تکنیک فشرده سازی تصاویر ثابت

اولین تکنیک فشرده سازی، تکنیک فشرده سازی تصاویر ثابت می باشد که متداولاً آن را JPEG می نامند. JPEG مخفف Joint Photographic Expert group است که یک فرمت فایل یا یک روش فشرده سازی برای تصاویر ثابت است. به دلیل کارکرد بسیار مناسب این روش می توان گفت که JPEG به عنوان یