از کدکننده مطلع از محدودیت‌های گیرنده پیشنهادی و در حالی که توان مصرفی کدگشائی به اندازه ۳۰٪ کاهش یافته است، فشرده می‌شود. همانطور که پیشتر نیز اشاره شد، کدکننده مطلع از محدودیت‌های گیرنده پیشنهادی با استفاده از کنترل کننده پیچیدگی و مدل پیچیدگی، توان مصرفی را بین واحدهای مختلف توزیع خواهد کرد. در جدول ۵-۳ کارائی کدکننده مطلع از محدودیت‌های گیرنده پیشنهادی هنگامیکه توان کدگشا به میزان ۳۰٪ کاهش یافته است، در مقایسه با کدکننده معمول آورده شده است. همانگونه که نتایج این جدول نشان می‌دهد میزان کاهش کیفیت بصورت میانگین ۱,۰۷ دسیبل می‌باشد، ضمن اینکه محدودیت توان گیرنده در حین کدکردن لحاظ شده است.

جدول ۵-۳ کارائی کدکننده مطلع از محدودیت‌های گیرنده در حالی که توان مصرفی گیرنده ۳۰٪ کاهش یافته است

اندازه ویدئو	ویدئو نمونه	کد کننده متداول		کدکننده مطلع از محدودیت‌های گیرنده	
		کیفیت (dB)	نرخ بیت (Kbps)	کیفیت (dB)	نرخ بیت (Kbps)
QCIF	Foreman	۳۵,۹۳	۱۲۰,۴۴	۳۴,۵۹	۹۰,۸۹
	Container	۳۴,۴	۵۰,۱۵	۳۴,۴۳	۴۸,۸۳
	Harbour	۳۲,۹۷	۳۵۰,۴۷	۳۰,۷۳	۲۲۹,۸۵
CIF	Container	۳۶,۴۶	۲۴۹,۹	۳۶,۳۸	۲۲۵,۲
	Coastguard	۳۴,۲۲	۱۳۰۳	۳۲,۷۴	۹۵۹,۸
	Flower	۳۲,۵۲	۱۵۵۷	۳۱,۴۳	۹۰۷,۷
	Stefan	۳۴,۷۵	۱۵۰۱	۳۲,۶۵	۱۰۳۷
HD	Into Tree	۴۰,۰۹	۱۸۰۰۰	۳۹,۲	۱۳۷۶۴
	Mobcal	۳۹,۰۸	۲۴۹۹۷	۳۷,۹	۲۰۷۸۱
	Old town	۴۰,۱۲	۱۷۰۰۱	۳۹,۵	۱۴۱۵۰

برای ارزیابی کارائی کدکننده مطلع از محدودیتهای گیرنده پیشنهادی بصورت عادلانه تر، یک شبیه‌سازی دیگر با کدکننده متداول انجام داده‌ایم. در این شبیه‌سازی، ویدئو نمونه با نرخ بیتی معادل با نرخ بیت حاصل از کدکننده مطلع از

محدودیت‌های گیرنده، در حالی که توان گیرنده ۳۰٪ کاهش یافته است، فشرده شده است. نتایج شبیه‌سازی این دو روش در شکل ۵-۶ آمده است. همانطور که در شکل نشان داده شده است، هنگامیکه نرخ بیت دو شبیه‌سازی یکسان است، کدکننده مطلع از محدودیت‌های گیرنده بسیار نزدیک به کدکننده متداول رفتار کرده است، که این رفتار کارائی کدکننده مطلع از محدودیت‌های گیرنده پیشنهادی را نشان می‌دهد.



شکل ۵-۶ کارائی کدکننده مطلع از محدودیت‌های گیرنده پیشنهادی با روش متداول در نرخ بیت یکسان

۵-۳ جمع بندی

در این فصل با دنبال کردن روند انتزاعی پیشنهادی در فصل ۳ یک کدکننده مطلع از محدودیتهای گیرنده طراحی شد. این کدکننده محدودیت نرخ بیت و پیچیدگی را بعنوان محدودیتهای منابع گیرنده در نظر می‌گیرد و با بکارگیری یک کنترل کننده اقدام به تولید رشته بیت مناسب می‌نماید. کدکننده مطلع از محدودیتهای گیرنده پیشنهادی برای بدست آوردن میزان پیچیدگی ترکیبات مختلف از مدل پیچیدگی کدگشا ارائه شده در فصل ۴ نیز بهره می‌گیرد. کارائی این کدکننده با روش متداول با در نظر گرفتن محدودیت پیچیدگی در سناریوهای مختلفی مقایسه شده است. در فصل آتی کدکننده مطلع از محدودیتهای گیرنده برای کاربردهای تطبیق را طراحی کرده‌ایم که در طراحی آن از کدکننده مطلع از محدودیتهای گیرنده که در این فصل ارائه شد استفاده خواهیم کرد.

فصل ششم

طراحی کدکننده مطلع از محدودیت‌های گیرنده برای کاربردهای تطبیق – نمونه مطالعاتی

در این فصل یک نمونه مطالعاتی برای کدکننده مطلع از محدودیتهای گیرنده برای کاربردهای تطبیق، طراحی خواهیم نمود. از آنجا که در نظر گرفتن ترکیبات مختلفی از محدودیت‌های گیرنده، تعداد گیرنده‌ها و روش تطبیق ممکن نمی‌باشد، لذا در این فصل تلاش می‌شود عملی بودن روش پیشنهادی در بخشهای قبل را با حل کامل مسئله برای زیر مجموعه‌ای از محدودیت‌ها و روش تطبیق نشان دهیم. برخی از مفروضات این نمونه مطالعاتی عبارتند از:

- در نظر گرفتن نرخ بیت (R) و توان مصرفی(C) بعنوان محدودیت‌های گیرنده‌ها
- استفاده از روش MV-Reuse بعنوان عملیات تطبیق

همانطور که پیشتر نیز اشاره شد، توان مصرفی گیرنده بدلیل فراگیر شدن کاربردهای چندرسانه‌ای در ابزارهای قابل حمل، از اهمیت قابل توجهی برخوردار است. روش تطبیق MV-Reuse نیز بدلیل متداول بودن آن و همچنین امکان مدل سازی آسان به کار گرفته شده است. به عبارت دیگر در MV-Reuse، عملا پارامتر چندی سازی برای هر گیرنده تابعی از پارامتر چندی سازی گیرنده پایه (Receiver 0) خواهد بود.

با در نظر گرفتن مفروضات بالا رابطه (۳-۵) را بصورت زیر خواهیم داشت.

$$\begin{cases} \text{Min D} = (\alpha_0 D_0 + \alpha_1 D_1 + \alpha_2 D_2 + \cdots + \alpha_k D_{\text{k}}), \\ \text{s.t. } R_0 < R_{C_0}, C_0 < C_{C_0}, \\ \text{s.t. } R_1 < R_{C_1}, C_0 < C_{C_1}, \\ \vdots \\ \vdots \\ \text{s.t. } R_n < R_{C_n}, C_n < C_{C_n}, \end{cases} \quad (۱-۶)$$

در رابطه (۶-۱)، D_{i} اعوجاج گیرنده‌های مختلف، R_{i} و C_i، به ترتیب نرخ بیت و توان مصرفی گیرنده‌های مختلف می‌باشند. همچنین R_{C_i} و C_{C_i} به ترتیب میزان نرخ بیت و توان مصرفی هدف می‌باشند.

این مسئله به مسئله آمده در (۶-۲) قابل تبدیل است که بایستی پارامترهای لاگرانژ $\lambda_{\text{R}_\text{i}}$ و $\lambda_{\text{C}_\text{i}}$ را برای آن بدست آورد.

$$Min(J) = \sum_{i=0}^{n} \alpha_{\text{i}} \left(D_{\text{i}} + \lambda_{\text{R}_\text{i}} R_i + \lambda_{\text{C}_\text{i}} C_i\right) \quad (۲-۶)$$

با حل این مسئله $2n$ پارامتر لاگرانژ خواهیم داشت که با جایگذاری آنها در رابطه (۶-۲) و به کار گیری این رابطه در مرحله انتخاب مد، می‌توان مصالحه بین محدودیت گیرنده‌های مختلف و مجموع اعوجاج را برآورده کرد.

۶-۱ پیاده سازی مدل انتزاعی برای کدکننده H.264/AVC مطلع از محدودیت پیچیدگی سه گیرنده با کاربرد تطبیق چندی سازی

مشابه با فصل قبل و با دنبال کردن روش انتزاعی پیشنهادی آمده در فصل ۳، اقدام به طراحی یک کنترل کننده پیچیدگی خواهیم کرد که بتواند محدودیت توان چند کدگشا را با توجه به عملیات نقطه تطبیق، در حین کد کردن در نظر بگیرد. لازم به ذکر است که مشابه فصل قبل تنها پارامتر چندی سازی بعنوان پارامتر کدکننده مد نظر قرار می‌گیرد و روابط

۶-۱-۱ بدست آوردن مدل هر منبع

از آنجا که روش تطبیق را MV-Reuse در نظر گرفته‌ایم، ضریب واقعی چندی سازی برای گیرنده ۱ عملا حاصلضرب پارامتر چندی سازی برای گیرنده صفر و گیرنده ۱ می‌باشد. به این ترتیب روابط آمده در (۶-۳) برای اعوجاج متصور است.

$$\begin{cases} D_0 = f(Q_0) = {Q_0^2}/{3} \\ D_1 = f(Q_0, Q_1) = {(Q_0 Q_1)^2}/{3} \\ D_2 = f(Q_0, Q_2) = {(Q_0 Q_2)^2}/{3} \\ \vdots \\ \vdots \\ D_n = f(Q_0, Q_n) = {(Q_0 Q_n)^2}/{3} \end{cases} \quad (۳-۶)$$

به همین ترتیب برای نرخ بیت روابط آمده در (۶-۴) را داریم.

$$\begin{cases} R_0 = \text{alog}_2^{\frac{\text{b}}{D_0}} = \text{alog}_2^{\frac{3\text{b}}{Q_0^2}} \\ R_1 = \text{alog}_2^{\frac{\text{b}}{D_1}} = \text{alog}_2^{\frac{3\text{b}}{(Q_0Q_1)^2}} \\ R_2 = \text{alog}_2^{\frac{\text{b}}{D_2}} = \text{alog}_2^{\frac{3\text{b}}{(Q_0Q_2)^2}} \\ \vdots \\ \vdots \\ R_n = \text{alog}_2^{\frac{\text{b}}{D_n}} = \text{alog}_2^{\frac{3\text{b}}{(Q_0Q_n)^2}} \end{cases} \quad (۴-۶)$$

برای پیچیدگی نیز با توجه به رابطه بدست آمده در فصل چهارم، روابط آمده در (۶-۵) قابل تعریف است.

$$\begin{cases} C_0 = \dfrac{m}{Q_0} + p \\ C_1 = \dfrac{m}{Q_0Q_1} + p \\ C_2 = \dfrac{m}{Q_0Q_2} + p \\ \vdots \\ \vdots \\ C_n = \dfrac{m}{Q_0Q_n} + p \end{cases} \quad (۵-۶)$$

۶-۱-۲ انتخاب پارامترهای لاگرانژ

با جایگزین کردن مجموعه روابط (۶-۳)، (۶-۴) و (۶-۵) در رابطه (۶-۲) به رابطه (۶-۶) خواهیم رسید.

$$\alpha_0\left(Q_0^2/_3 + \lambda_{\text{R}_0}\text{alog}_2^{\frac{3\text{b}}{Q_0^2}} + \lambda_{\text{C}_0}\left(\frac{m}{Q_0} + p\right)\right) + \sum_{i=1}^{n}\alpha_i\left((Q_0Q_i)^2/_3 + \lambda_{\text{R}_i}\text{alog}_2^{\frac{3\text{b}}{(Q_0Q_i)^2}} + \lambda_{\text{C}_i}\left(\frac{m}{Q_0Q_i} + p\right)\right) \quad (۶-۶)$$

با حل این مسئله به روابط زیر خواهیم رسید.

$$\frac{\partial J}{\partial Q_0} = \alpha_0\left(\frac{2\ln(2)}{3}Q_0^3 - 2\lambda_{\text{R}_0}Q_0 - \text{m}\ln(2)\lambda_{\text{C}_0}\right) + \sum_{i=1}^{k}\alpha_i\left(\frac{2\ln(2)}{3}Q_0^3Q_i^2 - 2\lambda_{\text{R}_i}Q_0 - \frac{\text{m}\ln(2)\lambda_{\text{C}_i}}{Q_i}\right) \quad (۷-۶)$$

$$\frac{\partial J}{\partial Q_i} = \alpha_i\left(\frac{2\ln(2)}{3}Q_0^2Q_i^3 - 2\lambda_{\text{R}_i}Q_i - \frac{\text{m}\ln(2)\lambda_{\text{C}_i}}{Q_0}\right) = \alpha_i\left(\frac{2\ln(2)}{3}(Q_0Q_i)^3 - 2\lambda_{\text{R}_i}Q_iQ_0 - \text{m}\ln(2)\lambda_{\text{C}_i}\right) \quad (۸-۶)$$

با بدست آوردن مقدار $\frac{2\ln(2)}{3}Q_0^3Q_i^2$ از رابطه (۶-۸) و جایگذاری آن در رابطه (۶-۷)، مقدار $\frac{\partial J}{\partial Q_0}$ طبق رابطه (۶-۹) بدست خواهد آمد.

$$\frac{\partial J}{\partial Q_0} = \alpha_0\left(\frac{2\ln(2)}{3}Q_0^3 - 2\lambda_{\text{R}_0}Q_0 - \text{m}\ln(2)\lambda_{\text{C}_0}\right) \quad (۹-۶)$$

رابطه بدست آمده در (۶-۹) دقیقا مشابه با رابطه‌ای است که در فصل قبل برای کدکننده مطلع از محدودیت‌های گیرنده بدست آوردیم که در آن تنها مشخصات نرخ بیت و پیچیدگی یک گیرنده لحاظ شده بود. همانطور که در رابطه (۶-۸) نیز

مشاهده می‌شود، رابطه مشابهی نیز برای گیرنده‌های دیگر بدست آمده است. به این ترتیب پارامترهای لاگرانژی که برای کدکننده مطلع از محدودیت‌های گیرنده در فصل قبل ارائه شد، برای این کدکننده نیز قابل استفاده خواهد بود با این تفاوت که پارامترهای چندی سازی در این کدکننده همگی تابعی از پارامتر چندی سازی اولین گیرنده خواهند بود.

با داشتن پارامترهای لاگرانژ می‌توان مسئله بهینه سازی را در حین عملیات انتخاب مد پیاده سازی کرد. در ادامه این فصل کنترل کننده پیچیدگی را ارائه خواهیم کرد.

۶-۱-۳ طراحی کنترل کننده پیچیدگی

کنترل کننده در کدکننده مطلع از محدودیت‌های گیرنده برای کاربردهای تطبیق ($RAEA^2$) بسیار مشابه با کنترل کننده برای کدکننده مطلع از محدودیت‌های یک گیرنده (RAE) می‌باشد که در فصل پیش به آن پرداختیم. مهمترین تفاوت این کنترل کننده، در نظر گرفتن محدودیت‌های گیرنده‌های مختلف و به روز کردن آنها به صورت مستقل پس از کدکردن هر سطح می‌باشد. علاوه بر این، عملیات انتخاب مد نیز به نحوی تغییر می‌یابد که بتواند محدودیت‌های تعداد متنوعی گیرنده و مجموع اعوجاج تمامی‌گیرنده‌ها را مد نظر قرار دهد. شبه کد مربوط به کنترل کننده در لیست ۶-۱ آمده است.

در ابتدای عملیات و در سطح فریم، کنترل کننده پیچیدگی پارامترهای هر گیرنده را بصورت مستقل به روز می‌کند. به این ترتیب که توان قابل مصرف فریم جاری هر گیرنده (C^i_{Frame}) را با توجه به میزان توان باقی مانده از هر گیرنده (C^i_{Total}) و تعداد فریم‌های کد نشده بدست می‌آورد. در همین مرحله، توان مصرف شده توسط آخرین فریم در هر گیرنده ($C^{i\,Consumed}_{Frame}$) با توان موجود برای فریم فعلی در همان گیرنده (C^i_{Frame}) مقایسه می‌شود. در صورتیکه توان مصرف شده توسط آخرین فریم کد شده از توان موجود برای فریم فعلی بیشتر باشد، فیلتر بلوک زدائی در سطح فریم – برای گیرنده در حال بررسی – غیر فعال خواهد شد. به این ترتیب برای تمامی ماکروبلوکهای موجود در فریم جاری این گیرنده عملیات بلوک زدائی انجام نخواهد شد. در ادامه عملیات اختصاص نرخ بیت در سطح فریم صورت می‌پذیرد که مهمترین خروجی آن مقدار پارامتر چندی سازی برای هر گیرنده (QP^i) خواهد بود. پس از تعیین مقدار پارامتر QP^i توسط کنترل کننده نرخ بیت، این پارامتر توسط پروسه $AdjustQP^i$ تنظیم خواهد شد. رفتار این تابع با تابع $AdjustQP$ که در فصل ۶ معرفی شد، یکسان است. به عبارت دیگر، پروسه $AdjustQP^i$ عملا مقدار QP^i را با توجه به میزان توان مصرفی و توان موجود گیرنده مد نظر تغییر خواهد داد.

پس از اختصاص توان به فریم جاری در هر گیرنده، عملیات اختصاص توان به BU هر گیرنده (C^i_{BU}) صورت می‌پذیرد. پس از اختصاص توان به هر BU، عملیات کنترل نرخ بیت در سطح BU برای هر گیرنده صورت می‌گیرد. خروجی این مرحله عملا مقدار پارامتر چندی سازی گیرنده‌های متفاوت (QP^i) خواهد بود. مشابه با بخش مربوط به اختصاص توان به فریم هر گیرنده، پس از تعیین مقدار پارامتر QP^i توسط کنترل کننده نرخ بیت، این پارامتر توسط پروسه $AdjustQP^i$ نیز تنظیم خواهد شد. با تنظیم پارامتر QP^i در سطح BU برای هر گیرنده، عملیات کدکردن در سطح ماکروبلوک شروع خواهد شد. در این مرحله، عملیات بهینه‌سازی نرخ بیت و اعوجاج برای مدهای مختلف صورت خواهد پذیرفت. برای این کار ابتدا میزان اعوجاج، نرخ بیت و پیچیدگی هر گیرنده به ازای مد فعلی ($D_i(k), R_i(k), C_i(k)$) استخراج می‌شود. همچنین با توجه به مقدار پارامتر چندی سازی به ازای هر گیرنده (QP^i)، پارامترهای λ_{R_i} و λ_{C_i} مربوطه نیز مقدار خواهند گرفت. با داشتن نرخ

بیت، مصرف توان و اعوجاج هر مد در هر گیرنده، می‌توان هزینه توامان نرخ بیت، پیچیدگی و اعوجاج مد فعلی (J_k) را محاسبه نمود. واضح است که از بین مدهای مختلف، مدی انتخاب می‌شود که منجر به J_k کمینه (J_k^{Min}) گردد. پس از انتخاب مد بهینه و به روز کردن میزان توان مصرفی توسط ماکروبلوک حاضر در هر گیرنده ($C_{MB}^{iConsumed}$)، میزان توان مصرفی BU هر گیرنده ($C_{BU}^{iConsumed}$) بروز می‌گردد. عملیات مذکور برای تمامی BUهای یک فریم تکرار خواهد شد. پس از اتمام کدکردن یک فریم، میزان توان باقیمانده از هر گیرنده (C_{Total}^{i}) با توجه به میزان توان مصرفی توسط فریم حاضر در همان گیرنده ($C_{Frame}^{Consumed}$) به روز خواهد شد و عملیات کنترل پیچیدگی بر روی فریم بعدی ادامه خواهد یافت.

For (each frame)
{
For (each Receiver)
{
$C_{Frame}^{i} = \frac{C_{Total}^{i}}{Number\ of\ remaining\ frames\ in\ sequence}$
If $(C_{Frame}^{i^{Consumed}} > C_{Frame}^{i})$
Disable DBF for current frame of receiver(i)
Perform frame level RateControl for receiver(i)
AdjustQP^{i}
}
For (each BasicUnit)
{
For (each Receiver)
{
$C_{BU}^{i} = \frac{C_{Frame}^{i}}{Number\ of\ Basic\ Units\ in\ frame}$
Perform BasicUnit level RateControl of receiver(i)
AdjustQP^{i} *for receiver(i)*
}
For (each MB)
{
For (each mode)
{
For (each Receiver)
{
Calculate $D_i(k), R_i(k), C_i(k)$
Assign λ_{R_i} *and* λ_{C_i} *based on* QP^i *value*
}
$J_k = \sum_{i=0}^{n} \alpha_i \left(D_i(k) + \lambda_{R_i} R_i(k) + \lambda_{C_i} C_i(k)\right)$
Update J_k^{Min} *& best mode*
}
For (each Receiver)
{
Update $C_{MB}^{i^{Consumed}}$
$C_{BU}^{i^{Consumed}} += C_{MB}^{i^{Consumed}}$
}
}
For (each Receiver)
$C_{Frame}^{i^{Consumed}} += C_{BU}^{i^{Consumed}}$
}
For (each Receiver)
$C_{Total}^{i} -= C_{Frame}^{i^{Consumed}}$
}

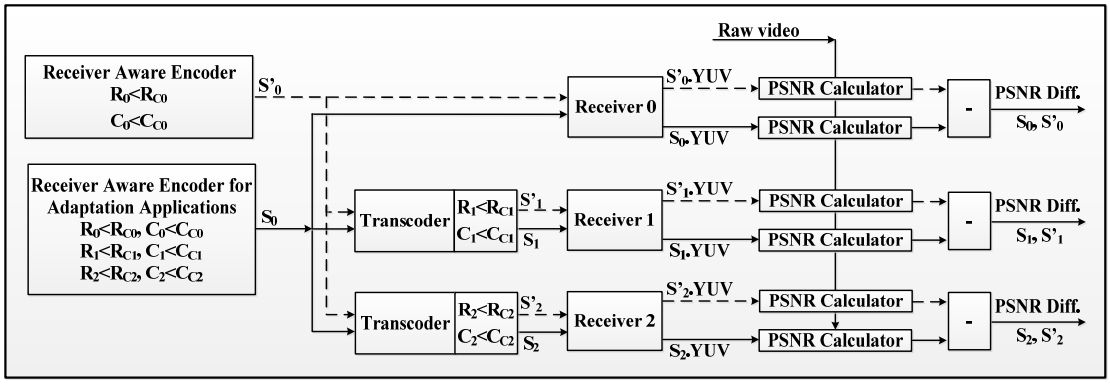
لیست ۶-۱ شبه‌کد کنترل کننده پیچیدگی برای کدکننده مطلع از محدودیت‌های گیرنده برای کاربردهای تطبیق

۶-۲ نتایج شبیه‌سازی

در این بخش به تشریح نحوه شبیه‌سازی عملکرد کدکننده مطلع از محدودیت‌های گیرنده برای کاربردهای تطبیق خواهیم پرداخت. برای آنکه انجام شبیه‌سازی عملی باشد علاوه بر فرضیات بخش قبل، یعنی در نظر گرفتن پیچیدگی و نرخ بیت بعنوان محدودیت‌های گیرنده و همچنین استفاده از روش MV-Reuse بعنوان روش تطبیق، فرضیات زیر را نیز در نظر گرفته‌ایم:

- مشارکت گیرنده‌های مختلف را یکسان در نظر گرفته‌ایم (به عبارت دیگر $(\forall i;\ 0 \le i \le n;\ \alpha_i = 1)$.
- بجای n گیرنده تنها سه گیرنده مد نظر قرار گرفته‌ایم.

ساختار کلی روش شبیه‌سازی در شکل ۶-۱ آمده است. همان گونه که در این شکل نشان داده شده است، برای ارزیابی روش پیشنهادی بایستی دو نمونه شبیه‌سازی صورت پذیرد. در یک شبیه‌سازی عملا یک کدکننده مطلع از محدودیت‌های گیرنده (RAE) را به کار خواهیم گرفت که تنها محدودیت‌های یک گیرنده (Receiver 0) را مد نظر قرار می‌دهد. رشته بیت تولید شده توسط این کدکننده (S'_0) به نقاط تطبیق و گیرنده‌های مختلف ارسال شده و در صورت لزوم با توجه به محدودیت گیرنده تطبیق داده خواهد شد. پس از انجام عملیات تطبیق این رشته بیتها برای گیرنده‌های مربوطه شان ارسال خواهند شد. در شبیه‌سازی دوم، با استفاده از کدکننده مطلع از محدودیت‌های گیرنده‌ای که در این فصل معرفی شد ($RAEA^2$)، عملیات فشرده‌سازی صورت می‌پذیرد. لازم به ذکر است که در این کدکننده، مشخصات هر سه گیرنده در حین کدشدن مد نظر قرار گرفته است تا رشته بیت S_0 را تولید کند. مشابه با شبیه‌سازی اول، رشته بیت S_0 نیز به نقاط تطبیق مختلف ارسال می‌شود تا با توجه به محدودیت‌های آنها تطبیق داده شده و به گیرنده متناظرش ارسال گردد. برای مقایسه این دو روش و ارزیابی عملکرد کدکننده پیشنهادی در این فصل، رشته بیتهای کدگشایی شده توسط این دو روش با ویدئو خام مقایسه شده و PSNR هر یک استخراج می‌شود. نهایتا تفاوت PSNRها را محاسبه خواهیم کرد.



شکل ۶-۱ ساختار کلی برای شبیه‌سازی

از آنجا که در شکل ۶-۱ و توضیحات مربوط به آن بصورت کلی مراحل شبیه‌سازی را تشریح نمودیم در ادامه سعی می‌کنیم با معرفی یک سناریوی شبیه‌سازی، به جزئیات بیشتری در این زمینه بپردازیم.

۶-۲-۱ سناریوی شبیه‌سازی

برای انجام شبیه‌سازی با در نظر گرفتن ساختار کلی معرفی شده در شکل ۶-۱ مراحل زیر را دنبال می‌کنیم:

۱. در ابتدا ویدئو نمونه خام را با یک کدکننده معمولی و با نرخ بیتهای متنوع R_0، R_1 و R_2 فشرده‌سازی کرده و سپس آنها را کدگشایی می‌کنیم تا به ازای هر نرخ بیت توان مصرفی مربوطه استخراج شود. این توانهای مصرفی برای R_0، R_1 و R_2 را به ترتیب C_0، C_1 و C_2 می‌نامیم.

برای اینکه در هر نرخ بیت رفتار کدکننده‌های RAE و $RAEA^2$ را در توانهای متنوعی ارزیابی کنیم، در هر نرخ بیت سه سطح پیچیدگی بالا (H)، متوسط (M) و پایین (L) در نظر می‌گیریم. به این ترتیب به ازای هر یک از نرخ بیتهای R_0، R_1 و R_2 سه سطح پیچیدگی طبق رابطه زیر استخراج می‌کنیم.

$$\begin{cases} C_i^H = H \times C_i \\ C_i^M = M \times C_i, i = 1..3 \\ C_i^L = L \times C_i \end{cases} \quad (۱۰-۶)$$

در شبیه‌سازیهایی که در ادامه انجام شده است مقادیر H=0.9، M = 0.7 و L = 0.5 در نظر گرفته شده است.

۲. با داشتن سه سطح پیچیدگی برای هر نرخ بیت ترکیبات متنوعی از نرخ بیت و پیچیدگی ایجاد می‌شود که در جدول ۶-۱ نشان داده شده‌اند. در این مرحله ویدئو نمونه را با استفاده از کدکننده مطلع از محدودیت گیرنده برای کاربردهای تطبیق ($RAEA^2$)، به دفعات مختلف و هر مرتبه با یکی از سطوح پیچیدگی تعریف شده در جدول ۶-۱ فشرده خواهیم کرد.

۳. هر یک از رشته بیتهای تولید شده از مرحله ۲ را ۹ مرتبه تطبیق می‌کنیم. به عبارت دیگر هر رشته بیت **S0_Level1_Level2_Level3** هربار به یکی از سطوح پیچیدگی ($C_i^{Level(i)}$) و در هر سطح پیچیدگی به نرخ بیتهای R_i، $R_i-\Delta_1$ و $R_i-\Delta_2$ (i=1..3) تطبیق داده خواهد شد. بعنوان مثال، برای رشته بیت تولید شده برای سطح ۲۷ مجموعه عملیات تطبیق آمده در جدول ۶-۲ بایستی صورت پذیرد.

۴. در این مرحله، همانطور که در شکل ۶-۱ مربوط به ساختار کلی شبیه‌سازیها آمده است PSNR بین هر یک از رشته بیتهای تولید شده در مرحله ۳ و ویدئو نمونه بدست خواهد آمد.

۵. از آنجا که تاکنون تنها با کدکننده مطلع از محدودیت‌های گیرنده برای کاربردهای تطبیق ($RAEA^2$) شبیه‌سازی انجام داده‌ایم، در این مرحله بایستی شبیه‌سازی‌ها را برای کدکننده مطلع از محدودیت‌های گیرنده (RAE) تکرار نمائیم. برای این منظور، مرحله ۲ را این بار با کدکننده مطلع از محدودیت گیرنده‌ای تکرار می‌کنیم که تنها محدودیت نرخ بیت R_0 و سطوح پیچیدگی مربوط به آن را در حین کدکردن مد نظر قرار می‌دهد. به عبارت دیگر از جدول ۶-۱ تنها ستون سوم برای پیچیدگی لحاظ می‌شود و نرخ بیت نیز برابر با R_0 خواهد بود.

۶. در ادامه، برای رشته بیتهای حاصل از مرحله ۵، عملیات مرحله ۳ را تکرار می‌کنیم.

۷. PSNR بین هر یک از رشته بیتهای تولید شده در مرحله ۶ و ویدئو نمونه بدست خواهد آمد.

جدول ۶-۱ سطوح مختلف پیچیدگی قابل تعریف برای نرخ بیتهای مختلف

نام رشته بیت	سطح پیچیدگی	پیچیدگی برای نرخ بیت R_0	پیچیدگی برای نرخ بیت R_1	پیچیدگی برای نرخ بیت R_2
S0_H_H_H	۲۷	C_1^H	C_2^H	C_3^H
S0_H_H_M	۲۶			C_3^M
S0_H_H_L	۲۵			C_3^L
S0_H_M_H	۲۴		C_2^M	C_3^H
S0_H_M_M	۲۳			C_3^M
S0_H_M_L	۲۲			C_3^L
S0_H_L_H	۲۱		C_2^L	C_3^H
S0_H_L_M	۲۰			C_3^M
S0_H_L_L	۱۹			C_3^L
S0_M_H_H	۱۸	C_1^M	C_2^H	C_3^H
S0_M_H_M	۱۷			C_3^M
S0_M_H_L	۱۶			C_3^L
S0_M_M_H	۱۵		C_2^M	C_3^H
S0_M_M_M	۱۴			C_3^M
S0_M_M_L	۱۳			C_3^L
S0_M_L_H	۱۲		C_2^L	C_3^H
S0_M_L_M	۱۱			C_3^M
S0_M_L_L	۱۰			C_3^L
S0_L_H_H	۹	C_1^L	C_2^H	C_3^H
S0_L_H_M	۸			C_3^M
S0_L_H_L	۷			C_3^L
S0_L_M_H	۶		C_2^M	C_3^H
S0_L_M_M	۵			C_3^M
S0_L_M_L	۴			C_3^L
S0_L_L_H	۳		C_2^L	C_3^H
S0_L_L_M	۲			C_3^M
S0_L_L_L	۱			C_3^L

جدول ۶-۲ مجموعه ترکیبات پیچیدگی و نرخ بیتهای تطبیق برای رشته بیت تولید شده با سطح پیچیدگی ۲۷

سطح پیچیدگی	نرخ بیت
C_1^H	
	I
C_2^H	
	/
C_3^H	
	/

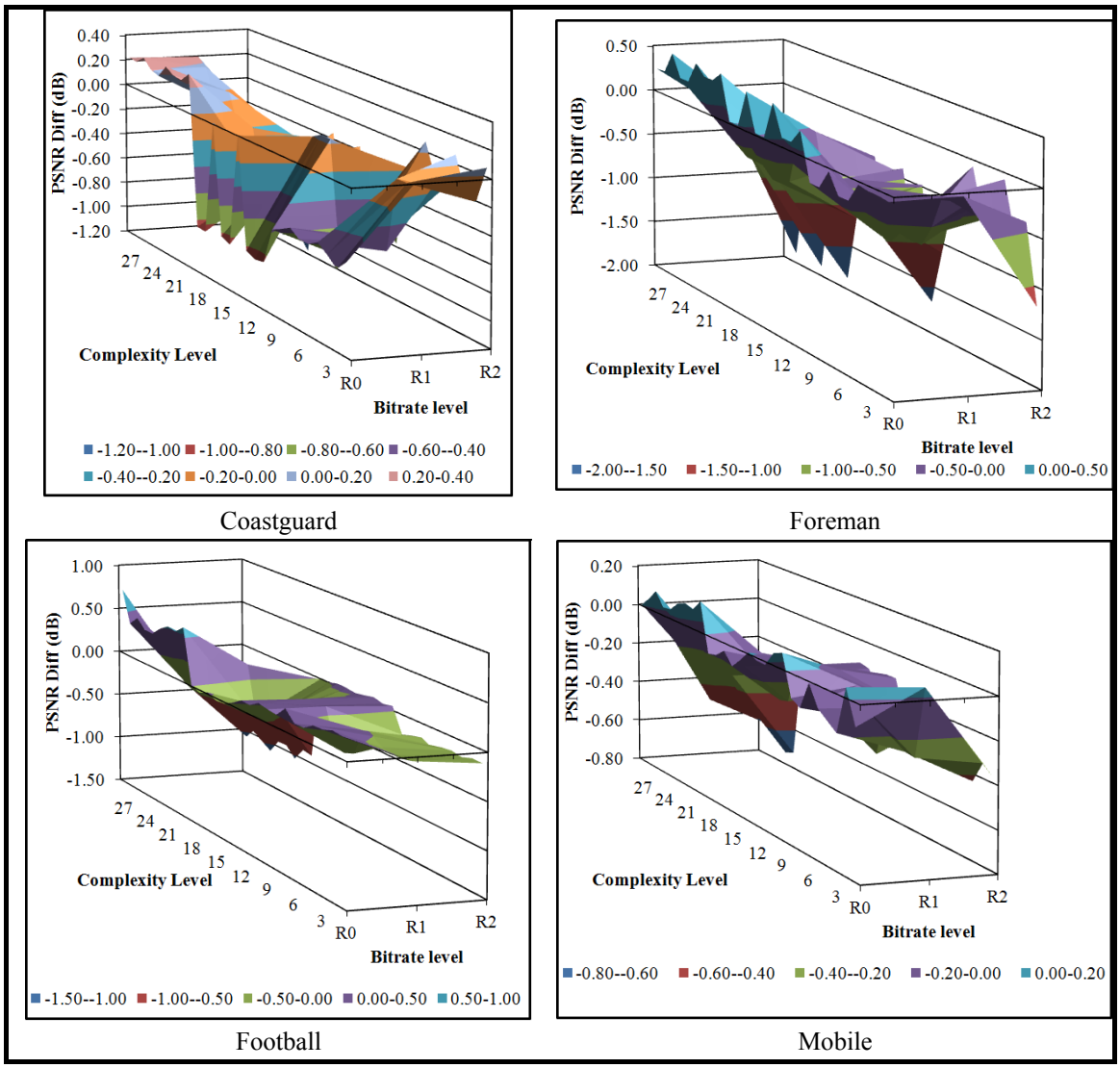
۸. برای هر یک از سطوح پیچیدگی، از بین نرخ بیتهای مختلف (R_i، $R_i-\Delta_1$ و $R_i-\Delta_2$) نرخ بیتی را انتخاب می‌کنیم که مقدار گزارش شده نرخ بیت آن در دو کدکننده RAE و $RAEA^2$، نزدیکترین باشد. به این ترتیب به ازای هر یک از ۲۷ سطح پیچیدگی تنها یک نرخ بیت استخراج خواهد شد که کارائی روش پیشنهادی در مقایسه با روش RAE – با استفاده از نتایج مراحل ۴ و ۷ – قابل ارزیابی خواهد بود.

با اجرا کردن مراحل فوق بر روی ویدئوهای نمونه مختلف کارائی کدکننده $RAEA^2$ را ارزیابی کرده ایم. در شبیه‌سازی‌های انجام گرفته، تنظیمات آمده در جدول ۶-۳ در حین کدکردن بکار گرفته شده است.

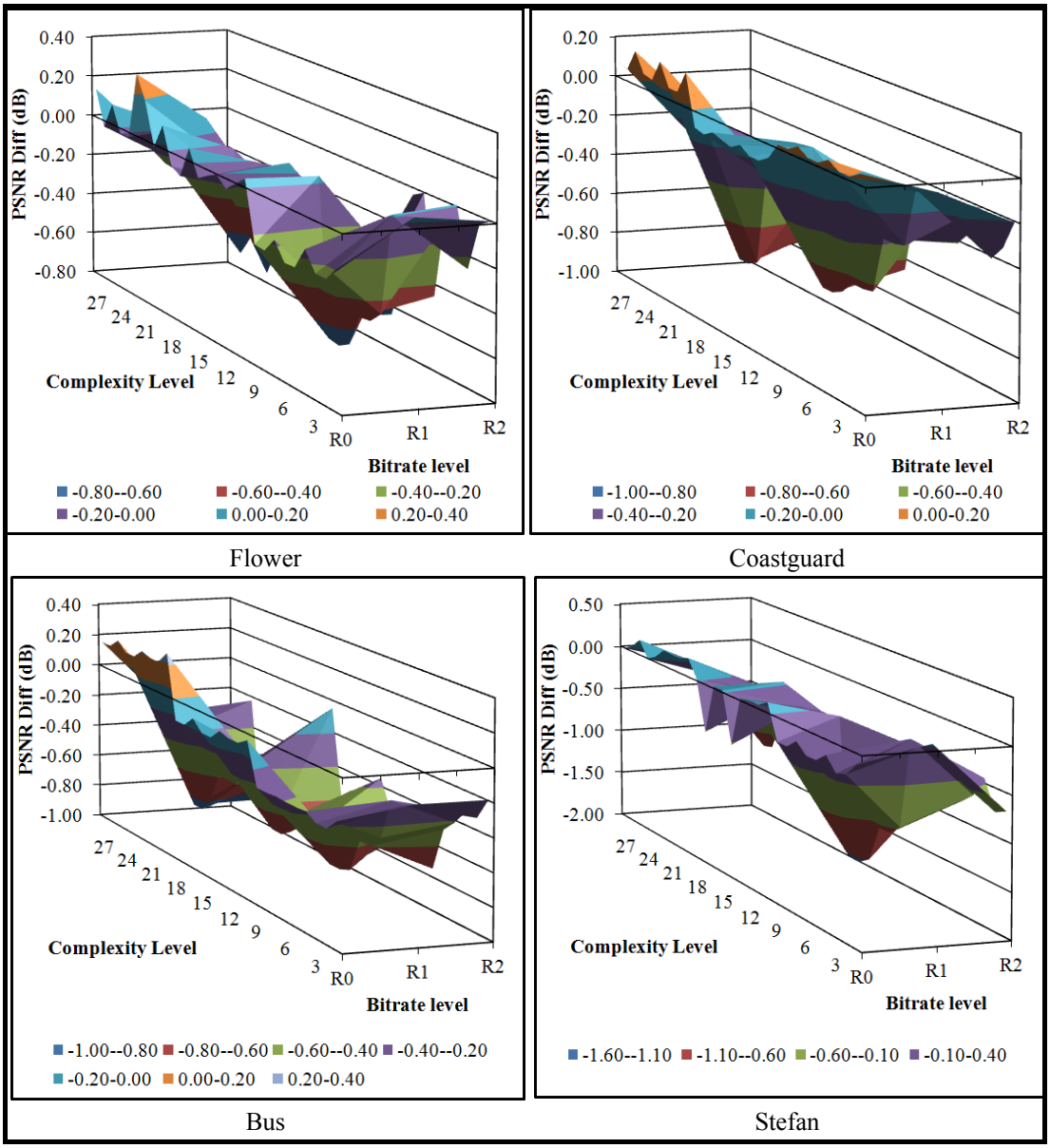
جدول ۶-۳ تنظیمات کدکننده H.264/AVC

پروفایل	Baseline
سطح	۳
تعداد فریم‌های کد شده	۱۰۰
تعداد فریم‌های مرجع	۱
محدوده جستجو	۱۶
بهینه ساز نرخ – اعوجاج (RDO)	روشن
کنترل کننده نرخ بیت	روشن
دقت تخمین حرکت	¼ پیکسل

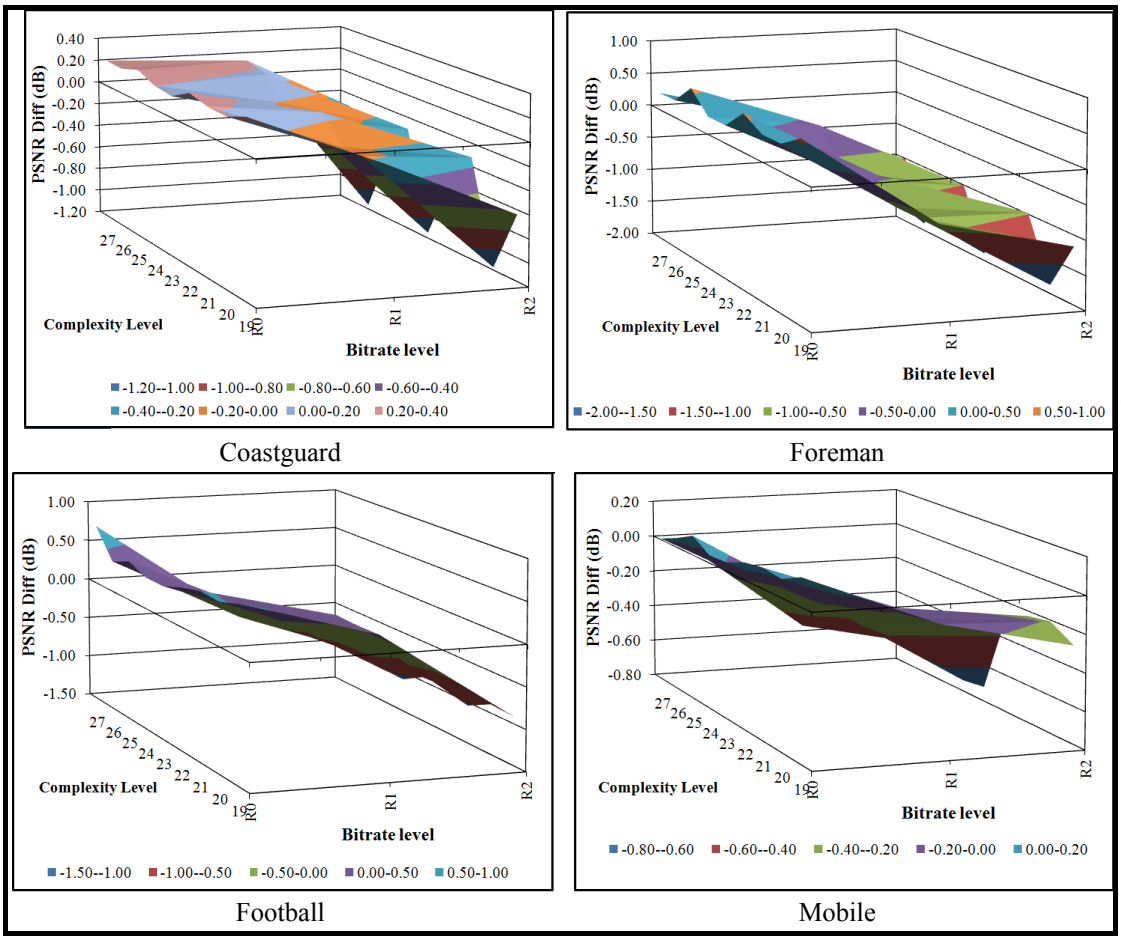
نتایج شبیه‌سازی برای تعدادی از ویدئوهای نمونه در شکل ۶-۲ و شکل ۶-۳ آمده است. در این شکل‌ها، تفاضل کیفیت بین RAE و $RAEA^2$ گزارش شده است. به این ترتیب اعداد مثبت نمایانگر عملکرد بهتر RAE و اعداد منفی نشانگر عملکرد بهتر $RAEA^2$ می‌باشد. همانطور که در شکل‌ها مشاهده می‌شود، برای تمامی ویدئوهای شبیه‌سازی شده، کدکننده RAE عملکرد بهتری در نرخ بیتهای پایه (R_0) دارد. این رفتار نیز قابل انتظار بود، چرا که در کدکننده مطلع از محدودیت‌های گیرنده برای کاربردهای تطبیق علاوه بر محدودیت گیرنده با بالاترین نرخ بیت، محدودیت گیرنده‌های دیگر نیز مد نظر قرار می‌گیرند. این مسئله باعث می‌شود که در حین عملیات انتخاب مد، مدهایی انتخاب شوند که محدودیت تمامی‌گیرنده‌ها را لحاظ کنند نه تنها یک گیرنده. این در حالی است که در کدکننده RAE، تنها محدودیت‌های یک گیرنده لحاظ می‌شود و واضح است که بهترین مدها برای یک گیرنده خاص انتخاب خواهند شد و به این ترتیب کیفیت بهتری بدست خواهد آمد. با کاهش نرخ بیت، از آنجا که کد کننده RAE، هیچ یک از محدودیت‌های گیرنده‌های دیگر را در نظر نگرفته است، در مجموع عملکرد بدتری نسبت به کدکننده $RAEA^2$ برای تمامی ویدئوهای شبیه‌سازی شده دارد. همانطور که نتایج شبیه‌سازی نشان می‌دهد، کدکننده $RAEA^2$ تا ۱/۸۹ دسیبل از کدکننده RAE بهتر کار می‌کند. به طور میانگین کدکننده پیشنهادی کیفیت را تا ۰/۴ دسیبل بهبود می‌بخشد. برای واضح‌تر شدن روند تغییر کیفیت عملکرد این دو روش در شکل ۶-۴ و شکل ۶-۵، تنها سطوح ۲۷ تا ۱۹ را نشان داده‌ایم. در تمامی نمودارها با کاهش نرخ بیت، یک روند نزولی در تفاضل کیفیت RAE در مقایسه با روش $RAEA^2$، مشاهده می‌شود که نشانگر عملکرد بهتر روش پیشنهادی است.



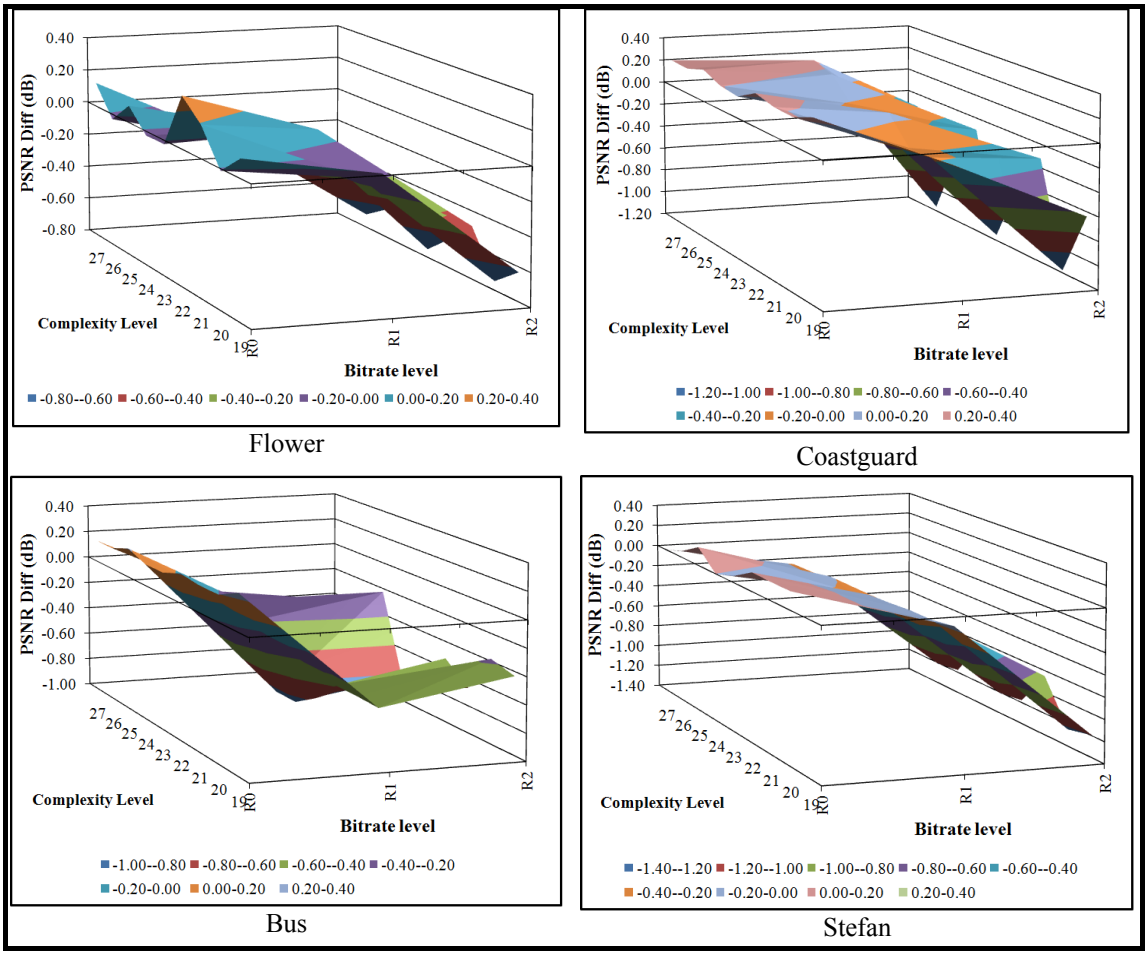
شکل ۶-۲ تفاوت عملکرد کدکننده $RAEA^2$ و RAE در سطوح ۲۷ گانه، برای ویدئوهای نمونه با اندازه QCIF



شکل ۶-۳ تفاوت عملکرد کدکننده RAEA[2] و RAE در سطوح ۲۷ گانه، برای ویدئوهای نمونه با اندازه CIF



شکل ۶-۴ تفاوت عملکرد کدکننده $RAEA^2$ و RAE در ۸ سطح، برای ویدئوهای نمونه با اندازه QCIF



شکل ۶-۵ تفاوت عملکرد کدکننده RAEA[2] و RAE در ۸ سطح، برای ویدئوهای نمونه با اندازه CIF

۶-۳ جمع بندی

در این فصل یک نمونه مطالعاتی برای کدکننده مطلع از محدودیتهای گیرنده برای کاربردهای تطبیق با دنبال کردن روش انتزاعی پیشنهادی در فصل ۳، ارائه شد. برای این کدکننده مشابه با فصل قبل محدودیتهای پیچیدگی و نرخ بیت لحاظ شده است. برای ارزیابی نمونه مطالعاتی نیز تنها ۳ گیرنده با مشارکتهای یکسان لحاظ شده‌اند. روش پیشنهادی در ۲۷ سطح مختلف پیچیدگی ارزیابی شده است و در این ارزیابیها با کدکننده مطلع از محدودیتهای گیرنده مقایسه شده است.

در فصل آینده نمونه مطالعاتی دیگری ارائه می‌کنیم که در آن کدکننده مطلع از محدودیتهای خویش طراحی خواهد شد.

فصل هفتم

طراحی کدکننده مطلع از محدودیتهای منابع خویش– نمونه مطالعاتی

در دو فصل قبل به محدودیت‌های گیرنده پرداختیم این در حالی است که روش انتزاعی پیشنهادی قابل اعمال به کدکننده نیز می‌باشد. در این فصل یک کدکننده مطلع از محدودیتهای خویش طراحی خواهیم کرد که علاوه بر نرخ بیت محدودیت پیچیدگی را نیز دارد. با داشتن محدودیت توان و نرخ بیت رابطه (۳-۳) بصورت زیر در می‌آید:

$$\begin{cases} \text{Min}\,(D), \\ \text{s.t. } C < C_c, R < R_c, \end{cases} \quad (۱-۷)$$

در این رابطه، D نشانگر اعوجاج کل، C نمایانگر پیچیدگی کل کدکننده، C_c پیچیدگی هدف برای کدکننده، R نرخ بیت نهایی و R_c نرخ بیت هدف می‌باشند.

در استاندارد H.264/AVC عملیات انتخاب مد و تخمین حرکت بیشترین پیچیدگی را به خود اختصاص می‌دهند. از بین واحد تخمین حرکت و انتخاب مد، واحد تخمین حرکت از اهمیت بیشتری برخوردار است، چرا که موثرترین و در عین حال پرهزینه ترین قسمت در استاندارد H.264/AVC می‌باشد. این واحد که شامل مراحل تخمین حرکت صحیح و تخمین حرکت اعشاری می‌باشد، در حالت جستجوی کامل حدود ۶۰ تا ۸۰ درصد [19] و در حالت جستجوی سریع حدود ۵۰ درصد کل محاسبات کدکننده را به خود اختصاص می‌دهد [12]. بنابراین برای این نمونه مطالعاتی واحد تخمین حرکت را برای دو حالت جستجوی کامل و سریع، هر دو شامل تخمین حرکت صحیح و اعشاری، در نظر خواهیم گرفت.

۷-۱ پیاده سازی مدل انتزاعی برای واحد تخمین حرکت کدکننده H.264/AVC مطلع از محدودیت‌های خویش

برای طراحی کدکننده مطلع از محدودیتهای خویش، طبق روند انتزاعی آمده در فصل ۳ اقدام به انتخاب پارامترهای کدکننده، استخراج ترکیبات بهینه، بدست آوردن پارامترهای لاگرانژ و طراحی کنترل کننده خواهیم کرد.

۷-۱-۱ انتخاب پارامترهای کدکننده

بر اساس روند انتخاب پارامترها، در اولین قدم کل پارامترهای کدکردن مربوط به واحد تخمین حرکت، که روی پیچیدگی تاثیر می‌گذارند، را لیست می‌کنیم که در لیست ۷-۱ آمده است.

- دقت بردار حرکت
- محدوده جستجو
- تعداد مدهای فعال Inter
- تعداد فریم‌های مرجع
- معیار مقایسه

لیست ۷-۱ پارامترهای کدکردن مربوط به واحد تخمین حرکت

لازم به ذکر است که با توجه به الگوریتم تخمین حرکت و بر اساس نتایج شبیه‌سازیهای انجام شده، پارامترهای دیگر نظیر مدهای Intra و Skip، همچنین پارامتر کوانتیزاسیون تاثیر مستقیمی بر پیچیدگی واحد تخمین حرکت ندارند و به همین دلیل در لیست ۷-۱ نیامده‌اند. البته برخی از این پارامترهای بعدا در بخش ۷-۱-۵-۱ و ۷-۱-۵-۲، که در آنها کنترل کننده پیچیدگی را طراحی خواهیم کرد، لحاظ خواهند شد.

بر اساس روند انتخاب پارامترها، تاثیر هر یک از پارامترهای آمده در لیست ۷-۱ بر پیچیدگی و کیفیت روش‌های تخمین حرکت با جستجوی سریع و جستجوی کامل با انجام مجموعه‌ای از شبیه‌سازیها بدست آمده‌اند. در این شبیه‌سازیها، ۱۶ ویدئو نمونه در اندازه‌های مختلف QCIF، CIF و 4CIF و برای تمامی محدوده قابل تعریف هر پارامتر و برای نرخ بیتهای متنوع کد شده‌اند. میانگین نتایج شبیه‌سازیها در جدول ۷-۱ آمده است. با توجه به محدودیت فضا، برای هر پارامتر تنها زیر مجموعه‌ای از مقدار پارامترها گزارش شده است. در این جدول، کیفیت با معیار PSNR و پیچیدگی با معیار تعداد پالس ساعت و با استفاده از نرم افزار VTune [69] استخراج شده‌اند. تنظیمات کدکننده در این آزمایش به ترتیب آمده در جدول ۷-۲ است.

جدول ۷-۱ درصد افزایش پیچیدگی (CI) و میزان افزایش کیفیت (QI) (dB) برای پارامترهای مختلف و ویدئوهای متفاوت

پارامترهای کدکردن	مقدار پارامترها	تخمین حرکت با جستجوی سریع						تخمین حرکت با جستجوی کامل					
		QCIF		CIF		4CIF		QCIF		CIF		4CIF	
		QI	CI	QI	CI	QI	CI	QI	CI	QI	CI	QI	CI
تعداد فریم‌های مرجع	۱	-	-	-	-	-	-	-	-	-	-	-	-
	۲	۰,۲۷	۹۳	۰,۲۷	۱۰۰	۰,۱۷	۹۹	۰,۳۲	۹۷	۰,۲۷	۹۹	۰,۱۶	۹۸
	۳	۰,۵۸	۱۸۵	۰,۴۰	۲۰۱	۰,۲۰	۲۰۱	۰,۶۳	۲۰۲	۰,۴۱	۱۹۷	۰,۲۰	۱۹۶
	۴	۰,۶۴	۲۸۴	۰,۴۹	۳۰۳	۰,۲۲	۳۰۵	۰,۷۰	۲۹۴	۰,۵۱	۲۹۵	۰,۲۲	۲۹۴
	۵	۰,۷۷	۳۸۲	۰,۵۶	۴۰۵	۰,۲۳	۴۱۲	۰,۸۰	۳۷۷	۰,۵۵	۳۹۲	۰,۲۴	۳۹۱
دقت عملیات تخمین حرکت	Full pel.	-	-	-	-	-	-	-	-	-	-	-	-
	Half pel.	۰,۶۷	۹۵	۱,۱۸	۹۲	۱,۱۱	۱۰۲	۰,۸۲	۹	۱,۳۵	۱۱	۱,۰۵	۱۲
	Quarter pel.	۱,۵۷	۱۸۲	۲,۲۷	۱۵۴	۱,۹۸	۲۰۱	۱,۵۴	۲۰	۲,۳۸	۲۳	۱,۸۲	۲۳
محدوده جستجو	۱	-	-	-	-	-	-	-	-	-	-	-	-
	۲	۰,۳۶	۱۸	۱,۰۳	۲۰	۰,۰۹	۲۱	۰,۲۱	۱۱	۰,۴۶	۱۲	۰,۲۶	۱۴
	۳	۰,۴۹	۲۹	۱,۴۲	۳۲	۰,۱۱	۲۸	۰,۲۹	۲۷	۰,۷۷	۳۰	۰,۴۶	۳۴
	۴	۰,۵۳	۳۴	۱,۶۰	۳۸	۰,۳۰	۳۴	۰,۳۵	۴۹	۰,۹۳	۵۴	۰,۵۳	۶۰
	۵	۰,۵۶	۳۷	۱,۶۸	۴۱	۰,۴۱	۳۷	۰,۳۷	۷۶	۱,۰۸	۸۳	۰,۵۸	۹۲
مدهای بین فعال Inter	P8x8	-	-	-	-	-	-	-	-	-	-	-	-
	16x16 – P8x8	۰,۲۲	۲۳	۰,۵۴	۲۳	۰,۸۹	۴۱	۰,۲۱	۱	۰,۵۱	۱	۰,۸۵	۱
	16x16 – 16x8 – P8x8	۰,۲۳	۲۹	۰,۶۱	۳۸	۰,۹۵	۹۳	۰,۲۴	۲	۰,۵۷	۲	۰,۹۳	۲
	All inter modes	۰,۲۸	۵۹	۰,۶۷	۶۱	۱,۰۵	۱۸۱	۰,۲۸	۴	۰,۶۱	۳	۱,۰۱	۴
معیار مقایسه	SAD	-	-	-	-	-	-	-	-	-	-	-	-
	SSE	۰,۰۴	۷	۰,۰۳	۸	−۰,۰۱	۸	۰,۰۲	۴	۰,۰۱	۳	−۰,۰,۲	۲

جدول ۷-۲ تنظیمات کدکننده H.264/AVC برای شبیه‌سازیهای انجام شده

پارامتر	مقدار
پروفایل	Baseline
سطح	۳
تعداد فریم کدشده	۱۰۰
نرخ فریم	۳۰
اندازه GOP	۱۵
روش RDO	روشن – خاموش
کنترل کننده نرخ بیت	روشن
کدکننده آنتروپی	CAVLC

با توجه به تنوع محتوای ویدئوهای مختلف و برای آنکه ارزیابی پارامترها را برای ویدئوهای نمونه با کیفیتی قابل قبول انجام دهیم، تلاش کرده‌ایم که کدکننده را به نحوی تنظیم کنیم که برای تمامی ویدئوها کیفیت یکسانی به دست دهد. به عبارت دیگر، برای هر ویدئو نمونه نرخ بیتی انتخاب شده است که منجر به کیفیتی در محدوده ۳۲ دسیبل گردد.

در شبیه‌سازیهای آمده در جدول ۷-۱، هنگام مطالعه رفتار یک پارامتر مقادیر پارامترهای دیگر همگی به بیشترین مقدار قابل تعریف در استاندارد ثابت شده است. به این ترتیب سعی شده است که تنها یک پارامتر بر پیچیدگی و کیفیت تاثیر گذارد. بعنوان نمونه، برای ارزیابی تاثیر پارامتر تعداد فریم‌های مرجع بر کیفیت و پیچیدگی، پارامتر دقت تخمین حرکت به Quarter pel.، محدوده جستجو به ۳۲ پیکسل، مدهای Inter همگی فعال و معیار مقایسه به SSE تنظیم شده‌اند.

شبیه‌سازیهای جدول ۷-۱ بر روی یک پردازنده ۴ هسته‌ای اینتل، با ۴ گیگابایت حافظه موقت و بر روی سیستم عامل ویندوز ۷ صورت گرفته است. درصد افزایش پیچیدگی با تقسیم پیچیدگی هر ترکیب بر اولین ترکیب همان پارامتر بدست آمده است. میزان افزایش کیفیت نیز با محاسبه تفاضل کیفیت هر ترکیب از کیفیت اولین ترکیب از همان پارامتر بدست آمده است.

برای پارامترهایی نظیر تعداد فریم‌های مرجع، دقت عمیات تخمین حرکت، محدوده جستجو و معیار مقایسه میزان افزایش پیچیدگی و کیفیت برای ویدئوهای با اندازه مختلف در هر دو روش تخمین حرکت مشابه است. این در حالی است که برای پارامتر تعداد مدهای فعال Inter، در هر دو روش تخمین حرکت، کیفیت با افزایش اندازه ویدئو، افزایش می‌یابد. میزان افزایش پیچیدگی برای این مد، برای اندازه‌های مختلف ویدئو در هر یک از روش‌های کامل و سریع مشابه است. اما، رفتار دو روش تخمین حرکت به ازای این پارامتر کاملا متفاوت می‌باشد که علت این مسئله بعدا در همین بخش توضیح داده خواهد شد.

همانطور که نتایج جدول ۷-۱ نشان می‌دهد، میزان افزایش پیچیدگی در روش تخمین حرکت کامل برای پارامترهای: تعداد فریم‌های مرجع، دقت عملیات تخمین حرکت و محدوده جستجو، قابل توجه است اما برای پارامتر تعداد مدهای فعال Inter ناچیز می‌باشد. به عبارت دیگر پارامتر تعداد مدهای فعال Inter تاثیر ناچیزی بر روی پیچیدگی روش تخمین حرکت کامل دارد. دلیل این تاثیر ناچیز استفاده از روش SAD reuse در عملیات تخمین حرکت کامل می‌باشد. با استفاده از این روش، مقدار SAD بلوکهای با اندازه بزرگتر، با استفاده از مقدار SAD بلوکهای با اندازه ۴×۴ (SAD4×4) بدست می‌آید، به این ترتیب با افزایش تعداد مدهای Inter عملا ترکیبات جدیدی از مقادیر SAD4×4 بایستی با هم جمع شود که این عملیات پیچیدگی ناچیزی به کل مجموعه عملیات تحمیل می‌کند.

با توجه به تاثیر ناچیز پارامتر تعداد مدهای فعال Inter بر پیچیدگی روش تخمین حرکت کامل، و بر اساس روند پیشنهادی انتخاب پارامترها، این پارامتر بایستی از لیست پارامترهای تخمین حرکت کامل حذف گردد. با حذف این پارامتر، بیشترین مقدار (یعنی فعال بودن تمامی مدهای Inter) برای آن لحاظ خواهد شد.

بر خلاف روش تخمین حرکت کامل، در روش تخمین حرکت سریع پارامتر تعداد مدهای فعال Inter تاثیر قابل توجهی بر روی پیچیدگی دارد. به این ترتیب این پارامتر به همراه پارامترهای تعداد فریم‌های مرجع، دقت عملیات تخمین حرکت و محدوده جستجو در لیست پارامترهای روش تخمین حرکت سریع در نظر گرفته خواهد شد.

لازم به ذکر است که مطابق نتایج جدول ۷-۱، پارامتر معیار مقایسه تاثیر ناچیزی در افزایش کیفیت برای هر دو روش تخمین حرکت مدنظر دارد. به این ترتیب این پارامتر نیز از لیست پارامترهای هر دو روش تخمین حرکت حذف شده و مقدار آن به SAD تنظیم خواهد شد.

به این ترتیب پارامترهای کدکردنی که بیشترین تاثیر را روی پیچیدگی و کیفیت روش‌های تخمین حرکت کامل و سریع دارند را در لیست ۷-۲ آورده ایم.

تخمین حرکت با جستجوی کامل:	تخمین حرکت با جستجوی سریع:
– دقت بردار حرکت (*Res*)	– دقت بردار حرکت (*Res*)
– محدوده جستجو (*SR*)	– محدوده جستجو (*SR*)
– تعداد فریم‌های مرجع (*NRef*)	– تعداد فریم‌های مرجع (*NRef*)
	– تعداد مدهای فعال Inter (*EnM*)

لیست ۷-۲ پارامترهای کدکردن مربوط به واحد تخمین حرکت

در زیر بخشهایی که در ادامه می‌آید انتخاب ترکیبات بهینه، مدل اعوجاج–پیچیدگی و طراحی مربوط به کنترل کننده پیچیدگی ارائه خواهد شد. برای آنکه روش یکسانی برای هر دو روش تخمین حرکت کامل و روش تخمین حرکت سریع داشته باشیم در زیر بخش اول و دوم تنها پارامترهای مشترک بین دو روش که عبارتند از محدوده جستجو، دقت بردار حرکت و تعداد فریم‌های مرجع را در نظر می‌گیریم. تاثیر پارامتر تعداد مدهای فعال برای روش تخمین حرکت سریع در طراحی کنترل کننده پیچیدگی در زیر بخش سوم لحاظ خواهد شد.

۷-۱-۲ انتخاب ترکیبات بهینه

برای هر ترکیبی از {*NRef*, *SR*, *Res*}، یک زوج پیچیدگی و اعوجاج تولید خواهد شد. هدف این زیربخش یافتن زیر مجموعه‌ای از ترکیبات {*NRef*, *SR*, *Res*} است که برای هر سطح پیچیدگی منجر به کمترین اعوجاج ممکن شوند. این زیر مجموعه از ترکیبات را نمودار محدب[1] می‌نامیم. بر اساس روند انتزاعی پیشنهادی، برای استخراج این مجموعه دو قانون زیر را برای ترکیبیات مختلف اجرا می‌کنیم:

۱. در بین ترکیبات دارای اعوجاج یکسان ترکیبی انتخاب می‌شود که کمترین پیچیدگی را دارد.

۲. در بین ترکیبات با پیچیدگی یکسان ترکیبی انتخاب می‌شود که کمترین اعوجاج را دارد.

برای اجرای قوانین بالا، ابتدا بایستی مقادیر اعوجاج–پیچیدگی ترکیبات مختلف را استخراج نماییم. برای استخراج پیچیدگی هر ترکیب از فرمولهای پیچیدگی آمده در بخش ۴-۲-۱-۲-۴ استفاده می‌کنیم. برای استخراج اعوجاج هر ترکیب نیز، یکسری ویدئو نمونه را با کدکننده H.264/AVC، در حالیکه پارامترهای کدکردن آن بر اساس ترکیب مد نظر تنظیم شده‌اند، کد کرده و اعوجاج متناظر را بدست می‌آوریم. لازم به ذکر است که اعوجاج با معیار MSE محاسبه می‌گردد.

از آنجا که هر یک از پارامترهای محدوده جستجو، دقت بردار حرکت و تعداد فریم‌های مرجع در محدوده متنوعی تغییر می‌کنند، انجام شبیه‌سازی برای تمامی ترکیبات ممکن میسر نخواهد بود. به هیمن دلیل ابتدا با انجام یک شبیه‌سازی اولیه بر روی ۱۶ ویدئو مختلف، محدوده موثر برای هر یک از پارامترها را بدست آورده ایم. محدوده موثر هر پارامتر با تعریف نقطه

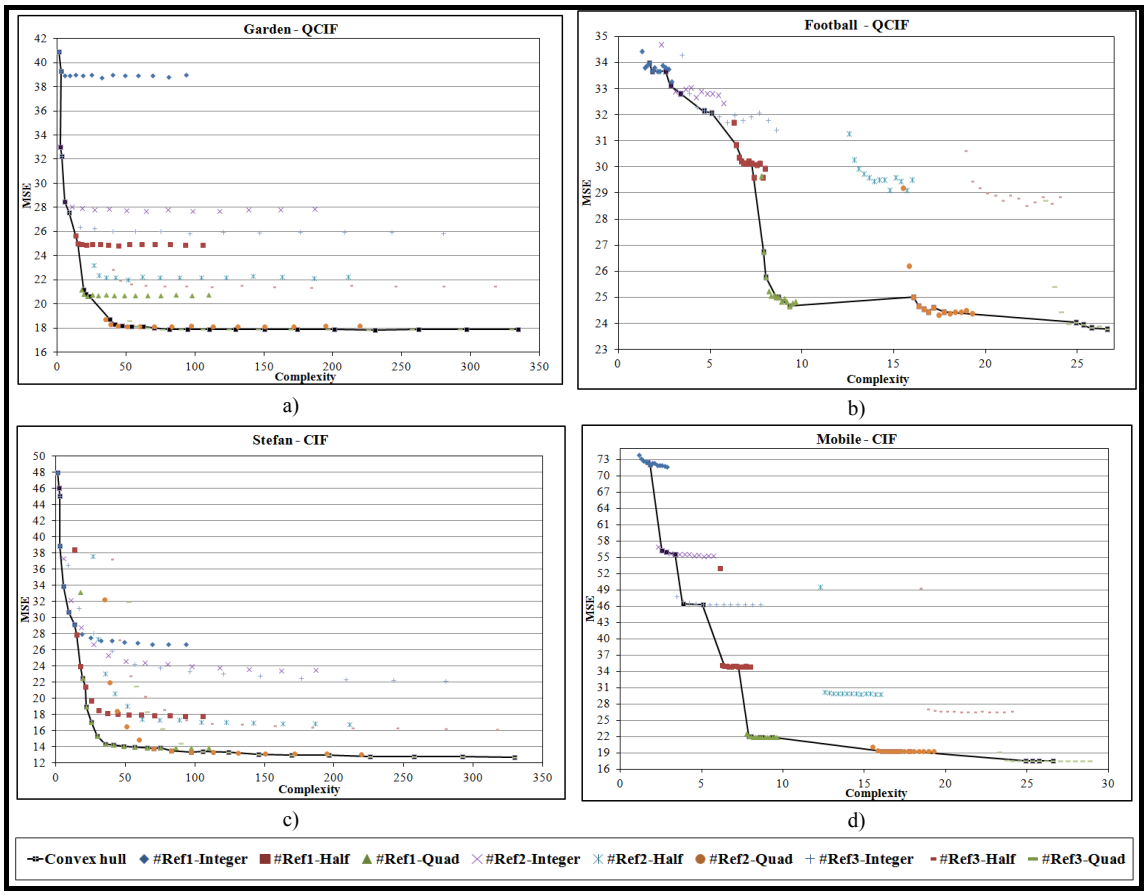
[1] Convex Hull

اشباع بدست آمده است. نقطه اشباع، نقطه‌ای است که در آن میزان افزایش کیفیت در مقابل میزان افزایش پیچیدگی ناچیز می‌باشد. بر اساس نتایج شبیه‌سازیها، محدوده پارامترهای مختلف طبق جدول ۷-۳ بدست آمده است.

جدول ۷-۳ محدوده موثر برای پارامترهای پیچیدگی مختلف

متغیر	SR	NRef	Res
محدوده	{1,2,…, 14}	{1,2,3}	{Full, Half, Quarter}

پس از تعیین محدوده برای هر متغیر نمودار پیچیدگی–اعوجاج را بدست می‌آوریم. برای اینکار تمامی ۱۲۶ ترکیب ممکن از سه متغییر مذکور در نظر گرفته شده و پیچیدگی و اعوجاج برای هر ترکیب محاسبه شده است. این ترکیبات برای ۱۶ ویدئو استاندارد با اندازه‌های QCIF، CIF و 4CIF استخراج شده‌اند. شکل ۷-۱ زوجهای مختلف اعوجاج–پیچیدگی را به همراه نمودار محدب آنها برای هر یک از روش‌های تخمین حرکت کامل و سریع و برای ۴ ویدئو مختلف نشان می‌دهد. لازم به ذکر است که پیچیدگی با اختصاص مقدار ۱ به ترکیب با کمترین پیچیدگی ($NRef = 1, SR = 1, Res = \text{Full}$) نرمالیزه شده است. همانگونه که شکل ۷-۱ نشان می‌دهد، در هر یک از روش‌های تخمین حرکت، تنظیمات بدست آمده در ویدئوهای مختلف تا حد زیادی به همدیگر نزدیک است. بنابراین، برای هر یک از روش‌های تخمین حرکت، می‌توان یک مجموعه تنظیمات یکسان را با دقت قابل قبولی برای تمامی ویدئوها ارائه کرد. به این ترتیب مجموعه تنظیمات آمده در جدول ۷-۴ بعنوان بهترین تنظیمات برای رسیدن به کمترین اعوجاج به ازای پیچیدگی تعریف شده است.



شکل ۷-۱ زوجهای اعوجاج–پیچیدگی و نمودار محدب برای ویدئوهای مختلف a) و c) برای تخمین حرکت کامل و b) و d) برای تخمین حرکت سریع

جدول ۷-۴ مجموعه پارامترهای بهینه (پیچیدگی-اعوجاج) برای سطوح مختلف پیچیدگی

سطح پیچیدگی	تخمین حرکت سریع			تخمین حرکت کامل		
	NRef	SR	Res	NRef	SR	Res
۱	۱	۶	F	۱	۱	F
۲	۱	۷	F	۲	۱	F
۳	۲	۳	F	۱	۲	F
۴	۲	۴	F	۳	۱	F
۵	۲	۶	F	۱	۱	H
۶	۳	۵	F	۱	۱	Q
۷	۳	۶	F	۱	۳	Q
۸	۱	۳	H	۱	۴	Q
۹	۱	۶	H	۲	۳	Q
۱۰	۱	۹	H	۱	۷	Q
۱۱	۱	۲	Q	۳	۳	Q
۱۲	۱	۳	Q	۳	۴	Q
۱۳	۱	۷	Q	۲	۷	Q
۱۴	۱	۸	Q	۱	۱۱	Q
۱۵	۱	۱۲	Q	۳	۶	Q
۱۶	۲	۳	Q	۳	۷	Q
۱۷	۲	۴	Q	۳	۸	Q
۱۸	۲	۵	Q	۳	۹	Q
۱۹	۲	۶	Q	۲	۱۲	Q
۲۰	۲	۷	Q	۲	۱۳	Q
۲۱	۲	۹	Q	۳	۱۱	Q
۲۲	۳	۵	Q	۳	۱۲	Q
۲۳	۳	۶	Q	۳	۱۳	Q
۲۴	۳	۷	Q	۳	۱۴	Q
۲۵	۳	۹	Q			

در این جدول، پیچیدگی در ۲۵ سطح مختلف برای تخمین حرکت سریع و در ۲۴ سطح مختلف برای روش تخمین حرکت کامل تعریف شده است. در جاییکه میزان پیچیدگی با بالا رفتن سطح، افزایش می‌یابد. میزان واقعی پیچیدگی برای هر سطح، به سادگی با مقدار دهی پارامترهای *NRef*، *SR*، *h* و *q* در روابط (۴-۵۷)-(۴-۵۴) برای روش تخمین حرکت با جستجوی کامل و (۴-۶۱)-(۴-۵۸) برای روش تخمین حرکت با جستجوی سریع، قابل محاسبه است. این مقادیر پیچیدگی را می‌توان در یکسری جدول جستجو[1] ذخیره کرد.

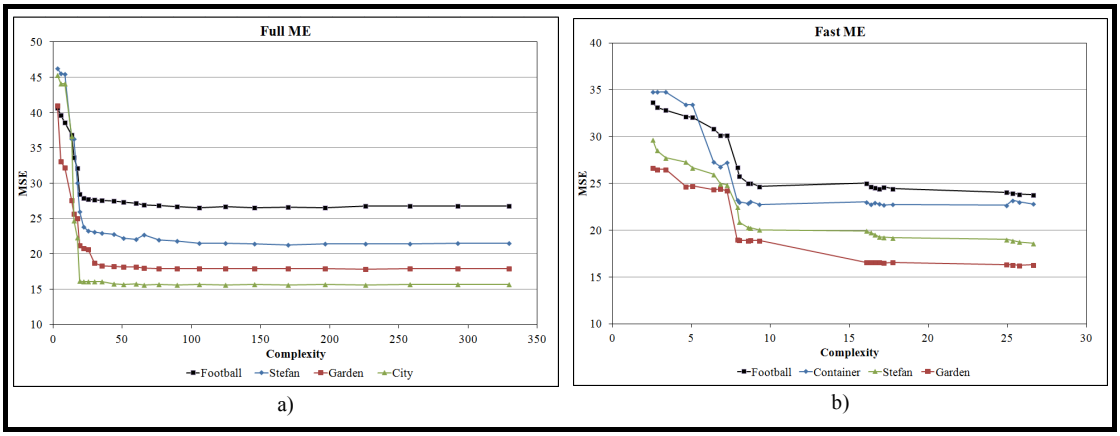
با استفاده از پارامترهای پیشنهادی در جدول ۷-۴ می‌توان پارامترهای کدکردن را در سطح ماکروبلوک تنظیم نمود اما با این شیوه بهترین نتیجه حاصل نمی‌شود، چرا که در این روش تنها پیچیدگی لحاظ می‌شود و اعوجاج تحمیلی متناظر با هر سطح پیچیدگی در نظر گرفته نمی‌شود. بنابراین روش مناسب‌تر روشی خواهد بود که در تعیین سطح پیچیدگی برای هر ماکروبلوک تاثیر توأمان پیچیدگی و اعوجاج را لحاظ کند. به عبارت دیگر، برای یافتن بهترین سطح پیچیدگی برای هر ماکروبلوک بایستی

[1] Look-up Table

پیچیدگی و اعوجاج سطوح مختلف را استخراج کنیم و سطحی را انتخاب کنیم که منجر به بهترین مصالحه بین پیچیدگی و اعوجاج گردد. در بخش بعدی این مسئله را با یافتن یک مدل مناسب برای پیچیدگی و اعوجاج و سپس استفاده از روش بهینه‌سازی لاگرانژ بیشتر بررسی خواهیم کرد.

۷-۱-۳ بدست آوردن مدل اعوجاج-پیچیدگی

برای ارائه یک مدل اعوجاج-پیچیدگی برای ترکیبات آمده در جدول ۷-۴، منحنی محدب (بهینه) اعوجاج-پیچیدگی را برای ۴ ویدئو QCIF، ۳ ویدئو CIF و ۳ ویدئو 4CIF استخراج کرده‌ایم. در شکل ۷-۲ برخی از این منحنی‌ها را برای ویدئوهای QCIF و برای هر دو روش تخمین حرکت سریع و تخمین حرکت کامل، آورده ایم.



شکل ۷-۲ منحنی محدب اعوجاج-پیچیدگی برای ویدئوهای QCIF، a) تخمین حرکت کامل، b) تخمین حرکت سریع

همانطور که در این شکل آمده است، در هر یک از روش‌های تخمین حرکت، رفتار نمودارهای اعوجاج-پیچیدگی برای ویدئوهای مختلف رفتار نمایی مشابهی دارد. تفاوت منحنی‌ها تنها در میزان اعوجاج می‌باشد که به دلیل تفاوت محتوا و نرخ بیت ویدئوهای مختلف ایجاد شده است. بر اساس شبیه‌سازیهای انجام شده، رفتار مشابهی برای ویدئوهای با اندازه دیگر نیز مشاهده کردیم.

با توجه به نمودارهای اعوجاج-پیچیدگی استخراج شده برای ۱۰ ویدئو و با استفاده از ابزار برازش منحنی[1] یک رابطه کلی برای اعوجاج-پیچیدگی برای هر دو روش تخمین حرکت بدست می‌آوریم که در رابطه (۷-۲) آمده است.

$$D = a_0 \times e^{-a_1 \times (C+\alpha)} + a_2 \times e^{-a_3 \times (C+\alpha)} - \beta \qquad (۷-۲)$$

در این رابطه، D نمایانگر اعوجاج می‌باشد که با معیار MSE محاسبه شده است و C معیار پیچیدگی می‌باشد که بر اساس تعداد پالس ساعت بدست آمده است. لازم به ذکر است که اگر چه در جدول ۷-۴، دو مجموعه متفاوت از پارامترهای پیچیدگی داریم اما در اینجا با استفاده از a_0، a_1، a_2 و a_3 که اعداد حقیقی ثابتی هستند، با یک رابطه اعوجاج-پیچیدگی هر دو روش را مدل کرده‌ایم. مقادیر a_0، a_1، a_2 و a_3 در جدول ۷-۵ برای روش‌های تخمین حرکت کامل و روش تخمین حرکت سریع آمده است. این مقادیر با برازش رابطه آمده در (۷-۲) بر یکی از نمودارهای بهینه اعوجاج-پیچیدگی برای هر یک از

[1] Curve Fitting

روش‌های تخمین حرکت بدست آمده است، در حالیکه پارامترهای α و β مقدار صفر داشته‌اند. نمودار اعوجاج-پیچیدگی برای ویدئوهای دیگر همین رفتار نمایی را با عرض از مبدا و طول از مبدا متفاوتی خواهد داشت که با تنظیم α و β بدست خواهد آمد. به عبارت دیگر مقادیر α و β براساس محتوای ویدئو و نرخ بیت آن تنظیم خواهند شد. برای هر ویدئوی جدید، ابتدا با یک مقدار پیش‌گزیده برای α و β شروع می‌کنیم، سپس آنها را در ابتدای هر GOP و بصورت زمان واقعی به روز می‌کنیم. برای بروز کردن این پارامترها، زوجهای مختلف اعوجاج-پیچیدگی تمامی ماکروبلوکهای اولین فریم P از هر GOP را بررسی می‌کنیم. از بین این زوجها ۱۲ زوج متمایز را جدا کرده و با استفاده از رابطه (۷-۲) و حل معادلات خطی، مقادیر α و β را به روز می‌کنیم.

جدول ۷-۵ مقادیر a_0، a_1، a_2 و a_3 برای ویدئوهای مختلف

روش تخمین حرکت	a_0	a_1	a_2	a_3
تخمین حرکت کامل	34.02	1.232×10^{-5}	18.83	4.631×10^{-9}
تخمین حرکت سریع	16.95	8.313×10^{-7}	19.71	9.844×10^{-9}

۷-۱-۴ حل مسئله بهینه‌سازی و استخراج پارامترهای لاگرانژ

رابطه (۷-۱) با استفاده از روش بهینه‌سازی مقید لاگرانژ به رابطه غیر مقید زیر تبدیل می‌شود:

$$J = D + \lambda \times C \qquad (۷-۳)$$

ضریب لاگرانژ در رابطه (۷-۳) با در نظر گرفتن رابطه (۷-۲) و روابط (۴-۶۱)–(۴-۵۴) آمده در فصل ۴ بصورت زیر محاسبه خواهد شد:

$$\lambda = e^{-(C+\alpha)(a_1+a_3)}\left[a_0 a_1 e^{a_3(C+\alpha)} + a_2 a_3 e^{a_1(C+\alpha)}\right] \qquad (۷-۴)$$

با داشتن مقدار λ، رابطه (۷-۳) درون یک کنترل کننده پیچیدگی استفاده خواهد شد تا در سطح هر ماکروبلوک و بر اساس منابع موجود بهترین سطح پیچیدگی ممکن را ارائه نماید. در زیر بخش پیش رو جزئیات این کنترل کننده پیچیدگی را برای هر دو روش تخمین حرکت خواهیم آورد.

۷-۱-۵ طراحی کنترل کننده پیچیدگی

پس از ارائه یک مدل برای پیچیدگی و اعوجاج، می‌توان این مدل را در کدکننده پیاده کرد و یک کنترل کننده دو مرحله‌ای طراحی نمود. با توجه به اینکه روش‌های تخمین حرکت کامل و تخمین حرکت سریع دارای پارامترهای متفاوتی هستند برای هر یک کنترل کننده پیچیدگی مستقلی ارائه خواهیم کرد. در زیر بخشهایی که در ادامه می‌آیند به ترتیب کنترل کننده پیچیدگی برای تخمین حرکت کامل و کنترل کننده پیچیدگی برای تخمین حرکت سریع را تشریح خواهیم کرد.

۷-۱-۵-۱ کنترل کننده پیچیدگی برای تخمین حرکت کامل

با استفاده از مدل اعوجاج-پیچیدگی که در بخش ۷-۱-۳ ارائه شد، می‌توان یک کنترل کننده پیچیدگی دو مرحله‌ای طراحی نمود. در مرحله اول، پارامترهای مربوط به واحد تخمین با استفاده از جدول ۷-۴، روابط (۴-۵۷)–(۴-۵۴) و (۷-۲) تنظیم خواهند شد. در مرحله دوم، عملیات انتخاب مد با به کار گرفتن هر یک از روش‌های R-D-C موجود انجام خواهد شد. در

این بخش و بخش بعد، یک روش انتخاب مد ساده برای مرحله دوم استفاده خواهیم کرد. برای درک بهتر چگونگی انجام عملیات کنترل پیچیدگی برای تخمین حرکت کامل، شبه‌کد مربوط به این عملیات را در قسمت (a) از لیست ۷-۳ آورده‌ایم. همانطور که در شبه‌کد مشخص است، پیش از کد کردن هر فریم، پیچیدگی موجود برای آن فریم (C_{Frame}) بر اساس کل پیچیدگی موجود (C_{Total})، تعیین خواهد شد. بطور مشابه، پیچیدگی موجود برای هر ماکروبلوک (C_{MB}) نیز بر اساس پیچیدگی باقیمانده در سطح فریم و تعداد ماکروبلوکهای باقیمانده در فریم جاری تعیین خواهد شد.

پیچیدگی واحد تخمین حرکت (C_{ME}) نیز بعنوان بخشی از پیچیدگی ماکروبلوک در نظر گرفته می‌شود. برای تخمین حرکت کامل مقدار ۰٫۶ را برای پارامتر γ بصورت تجربی بدست آورده ایم. مقدار این پارامتر با استخراج میزان مشارکت روش تخمین حرکت و روش RDO بکارگرفته شده در پیچیدگی کل ماکروبلوک قابل تنظیم است.

برای یافتن مناسبترین سطح پیچیدگی برای واحد تخمین حرکت هر ماکروبلوک، ابتدا مقدار C_{ME} را با مقدار پیچیدگی هر یک از سطوح آمده در جدول ۷-۴ مقایسه می‌کنیم تا نزدیکترین سطح به پیچیدگی موجود را پیدا کنیم. سپس از بین این سطح و دو سطح مجاورش که طبق جدول ۷-۴ مشخص می‌شوند، سطحی که منجر به کمترین هزینه (J_{K}) طبق رابطه (۷-۳) می‌شود را بعنوان سطح بهینه ($Level^{*}$) انتخاب می‌کنیم. لازم به ذکر است که پیچیدگی (C) و اعوجاج (D) با استفاده از روابط (۴-۵۷)–(۴-۵۴) و (۷-۲) و پارامتر λ با استفاده از رابطه (۷-۴) بدست می‌آیند.

پس از بدست آوردن سطح بهینه ($Level^{*}$)، پارامترهای کدکردن از روی جدول ۷-۴، بدست آمده و عملیات تخمین حرکت کامل صورت خواهد پذیرفت. سپس میزان پیچیدگی موجود در سطح ماکروبلوک با کسر پیچیدگی مصرفی توسط واحد تخمین حرکت ($C_{\mathrm{ME}}^{\mathrm{Consumed}}$)، به روز خواهد شد.

دومین مرحله از کنترل کننده پیچدگی، مرحله تخصیص پیچیدگی به عملیات انتخاب مد و انجام این عملیات می‌باشد. در این مرحله، با توجه به پیچیدگی باقی مانده عملیات انتخاب مد بصورت کامل یا جزیی با شروع از مد ۱۶×۱۶، انجام خواهد شد. پس از هر با انجام عملیات انتخاب مد، میزان پیچیدگی موجود با توجه به پیچیدگی مصرف شده ($C_{\mathrm{MD}}^{\mathrm{Consumed}}$) به روز می‌شود. عملیات انتخاب مد تا هنگامی که پیچیدگی موجود در سطح مد پایان یابد یا کل مدها بررسی گردند ادامه خواهد یافت.

پس از کدکردن هر ماکروبلوک، پیچیدگی فریم با در نظر گرفتن پیچیدگی مصرف شده توسط آخرین ماکروبلوک کدشده ($C_{\mathrm{MB}}^{\mathrm{Consumed}}$) به روز خواهد شد. نهایتا، پس از کدکردن هر فریم پیچیدگی باقیمانده کل با لحاظ کردن پیچیدگی مصرف شده توسط آخرین فریم ($C_{\mathrm{Frame}}^{\mathrm{Consumed}}$) به روز خواهد شد.

۷-۱-۵-۲ کنترل کننده پیچیدگی برای تخمین حرکت سریع

همانگونه که پیشتر اشاره شد، برای روش تخمین حرکت سریع، پارامتر تعداد مدهای فعال نیز بایستی در الگوریتم کنترل کننده پیچیدگی مشارکت داشته باشد. این مسئله باعث می‌شود که الگوریتم متفاوتی برای روش تخمین حرکت سریع طراحی کنیم. شبه‌کد مربوط به عملیات کنترل پیچیدگی برای تخمین حرکت سریع را در قسمت (b) از لیست ۷-۳ آورده‌ایم. مشابه با کنترل کننده پیچیدگی برای روش تخمین حرکت صحیح، در اینجا نیز یک کنترل کننده دو سطحی خواهیم داشت. در مرحله

اول از این کنترل کننده، پارامترهای مربوط به واحد تخمین حرکت با استفاده از جدول ۷-۴ و روابط (۴-۶۱)–(۴-۵۸) به ازای هر مد تنظیم خواهد شد. در مرحله دوم، عملیات انتخاب مد با استفاده از هر یک از روش‌های معمول R-D-C قابل انجام خواهد بود. لازم به ذکر است که پیچیدگی فریم (C_{Frame}) و پیچیدگی ماکروبلوک (C_{MB}) به طریق مشابهی که در بخش پیشین برای روش تخمین حرکت کامل توضیح داده شده، صورت خواهد پذیرفت. در این الگوریتم به γ بصورت تجربی مقدار ۰٫۵ اختصاص داده شده است. در روش تخمین حرکت سریع، عملیات تخمین حرکت بایستی به ازای هر یک از مدهای فعال تکرار شود. از آنجا که پیچیدگی عملیات تخمین حرکت برای مدهای مختلف یکسان نیست، از یک پارامتر وزن مد (W_{mode}) در این کنترل کننده پیچیدگی استفاده کرده‌ایم. مقدار این پارامتر را نیز بصورت تجربی تعیین می‌کنیم که برای مدهای ۱۶×۱۶، ۱۶×۸ و ۸×۱۶ مقدار یک و برای مد ۸×۸ و زیر مدهای آن، مقدار ۴ را خواهد گرفت.

جهت یافتن مناسبترین سطح برای هر مد، پیچیدگی وزن دار هر سطح با C_{ME} مقایسه خواهد شد. در صورتیکه پیچیدگی وزن دار باقیمانده ($\frac{C_{ME}}{8-\sum_{i=1}^{mode} W_i}$) معادل هیچ سطحی نباشد (به عبارت دیگر level = -1)، مد فعلی از لیست مدهای فعال حذف خواهد شد. به این ترتیب عملیات تخمین حرکت و عملیات انتخاب مد برای این مد انجام نخواهد پذیرفت. در غیر اینصورت، بر اساس سطح استخراج شده توسط این بخش الگوریتم و با در نظر گرفتن سطوح همسایه اش از جدول ۷-۴، سطح بهینه ($Level^*$) که دارای کمترین هزینه می‌باشد استخراج خواهد شد. لازم به ذکر است که در اینجا، پیچیدگی (C) و اعوجاج (D) با استفاده از روابط (۴-۶۱)–(۴-۵۸) و (۷-۲) و پارامتر λ با استفاده از رابطه (۷-۴) بدست می‌آیند.

در این الگوریتم نیز، مجموعه پارامتر معادل بهترین سطح از جدول ۷-۴ استخراج شده و سپس عملیات تخمین حرکت صحیح انجام خواهد شد. این عملیات برای تمامی مدهای فعال تکرار خواهد شد.

مرحله دوم این الگوریتم مشابه الگوریتم آمده برای روش تخمین حرکت کامل در بخش قبل است با این تفاوت که عملیات انتخاب مد تنها بر روی مدهای فعال باقیمانده صورت خواهد پذیرفت.

a)

For (each frame)

{

$$C_{Frame} = \frac{C_{Total}}{Number\ of\ remaining\ frames\ in\ sequence}$$

For (each MB)

{

First Phase

$$C_{MB} = \frac{C_{Frame}}{Number\ of\ remaining\ MBs\ in\ frame}$$

$C_{ME} = \gamma . C_{MB}$

Level = Max_Level

While $(C(Level) > C_{ME})$

Level - -

For (k = Level + 1; k ≥ Level -1; k- -)

{

$J_k = D(k) + \lambda . C(k)$

Update J_k^{Min} *and Level**

}

*Update SR, Nref, Res from Table 5, based on Level**

Second Phase

Perform ME

$C_{MB} = C_{MB} - C_{ME}^{Consumed}$

While ($C_{MB} > 0$ *or there is any other enabled mode)*

{

Perform MD for one mode

$C_{MB} = C_{MB} - C_{MD}^{Consumed}$

}

$C_{Frame} = C_{Frame} - C_{MB}^{Consumed}$

}

b)

For (each frame)

{

$$C_{Frame} = \frac{C_{Total}}{Number\ of\ remaining\ frames\ in\ sequence}$$

For (each MB)

{

First Phase

$$C_{MB} = \frac{C_{Frame}}{Number\ of\ remaining\ MBs\ in\ frame}$$

$C_{ME} = \gamma . C_{MB}$

L1: *For (all enabled modes)*

{

Level = Max_Level

While $\left(\frac{W_{mode}}{7} C(Level) > \frac{C_{ME}}{8 - \sum_{i=1}^{mode} W_i}\right)$

Level - -

If $(Level == -1)$

Disable current mode, go to L1

For (k = Level + 1; k ≥ Level -1; k- -)

{

$J_k = D(k) + \lambda . C(k)$

Update J_k^{Min} *and Level**

}

*Update SR, Nref, Res from Table 5, based on Level**

Perform ME

$C_{ME} = C_{ME} - C_{ME}^{mode}$

}

Second Phase

$C_{MB} = C_{MB} - C_{ME}^{Consumed}$

While ($C_{MB} > 0$ *or there is any other enabled mode)*

{

Perform MD for one mode

$C_{MB} = C_{MB} - C_{MD}^{Consumed}$

}

$C_{Frame} = C_{Frame} - C_{MB}^{Consumed}$

لیست ۷-۳ شبه‌کد پیشنهادی برای کنترل کننده پیچیدگی، a) تخمین حرکت کامل، b) تخمین حرکت سریع

۷-۲ نتایج شبیه‌سازی

برای ارزیابی کارایی روش پیشنهادی، این روش با روش DRA[1] مقایسه شده است. روش DRA یک روش پایه اختصاص منابع می‌باشد. مقالات دیگری نظیر [45] و [48] نیز خود را با این روش مقایسه کرده‌اند.

در روش DRA، ابتدا عملیات تخمین حرکت صحیح صورت می‌گیرد، در صورتیکه میزان منابع باقیمانده کافی باشد عملیات تخمین حرکت اعشاری نیز انجام خواهد شد. در قسمت انتخاب مد نیز، ابتدا مدهای Skip و ۱۶×۱۶ بررسی می‌شوند و سپس با توجه به میزان پیچیدگی باقیمانده بقیه مدها ارزیابی خواهند شد.

در شبیه‌سازیهای انجام شده، تنظیمات آمده در جدول ۷-۶ برای کدکننده لحاظ شده‌اند.

جدول ۷-۶ تنظیمات کدکننده H.264/AVC برای شبیه‌سازیهای انجام شده

پارامتر	مقدار
پروفایل	Baseline
سطح	۳
تعداد فریم کدشده	۱۰۰
نرخ فریم	۳۰
اندازه GOP	۱۵
تعداد فریمهای مرجع	۲
محدوده جستجو	۱۶
کنترل کننده نرخ بیت	روشن
اندازه ویدئوهای شبیه سازی شده	QCIF، CIF و 4CIF

لازم به ذکر است که عملیات انتخاب مد برای هر دو روش پیشنهادی و روش DRA، یکسان است تا به این ترتیب بتوان مقایسه منصفانه‌ای بین دو روش مذکور انجام داد.

در سناریوی شبیه‌سازی این مرحله، ابتدا هر ویدئو نمونه بصورت معمولی و بدون تعریف هیچ قیدی برای پیچیدگی کد می‌شود و مقدار واقعی پیچیدگی مصرفی آن استخراج می‌شود. سپس بر اساس پیچیدگی استخراج شده، عملیات کدکردن با روش پیشنهادی و روش DRA برای سطوح مختلف پیچیدگی (از ۸۰ درصد تا ۱۰ درصد پیچیدگی واقعی برای روش تخمین حرکت کامل و از ۸۰ درصد تا ۴۰ درصد پیچیدگی واقعی برای روش تخمین حرکت سریع) صورت خواهد پذیرفت.

نتایج ارزیابی برای ویدئوهای مختلف در جدول ۷-۷، جدول ۷-۸ و شکل ۷-۳ آمده است. همانطور که در جداول و شکل مشاهده می‌شود، روش پیشنهادی از روش DRA در تمامی حالتهای شبیه‌سازی بهتر عمل می‌کند.

[1] Direct Resource Allocation

کارایی پایینتر روش DRA ناشی از کنترل تنها یک پارامتر (*Res*) و عدم کنترل پارامترهای *NREF* و *SR* می‌باشد. در حالاتی که تعداد زیادی از بردارهای حرکت بایستی بصورت اعشاری کد شوند اما مرحله تخمین حرکت اعشاری بدلیل کمبود منبع غیرفعال شده است، خرابی کیفیت و عملکرد بدتر روش DRA بیشتر مشهود است. بعنوان نمونه در جدول ۷-۷ و برای ویدئو Bus که حرکت اشیاء زیاد است و یا برای ویدئو Silent که تعداد زیادی شئی با ابعاد کوچک داریم، کیفیت روش DRA به میزان قابل توجهی کاهش یافته است. کارائی بهتر روش پیشنهادی ما نیز نه تنها به دلیل در نظر گرفتن تاثیر پارامترهای {*NREF*, *SR*, *Res*} بلکه بدلیل در نظر گرفتن توالی تغییر این پارامترها می‌باشد. بعبارت دیگر، طبق آنچه در جدول ۷-۴ مد نظر قرار گرفته است، کاهش پارامترهای *NREF* و *SR* اولویت بیشتری نسبت به کاهش پارامتر *Res* از منظر کیفیت دارد. همانطور که در جدول ۷-۴ آمده است، تخمین حرکت اعشاری در ۷۵٪ سطوح فعال می‌باشد که این بدلیل تاثیر بیشتر این پارامتر بر کیفیت در مقایسه با دیگر پارامترها می‌باشد.

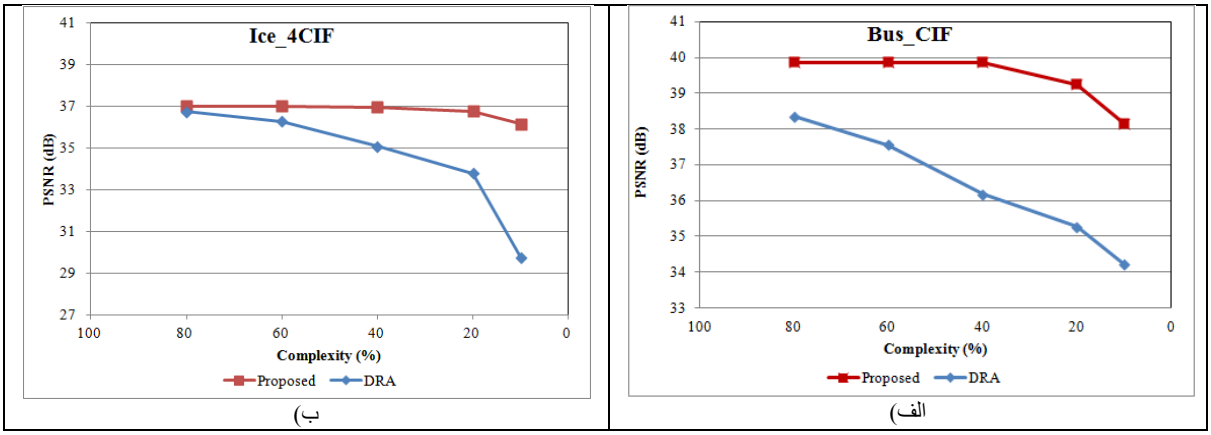
بر اساس نتایج شبیه‌سازیها، روش پیشنهادی بصورت میانگین به ترتیب ۱,۰۲ dB و ۱,۰۶ dB برای روش‌های تخمین حرکت کامل و روش تخمین حرکت سریع، از روش DAR بهتر عمل می‌کند. به علاوه در بیشتر موارد شبیه‌سازی، کیفیت روش پیشنهادی به روش معمول که هیچ قیدی بر روی پیچیدگی نداریم بسیار نزدیک است. بطور میانگین برای روش تخمین حرکت کامل روش پیشنهادی ۰,۱۳ dB بدتر از روش معمول عمل می‌کند. این در حالیست که این پارامتر برای روش DRA حدود ۰,۸۹ dB می‌باشد. برای روش تخمین حرکت سریع، روش پیشنهادی ۰,۵۹ dB و روش DRA در حدود ۱,۵۷ dB بدتر از روش معمول کار می‌کنند.

جدول ۷-۷ ارزیابی کارایی روش پیشنهادی و روش DRA برای ویدئوهای مختلف – تخمین حرکت کامل

اندازه ویدئو	ویدئو نمونه	نرخ بیت (Kbps)	کیفیت بدون محدودیت پیچیدگی	کیفیت برای ۸۰٪ پیچیدگی (dB)		کیفیت برای ۶۰٪ پیچیدگی (dB)		کیفیت برای ۴۰٪ پیچیدگی (dB)		کیفیت برای ۲۰٪ پیچیدگی (dB)		کیفیت برای ۱۰٪ پیچیدگی (dB)	
				DRA	پیشنهادی	DRA	پیشنهادی	DRA	پیشنهادی	DRA	پیشنهادی	DRA	پیشنهادی
QCIF	Akiyo	۴۸	۳۹,۳۹	۳۹,۱۵	۳۹,۳۱	۳۹,۲۰	۳۹,۳۹	۳۹,۱۹	۳۹,۳۷	۳۹,۲۷	۳۹,۴۰	۳۸,۵۸	۳۹,۴
	Salesman	۶۴	۳۴,۹۷	۳۴,۵۸	۳۴,۹۷	۳۴,۵۶	۳۴,۹۲	۳۴,۵۸	۳۴,۹۰	۳۴,۶۸	۳۴,۸۹	۳۳,۹۳	۳۴,۸۹
	Silent	۹۶	۳۷,۴۴	۳۶,۷۷	۳۷,۴۷	۳۶,۸۵	۳۷,۴۴	۳۶,۷۶	۳۷,۴۲	۳۶,۸۱	۳۷,۳۳	۳۵,۸۷	۳۷,۳۳
CIF	Bus	۲۸۷۰	۳۹,۸۸	۳۸,۳۶	۳۹,۸۷	۳۷,۵۵	۳۹,۸۷	۳۶,۱۸	۳۹,۸۶	۳۵,۲۶	۳۹,۲۴	۳۴,۲۳	۳۸,۱۵
	Container	۱۵۰	۳۴,۹۱	۳۴,۸۱	۳۴,۹۰	۳۴,۷۵	۳۴,۸۹	۳۴,۸۱	۳۴,۸۴	۳۴,۷۴	۳۴,۸۶	۳۴,۴۳	۳۴,۸۹
	Flower	۲۰۰۰	۳۶,۲۷	۳۵,۹۱	۳۶,۲۸	۳۵,۷۸	۳۶,۲۴	۳۵,۸۵	۳۶,۲۷	۳۵,۸۲	۳۶,۲۴	۳۴,۹۸	۳۵,۲۸
4CIF	Harbour	۳۶۰۰	۳۵,۴۹	۳۵,۳۹	۳۵,۵۰	۳۵,۳۶	۳۵,۴۹	۳۵,۳۵	۳۵,۴۹	۳۵,۳۱	۳۵,۴۹	۳۵,۸۳	۳۴,۹۲
	Ice	۳۷۰	۳۷,۰۰	۳۶,۷۴	۳۷,۰۰	۳۶,۲۸	۳۷,۰۱	۳۵,۰۷	۳۶,۹۷	۳۳,۷۸	۳۶,۷۶	۲۹,۷۴	۳۶,۱۵
	Walk	۲۵۰۰	۳۴,۷۸	۳۴,۶۲	۳۴,۷۵	۳۴,۵۲	۳۴,۶۶	۳۳,۷۸	۳۴,۵۱	۳۲,۰۲	۳۴,۰۴	۲۸,۲۹	۳۲,۵۲

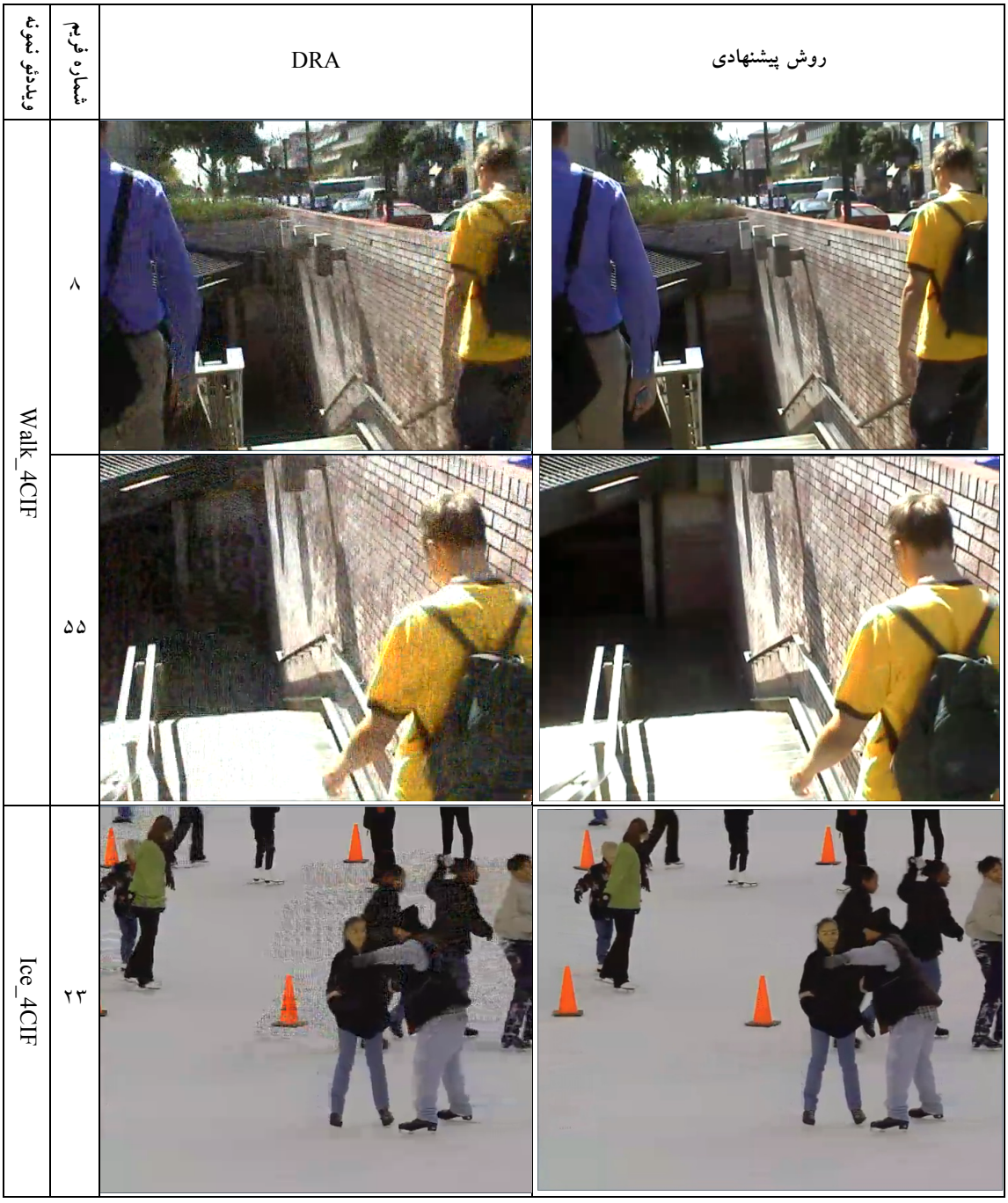
جدول ۷-۸ ارزیابی کارایی روش پیشنهادی و روش DRA برای ویدئوهای مختلف – تخمین حرکت سریع

اندازه ویدئو	ویدئو نمونه	نرخ بیت (Kbps)	کیفیت بدون محدودیت پیچیدگی	کیفیت برای ۸۰٪ پیچیدگی (dB)		کیفیت برای ۷۰٪ پیچیدگی (dB)		کیفیت برای ۶۰٪ پیچیدگی (dB)		کیفیت برای ۵۰٪ پیچیدگی (dB)		کیفیت برای ۴۰٪ پیچیدگی (dB)	
				DRA	پیشنهادی	DRA	پیشنهادی	DRA	پیشنهادی	DRA	پیشنهادی	DRA	پیشنهادی
QCIF	Akiyo	۴۸	۳۹,۳	۳۸,۵۵	۳۹,۲۴	۳۸,۵	۳۹,۱۶	۳۸,۳۷	۳۹,۰۴	۳۸,۳۳	۳۸,۸۴	۳۸,۱۹	۳۸,۶۷
	Salesman	۶۴	۳۴,۸۶	۳۳,۸۴	۳۴,۶۸	۳۳,۷۱	۳۴,۵۸	۳۳,۶۵	۳۴,۴۶	۳۳,۵۹	۳۴,۲۹	۳۳,۵۳	۳۴,۱۳
	Silent	۹۶	۳۷,۴۱	۳۶,۱۷	۳۶,۷۵	۳۵,۹۱	۳۶,۶۹	۳۵,۷۹	۳۶,۶۶	۳۵,۷۱	۳۶,۵۶	۳۵,۶۳	۳۶,۳۳
CIF	Bus	۲۸۷۰	۳۹,۸۶	۳۵,۸۴	۳۸,۵	۳۵,۳۵	۳۸,۳۴	۳۴,۹۱	۳۸,۰۸	۳۴,۶۵	۳۷,۸۱	۳۴,۲۸	۳۷,۷۳
	Container	۱۵۰	۳۴,۸۶	۳۴,۴۹	۳۴,۷۶	۳۴,۴۸	۳۴,۶۹	۳۴,۴۴	۳۴,۶۷	۳۴,۴۱	۳۴,۶۵	۳۴,۳۶	۳۴,۶۱
	Flower	۲۰۰۰	۳۶,۲۳	۳۵,۰۹	۳۵,۹۸	۳۴,۸	۳۵,۹۶	۳۴,۷۱	۳۵,۸۶	۳۴,۶۳	۳۵,۶۶	۳۴,۵۸	۳۵,۳۳
4CIF	Harbour	۳۶۰۰	۳۵,۴۸	۳۴,۹۴	۳۵,۳۶	۳۴,۹۳	۳۵,۳۲	۳۴,۹۴	۳۵,۳۱	۳۴,۹۲	۳۵,۲۳	۳۴,۹۱	۳۵,۱۱
	Ice	۳۷۰	۳۶,۸۹	۳۵,۳۰	۳۵,۴۷	۳۳,۹۷	۳۴,۸۴	۳۲,۹۶	۳۴,۱۶	۳۲,۰۲	۳۳,۷۸	۳۱,۹۹	۳۳,۳۶
	Walk	۲۵۰۰	۳۴,۷۲	۳۴,۲۲	۳۴,۵	۳۳,۹۳	۳۴,۲۱	۳۲,۴۳	۳۳,۹۸	۳۰,۳۶	۳۳,۳۶	۲۹,۲۶	۳۲,۵۱



شکل ۷-۳ رفتار روش پیشنهادی و الگوریتم DRA برای سطوح مختلف پیچیدگی برای الف) ویدئو نمونه Bus با سایز CIF و ب) ویدئو نمونه Ice با سایز 4CIF

برای مشاهده شهودی تفاوت این دو روش چند فریم از ویدئوهای نمونه Walk و Ice در شکل ۷-۴ نشان داده شده است. همانگونه که مشاهده می‌شود، روش پیشنهادی نسبت به روش DRA از کیفیت شهودی بهتری برخوردار است.



شکل ۷-۴ مقایسه شهودی کیفیت روش پیشنهادی در مقایسه با روش DRA – برای کنترل کننده پیچیدگی با تخمین حرکت جستجوی سریع

۷-۳ جمع بندی

در این فصل یک نمونه مطالعاتی برای کدکننده مطلع از محدودیتهای خویش با دنبال کردن روند انتزاعی پیشنهادی در فصل ۳ ارائه شد. از آنجا که واحد تخمین حرکت یکی از پیچیده‌ترین واحدهای کدکننده H.264/AVC می‌باشد، لذا در این فصل به کنترل پیچیدگی این واحد پرداختیم. کنترل کننده پیچیدگی پیشنهادی در این فصل قابل ترکیب با هر کدام از روشهای موجود برای کنترل پیچیدگی در سطح ماکروبلوک می‌باشد. به این ترتیب کنترل کننده پیشنهادی برای کنترل پیچیدگی

عملیات تخمین حرکت و کنترل کننده سطح ماکروبلوک نیز برای کنترل پیچیدگی در حین عملیات انتخاب مد به کار خواهد رفت.

در فصل آتی مباحث رساله را جمع بندی کرده و کارهایی که ممکن است در ادامه این رساله قابل انجام باشند را ارائه خواهیم کرد.

فصل هشتم

نتیجه گیری و کارهای آینده

با پیشرفت تکنولوژی استفاده از کاربردهای چند رسانه‌ای در ابزارهای قابل حمل گسترش یافته است. این در حالی است که این ابزارها دارای محدودیتهای منابع مختلفی می‌باشند. هدف اصلی این رساله طراحی کدکننده مطلع از محدودیتهای منابع – برای در نظر گرفتن این محدودیتها در حین کدکردن و رسیدن به اعوجاج کمتر در سمت گیرنده – در نظر گرفته شده بود. در کنار این هدف اصلی سه زیر مسئله ارائه شد که به اختصار به هر یک اشاره می‌کنیم. در مسئله اول، طراحی کدکننده مطلع از محدودیتهای گیرنده مد نظر است. در این مسئله یک گیرنده در سمت سرویس گیرنده و یک کدکننده در سمت سرویس دهنده فرض می‌شود. هدف، طراحی کدکننده‌ای است که از محدودیتهای گیرنده در این ساختار مطلع باشد. به بیان دیگر، کدکننده پیشنهادی با توجه به مشخصات گیرنده، عملیات کدکردن را به نحوی انجام می‌دهد که رشته بیت تولید شده پس از کدگشایی مصرف منابعی برابر با مقتضیات دستگاه گیرنده داشته باشد. در مسئله دوم، به جای یک گیرنده تعداد متنوعی گیرنده خواهیم داشت. در این ساختار همچنین نیاز به انجام عملیات تطبیق برای برآوردن مشخصات گیرنده‌های مختلف می‌باشد. در این مسئله، هدف طراحی کدکننده‌ای است که با توجه به محدودیتهای گیرنده‌های مختلف و با توجه به رفتار نقطه تطبیق، رشته بیتی تولید کند که علاوه بر برآوردن محدودیت گیرنده‌های مختلف، اعوجاج مجموع مناسبی داشته باشد. بالاخره در مسئله سوم، با توجه به اینکه ممکن است کدکننده بر روی دستگاهی با محدودیت منابع واقع باشد، هدف طراحی کدکننده مطلع از محدودیتهای منابع خویش می‌باشد.

از آنجا که هر سه مسئله فوق ساختار مشابهی دارند، یک روش انتزاعی برای طراحی کدکننده مطلع از محدودیت منابع ارائه کردیم. در طراحی این روند انتزاعی محدودیتهای روشهای مشابه – نظیر در نظر گرفتن زیر مجموعه نامناسبی از پارامترهای کدکردن و عدم امکان گسترش – را برطرف کرده‌ایم. روش انتزاعی پیشنهادی را برای هر یک از سه مسئله مذکور سفارشی

سازی کرده و جزئیات بیشتری برای هر مسئله استخراج نمودیم. در ادامه برای هر یک از مسائل مطرح شده یک نمونه مطالعاتی ارائه شده و با دنبال کردن روند انتزاعیِ سفارشی سازی شده، مسئله مربوط به نمونه مطالعاتی را حل نمودیم. در این نمونه‌های مطالعاتی محدودیتهای نرخ بیت و پیچیدگی را بعنوان منابع و روش MV-Reuse را بعنوان روش تطبیق دهنده به کار گرفتیم. نمونه‌های مطالعاتی بر روی نرم افزار مرجع استاندارد H.264/AVC پیاده سازی و ارزیابی گردید. بر اساس نتایج بدست آمده، در تمامی نمونه‌های مطالعاتی روش پیشنهادی عملکرد بهتری نسبت به روشهای معمول دارد.

از آنجا که برای انجام پیاده سازی و در کاربردهای عملی نیاز به تخمین میزان مصرف منابع داریم، در فصل جداگانه‌ای به ارائه روش شناسی تخمین مصرف منابع پرداختیم. در این رابطه، با در نظر گرفتن الزامات مدلسازی تخمین مصرف منابع یک روند کلی برای آن پیشنهاد کردیم. از جمله این الزامات می‌توان به لزوم استفاده از یک معیار مصرف قابل تبدیل به مصرف واقعی و طراحی عام منظوره مدل تخمین اشاره کرد. به بیان دیگر، در روش شناسی پیشنهادی به شیوه‌ای پیشنهاد می‌شود که مدل تخمین حاصل قابل استفاده برای سکوها و پیاده سازی‌های مختلف باشد. با استفاده از این شیوه شناسی، دو نمونه مطالعاتی برای تخمین مصرف پیچیدگی در کدکننده و کدگشای استاندارد H.264/AVC ارائه و ارزیابی شد. از این مدلهای تخمین در طراحی نمونه‌های مطالعاتی کدکننده مطلع از محدودیتهای منابع استفاده گردید.

در این رساله با توجه به اهمیت مصرف توان یا همان پیچیدگی، این محدودیت در نمونه‌های مطالعاتی مختلف به کار گرفته شد. این محدودیت نه تنها در تخمین میزان مصرف منابع بلکه در ارائه انواع کدکننده مطلع از محدودیت، در نظر گرفته شده است. با استفاده از روند انتزاعی پیشنهادی، مباحث ارائه شده در نمونه‌های مطالعاتی این رساله قابل گسترش به محدودیتهای دیگر نیز می‌باشد. به عبارت دیگر، علاوه بر پیچیدگی، محدودیتهای دیگری نیز می‌توان مد نظر قرار داد که در کاربردهای خاص اهمیت ویژه‌ای دارند. از جمله محدودیتهای دیگری که می‌توان به آن پرداخت به توان محاسباتی پردازنده و میزان مصرف حافظه می‌توان اشاره کرد. برای روش تطبیق نیز شرایط به همین منوال است. در این رساله تنها به عملیات تطبیق MV-Reuse اشاره کردیم، در حالی که روشهای تطبیق دیگر نیز قابل بکارگیری – با در نظر داشتن روند شیوه انتزاعی – خواهند بود. بعنوان نمونه برای دستگاهی که دارای پردازنده با قدرت پردازش پایین است و از عملیات تطبیق تغییر نرخ فریم استفاده می‌کند نیز می‌توان یک کدکننده مطلع از محدودیتهای منابع – با دنبال کردن روش انتزاعی پیشنهادی – ارائه نمود.

روش شناسی پیشنهادی برای تخمین میزان مصرف منابع نیز برای تخمین منابع دیگر دستگاه گیرنده نظیر میزان حافظه مصرفی قابل بکارگیری خواهد بود. بعلاوه روش شناسی پیشنهادی در کاربردهای دیگر که نیاز به تخمین مصرف منابع داشته باشند نیز قابل استفاده است. بعنوان نمونه در روشهای کنترل دینامیک ولتاژ بر اساس توان موجود، نیاز به تخمین حجم کار سیستم[1] در هر لحظه می‌باشد. با استفاده از شیوه شناسی پیشنهادی می‌توان میزان حجم کار سیستم را تخمین زده و آن را بعنوان ورودی به سیستم کنترل دینامیک ولتاژ داد. سپس این سیستم بر اساس میزان حجم کار تخمین زده شده، اقدام به تنظیم ولتاژ خواهد نمود.

[1] Workload

۸-۱ کارهای آینده

با توجه به اینکه در این رساله یک روش انتزاعی برای طراحی کدکننده مطلع از محدودیتهای منابع پیشنهاد کردیم، می‌توان روش انتزاعی را گسترش داد یا آن را با روشهای دیگر ترکیب نمود. مطلع بودن از منابع مصرفی مفهوم دیگری است که ممکن است در کاربردهای دیگر نیز قابل توجیه باشد. در ادامه برخی از فعالیتهایی که می‌توان در ادامه این رساله انجام داد، را معرفی خواهیم کرد.

۸-۱-۱ تعیین نوع عملیات تطبیق در حین کدکردن بر اساس میزان منابع موجود

با این فرض که در نقطه تطبیق می‌توان عملیات تطبیق متنوعی انجام داد، می‌توان مسئله ارائه شده در این رساله را گسترش داد. در این ساختار می‌توان بین عملیات متنوع تطبیق، یک عملیات را انتخاب کرد. مثلا هم می‌توانیم عملیات تطبیق MV-Reuse و هم عملیات حلقه باز را صورت دهیم. در چنین شرایطی میزان مصرف منابع و اعوجاج تحمیلی روشهای مختلف تطبیق، متفاوت خواهد بود و انتخاب بین روشهای مختلف تطبیق به شرایط مسئله اضافه خواهد شد. به عبارت دیگر، در حین کدکردن با توجه به محدودیتهای هر گیرنده و میزان منابع موجود می‌توان برای هر گیرنده عملیات تطبیقی را پیشنهاد کرد که پس از اعمال شدن به رشته بیت، علاوه بر برآوردن محدودیتهای گیرنده متناظرش، کمترین اعوجاج ممکن را نیز لحاظ نماید. برای چنین ساختاری نیاز به طراحی یک کنترل کننده جدید می‌باشد که با توجه به مشخصات عملیات تطبیق، میزان مصرف منابع هر یک و اعوجاج تحمیلی، مصالحه‌ای بین این دو روش ایجاد نماید. لازم به ذکر است که در این مسئله، نحوه تغییر عملیات تطبیق نیز حائز اهمیت است. به بیان دیگر، اینکه با چه فاصله زمانی می‌توان عملیات تطبیق را تغییر داد نیز لازم به بررسی می‌باشد.

۸-۱-۲ در نظر گرفتن محدودیتهای کدکننده و کدگشا بصورت توامان

در این رساله در فصلهای ۵ و ۶، به محدودیتهای کدگشا پرداختیم. در فصل ۷ نیز محدودیتهای کدکننده را در نظر گرفتیم. می‌توان این دو مسئله را توامان در نظر گرفت به این ترتیب که میزان مصرف منابع کدکننده و کدگشا را بصورت یک مسئله دید. چرا که منابعی که کدکننده و کدگشا مصرف می‌کنند عملا از یک منبع مشترک تامین می‌شود که آن منابع مربوط به ابزار چندرسانه‌ای می‌باشد. به این ترتیب می‌توان، با در نظر گرفتن میزان فعالیت کدکننده و کدگشا و میزان منابع موجود مصالحه‌ای بین مصرف توان این دو بوجود آورد که در مجموع میزان اعوجاج کمینه شده و محدودیت منابع دستگاه نیز لحاظ گردد.

۸-۱-۳ در نظر گرفتن محدودیت‌های کانالهای ارتباطی بین سرویس دهنده و سرویس گیرنده‌ها

در این رساله به محدودیتهای کدکننده و کدگشا بر روی دستگاه قابل حمل پرداخته شد، می‌توان مباحث آمده در این رساله را با در نظر گرفتن محدودیتهای کانالهای ارتباطی گسترش داد. به عنوان نمونه می‌توان مشخصات شبکه نظیر نرخ اعوجاج شبکه، تاخیر ارسال و توان مصرفی برای انتقال را نیز به محدودیتهای موجود اضافه نمود تا به ساختار جامع تری دست یافت.

واژه نامه

Abstract	انتزاعی
Training	آموزش دادن
Curve fitting	برازش منحنی
Motion Vector	بردار حرکت
Online	برخط
Offline	برون خط
Quantization Parameter (QP)	پارامتر چندی سازی
Profile	پروفایل
Implementation	پیاده سازی
Complexity	پیچیدگی
Inter Frame Prediction	پیش بینی بین فریمی
Intra Frame Prediction	پیش بینی درون فریمی
Tablet	تبلت
Reference Picture	تصویر مرجع
Smart Phone	تلفن هوشمند
Look-up Table (LUT)	جدول جستجو
Quantization	چندی سازی
Cache memory	حافظه نهان
Residual	داده باقیمانده
Interpolation	درون یابی
Bit Stream	رشته بیت
Methodology	روش شناسی
Assembly	زبان ماشین
Header	سرآیند
Server	سرویس دهنده
Level	سطح
Platform	سکو
High-pass Filter	فیلتر بالا گذر
De-Blocking Filter	فیلتر بلوک زدایی
Low-Pass Filter	فیلتر پایین گذر
Decoding	کد گشایی (بازسازی)
Encoding	کدکردن
Entropy Coding	کدکننده آنتروپی
Self Resource Aware Encoder	کدکننده مطلع از محدودیت منابع خویش

کدکننده مطلع از محدودیت گیرنده	Receiver Aware Encoder (RAE)
کدکننده مطلع از محدودیت گیرنده برای کاربردهای تطبیق	Receiver Aware Encoder for Adaptation Applications ($RAEA^2$)
کیفیت بصری	Subjective Quality
کیفیت عددی	Objective Quality
گام چندی سازی	Quantization Step
واحد	Module
ماکروبلاک	Macro-Block
نرخ	Rate
نرخ - اعوجاج	Rate-Distortion
نرخ بیت	Bit Rate
نرخ عدم اصابت در حافظه نهان	Miss rate
نمودار محدب	Convex hull
نمونه مطالعاتی	Case Study

مراجع

[1] Advanced Video Coding for Generic Audiovisual Services, ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), ITU-T and ISO/IEC JTC 1, Version 8, July 2007.

[2] I.E.G. Richardson, H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia, John Wiley & Sons, 2003.

[3] T. C. Thang, J. G. Kim, J. W. Kang, and J.J Yoo, "SVC adaptation: Standard tools and supporting methods," *Signal Processing: Image Communication*, vol. 24, issue 3, pp. 214-228, 2009.

[4] A. Vetro and S. Chang, "Video Adaptation: Concepts, Technologies, and Open Issues," *Proceedings of the IEEE*, vol. 93 no. 1, pp. 148-158, 2005.

[5] J. Xin, C. W. Lin and M. T. Sun, "Digital Video Transcoding", *Proceedings of the IEEE*, Vol. 93, No. 1, January 2005.

[6] H. Liu, Y. K. Wang, H. Li, "A Comparison between SVC and Transcoding," *IEEE Transactions on Consumer Electronics*, Vol. 54, Issue 3, pp. 1439-1446, 2008.

[7] M. Li, and B. Wang, "An Efficient H.264 Transcoder with Spatial Downscaling for Wireless Communications", *International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1-4, Beijing, China, 2009.

[8] H. Shen, X. Sun, F.Wu, H. Li, and S. Li, "A fast downsizing video transcoder for H.264/AVC with rate-distortion optimal mode decision," *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 2017–2020, Toronto, ON, July 2006.

[9] J. De Cock, S. Notebaert, K. Vermeirsch, P. Lambert, and R. Van de Walle, "Efficient spatial resolution reduction transcoding for H.264/AVC," *IEEE International Conference on Image Processing (ICIP)*, pp. 1208-1211, San Diego, CA, 2008.

[10] C. Hsu, C. Yeh, C. Chen and M. Chen, "Arbitrary frame rate transcoding through temporal and spatial complexity," *IEEE Transactions on Broadcasting*, vol. 55, issue 4, pp. 767-775, 2009.

[11] J. De Cock, S. Notebaert, R. Van de Walle, "Combined SNR and temporal scalability for H.264/AVC using requantization transcoding and hierarchical B pictures," *IEEE International Conference on Multimedia and Expo (ICME)*, pp 448–451, Beijing, China, 2007.

[12] T. C. Chen, Y. H. Chen, S. F. Tsai, S. Y. Chien, and L. G. Chen, "Fast algorithm and architecture design of low-power Integer motion estimation for H.264/AVC," IEEE Transactions on Circuits and Systems for Video Technology, Issue 5, vol. 17, pp. 568-576, 2007.

[13] Z. Chen, J. Xu, Y. He, J. Zheng, "Fast integer-pel and fractional-pel motion estimation for H.264/AVC," *Journal of Visual Communication and Image Representation*, no. 17, pp. 264-290, 2006.

[14] H. Ko, K. Yoo, K. Sohn, "Fast mode-decision for H.264/AVC based on inter-frame correlations," *Signal Processing: Image Communication*, no. 24, pp. 803-813, 2009.

[15] A. Saha, J. Mukherjee, S. Sural, "A neighborhood elimination approach for block matching in motion estimation," *Signal Processing: Image Communication*, no. 26, pp. 438-454, 2011.

[16] W. Dai, O.C. Au, S. Li, L. Sun, R. Zou, "Adaptive search range algorithm based on Cauchy distribution," *Visual Communications and Image Processing Conference*, 2012.

[17] D. Y. Kim, Y. L. Lee, "A fast intra prediction mode decision using DCT and quantization for H.264/AVC," *Signal Processing: Image Communication*, no. 26, 455-465, 2011.

[18] M. De-Frutos-López, D. Orellana-Quirós, J.C. Pujol-Alcolado, F. Díaz-De-María, "An improved fast mode decision algorithm for intra prediction in H.264/AVC video coding," *Signal Processing: Image Communication*, no. 25, pp. 709-716, 2010.

[19] Y. J. Wang, C. C. Cheng, T. S. Chang, "A fast algorithm and its VLSI architecture for fractional motion estimation for H.264/MPEG-4 AVC video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, no. 17, vol. 5, pp. 578-583, 2007.

[20] X. Jin, S. Goto, "Encoder adaptable difference detection for low power video compression in surveillance system," *Signal Processing: Image Communication*, vol. 26, no. 3, pp. 130-142, 2011.

[21] M. Ali, B. Ayed, A. Samet and N. Masmoudi, "H.264/AVC prediction modules complexity analysis," *European Transactions on Telecommunications*, vol. 18, issue 2, pp. 169-177, December 2007.

[22] H. Kalva, "Complexity Estimation of the H.264 Coded Video Bitstreams," *The Computer Journal*, vol. 48, issue 5, pp. 504-513, 2005.

[23] J. Shen, C. Chen, C. Tsai and C. Lin, "Computation-Aware Intra-mode Decision for H.264 Coding and Transcoding," *IEEE International Symposium on Multimedia Workshops (ISMW),* pp. 427-433, Taichung, Taiwan, 2007.

[24] L. Su, Y. Lu, F. Wu, S. Li, and W. Gao, "Complexity-constrained H.264 video encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, issue 4, pp. 477-490, 2009.

[25] B. Lee, M. Kim, S. Hahm, I. J. Cho and C. Park, "A low complexity encoding scheme for coarse grain scalable video coding," *International Conference on Visual Information Engineering*, pp. 753-758, Xi'an, China, 2008.

[26] H. Ma, D. Gangadharan, N. Venkatasubramanian, R. Zimmermann, "Energy-aware Complexity Adaptation for Mobile Video Calls," *ACM Multimedia Conference and Co-Located Workshops*, pp. 1313-1316, 2011.

[27] H. Ma, R. Zimmermann, "Adaptive Coding with CPU Energy Conservation for Mobile Video Calls," *IEEE International Conference on Multimedia and Expo*, pp. 717-722, 2012.

[28] M. C. Chien, P.C. Chang, "Optimal model-based complexity control for H.264 video encoding," *IET Image Processing*, vol. 6, issue 1, Pages 60-71, 2012.

[29] L. W. Kang, C.S. Lu, C.Y Lin, "Low-complexity video coding via power-rate-distortion optimization," *Journal of Visual Communication and Image Representation*, vol. 23, issue 3, Pages 569-585, 2012.

[30] W. Ji, J. Liu, M. Chen, Y. Chen, "Power-efficient video encoding on resource-limited systems: A game-theoretic approach," *Future Generation Computer Systems*, vol. 28, issue 2, pp. 427-436, 2012.

[31] C. S. Kannangara, I. E.Richardson, and A. J. Miller, "Computational complexity management of a real-time H.264/AVC encoder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, issue 9, pp. 1191-1200, 2008.

[32] W. Kim, J. You, J. Jeong, "Complexity control strategy for real-time H.264/AVC encoder," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 2, pp. 1137-1143, 2010.

[33] H. Lee, B. Jung, J. Jung, and B. Jeon, "Computational complexity scalable scheme for power-aware H.264/AVC encoding," *IEEE International Workshop on Multimedia Signal Process*, pp. 1-6, 2009.

[34] H. Yu, L. Qing, S. Ma, and C.-C. Jay Kuo, "Joint rate-distortion-complexity optimization for H.264 motion search," *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1949-1952, Toronto, ON, 2006.

[35] C.E Rhee, J. Jung, and H. Lee, "A real-time H.264/AVC encoder with complexity-aware time allocation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 12, pp. 1848-1862, 2010.

[36] Y. Chen, T. Chen, C. Tsai, S. Tsai and L. Chen, "Algorithm and architecture design of power-oriented H.264/AVC baseline profile encoder for portable devices," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 8, pp. 1118-1128, 2009.

[37] R. Vanam, E. A. Riskin, and R. E. Ladner, "H.264/MPEG-4 AVC Encoder Parameter Selection Algorithms for Complexity Distortion Tradeoff," *IEEE Data Compression Conference*, pp. 372-381, Snowbird, Utah, 2009.

[38] R. Vanam, E. A. Riskin, R. E. Ladner, S. S. Hemami, "Fast algorithms for designing nearly optimal lookup tables for complexity control of the H. 264 encoder," *Signal, Image and Video Processing*, pp. 1-13, 2012.

[39] G. Corrêa, P. Assunção, L. Agostini, L. A. da Silva Cruz, "Performance and Computational Complexity Assessment of High-Efficiency Video Encoders," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1899-1909, 2012.

[40] X. Li, Y. Cui, Y. Xue, "Towards an Automatic Parameter-Tuning Framework for Cost Optimization on Video Encoding Cloud," *International Journal of Digital Multimedia Broadcasting*, 2012.

[41] J. Kim, J. Kim, G. Kim, S. Na, and C.-M. Kyung, "Event statistics and criticality-aware bitrate allocation to minimize energy consumption of memory-constrained wireless surveillance system," *IEEE International Conference on Multimedia and Expo*, pp. 7-12, 2010.

[42] J. Kim, J. Kim, G. Kim, and C.-M. Kyung, "Power-rate-distortion modeling for energy minimization of portable video encoding devices," *Midwest Symposium on Circuits and Systems*, pp. 1-4, 2011.
[43] X. Gao, K. M. Lam, L. Zhuo and L. Shen, "Complexity scalable control for H.264 motion estimation and mode decision under energy constraints," *Signal Processing*, vol. 90, no. 8, pp. 2468-2479, 2010.
[44] M.-Y. Chiu and W.-C. Siu, "Computationally-Scalable Motion Estimation Algorithm for H.264/AVC Video Coding," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 2, pp. 895–903, 2010.
[45] X. Li, M. Wien, and J.-R. Ohm, "Medium-granularity computational complexity control for H.264/AVC," *Picture Coding Symposium*, pp. 214-217, 2010.
[46] Z. Ma, H. Hu, Y. Wang, "On complexity modeling of H.264/AVC video decoding and its application for energy efficient decoding," *IEEE Transaction on Multimedia*, Issue 6, vol. 13, pp. 1240-1255, 2011.
[47] W. Pu, Y. Lu, and F. Wu, "Joint power-distortion optimization on devices with MPEG-4 AVC/H.264 codec," IEEE International Conference on Communications, pp. 441–446, 2006.
[48] X. Li, M. Wien, J.-R. Ohm, "Rate-complexity-distortion optimization for hybrid video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 7, pp. 957–970, 2011.
[49] M. Horowitz, A. Joch, F. Kossentini and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, issue 7, pp. 704-716, 2003.
[50] N. Kontorinis, Y. Andreopoulos, and M. van Der Schaar, "Statistical Framework for Video Decoding Complexity Modeling and Prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, issue 7, pp. 1000-1013, 2009.
[51] Y. Andreopoulos and M. van der Schaar, "Complexity-Constrained Video Bitstream Shaping," *IEEE Transactions on Signal Processing*, vol. 55, issue 5, pp. 1967-1974, 2007.
[52] E. Akyol and M. van der Schaar, "Complexity Model Based Proactive Dynamic Voltage Scaling for Video Decoding Systems," *IEEE Transactions on Multimedia*, vol. 9, no. 7, pp. 1475-1492, 2007.
[53] Y. Andreopoulos and M. van der Schaar, "Adaptive Linear Prediction for Resource Estimation of Video Decoding," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 17, issue 6, pp. 751-764, 2007.
[54] Z. Ma and Y. Wang, "Complexity modeling of scalable video decoding," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1125-1128. Las Vegas, NV, 2008.
[55] J. Lou, A. Jagmohan, D. He, and L. Lu, "H.264 Deblocking Speedup," *IEEE Transactions on Circuits and Systems*, vol. 19, issue 8, pp. 1178-1182, 2009.
[56] A. Ray and H. Radha, "Complexity-Distortion Analysis of H.264/JVT Decoders on Mobile Devices," *Picture Coding Symposium*, pp. 65-70, San Francisco, CA, 2004.
[57] Y. Wang and S. F. Chang, "Complexity adaptive H.264 encoding for light weight stream," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. II25–II28, 2006.
[58] H. M. Nam, J. Y. Jeong, K. Byun, J. O. Kim, and S. J. Ko, "A Complexity Scalable H.264 Decoder with Downsizing Capability for Mobile Devices," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 2, pp. 1025 - 1033, 2010.
[59] S. Lee and C. J. Kuo, "Motion Compensation Complexity Model for Decoder-Friendly H.264 System Design," *IEEE Workshop on Multimedia Signal Processing*, pp. 119-122, Crete, Greece, 2007.
[60] S. Lee, "H.264/AVC Decoder Complexity Modeling and its Applications," PhD Thesis, University of Southern California, 2008.
[61] S. Lee and C. J. Kuo, "Complexity Modeling of Spatial and Temporal Compensations in H.264/AVC Decoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 5, pp. 706–720, 2010.
[62] Z. Ma, Z. Zhang, Y. Wang, "Complexity modeling of H.264 entropy decoding," *IEEE International Conference on Image Processing (ICIP)*, pp. 2784-2787, 2008.
[63] S. W. Lee and C.-C. Jay Kuo, "H.264/AVC entropy decoder complexity analysis and its applications," *Journal of Visual Communication and Image Representation*, vol. 22, issue 1, Pages 61-72, 2011.

[64] S. W. Lee and C.-C. Jay Kuo, "Decoder friendly H.264/AVC deblocking filter design," *Proceedings of SPIE*, vol. 7798, 2010.

[65] M. Van Der Schaar, and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *IEEE Transactions on Multimedia*, vol. 7, issue 3, pp. 471-479, 2005.

[66] T. Burd and R. Broderson, "Processor design for portable systems," *Journal of VLSI Signal Processing*, vol. 13, no. 2, pp. 203–222, 1996.

[67] H.264/AVC JM Reference Software version 16.2, available at: http://iphome.hhi.de/suehring/tml/download/old_jm/, last retrieved on July 2013.

[68] x264 Reference Software version 1.1, available at:http://x264.nl/, last retrieved on July 2013.

[69] Intel, Intel VTune Performance Analyzer. Available at: http://software.intel.com/en-us/articles/intel-vtune-amplifier-xe/, last retrieved on July 2013.

[70] T. Wiegand and B. Girod, "Lagrange multiplier selection in hybrid video coder control," *IEEE International Conference on Image Processing*, vol. 3, pp. 542–545, 2001.

Abstract

With recent advances in computing and communication technologies, ubiquitous access to high quality multimedia content is a fact of our daily life. However, resource consumption is still a major concern, specially for portable devices. Today, any user is able to produce and consume multimedia contents using a connected portable device such as a smartphone or a tablet. In many applications, a multimedia content has to be sent to a vast range of receivers with different characteristics. To do so, a simple and affordable method is to generate a single layer bitstream and adapting it, through a set of middle nodes, to meet each receiver's characteristics and its resource limitations. In addition, with the advances in computing technologies, a consumer (decoder) of video content is potentially a producer (encoder) of this content in many occasions. As a result, constraints such as power consumption, computational power and memory limitations should be considered for most multimedia devices during the encoding phase, as well.

In this thesis, we start by highlighting the necessity of having a resource aware encoder. Then, this general problem is formulated in terms of an abstract method, and customized and solved for the special cases of receiver aware encoder, receiver aware encoder for adaptation applications, and self resource aware encoder. In a receiver aware encoder the resource limitations of the receiver side are taken into account during the encoding phase at the server side. This will ensure that the generated bitstream will satisfy the receiver's resource limitations during decoding. A receiver aware encoder for adaptation applications is an encoder which not only considers the receivers resource limitations but also considers the characteristics of the adaptation node and its operation during the encoding phase. Hence, generates a bitstream that meets the decoder requirements even after adaptation. Finally, a self resource aware encoder refers to an encoder which is running on a device with limited resources, such as a video encoder on a smartphone. Such an encoder is able to encode video while optimizing its operations according to the available resources.

The proposed abstract model is extensible, since it can be used for various combinations of resource limitations. In addition, it follows a methodology for extracting the proper subset of encoding parameters. In order to show the efficiency of the proposed approach, a case study is implemented for the reference software of H.264/AVC codec, for each of the three mentioned problems. In these case studies the computational complexity of decoder/encoder is considered as the limited resource.

Finally, in the second part of the thesis we propose a methodology for estimating resource consumption. This methodology was used for the implementation and evaluation of the case studies of our first contribution. In this context and in order to evaluate the performance of the proposed methodology, we have estimated the complexity of an H.264/AVC decoder and encoder.

Simulations results of all of the case studies show the applicability and the performance of the proposed abstract level resource aware encoder as well as the resource estimation methodology. Both of the proposed approaches in this thesis are generic and can be used for a wide range of applications. For instance, the proposed abstract level resource aware encoder can be customized, for any application which produces or consumes video content and has limited resources, in order to design a resource aware encoder. Similarly, for any application with a clearly specified algorithm and a distinctive resource consumption metric, a resource estimation model can be derived using the proposed resource estimation methodology.

University of Tehran

College of Engineering

School of Electrical and Computer Engineering

Title:

An H.264/AVC Receiver Aware Video Encoder for Adaptation Applications

By:
Mehdi Semsarzadeh

Supervisor:
Dr. Mahmoud Reza Hashemi

Advisor:
Dr. Shervin Shirmohammadi

A thesis submitted to the Graduate Studies Office in partial fulfillment of the requirements for the degree of PhD
In

Computer Engineering – Computer Architecture

September 2013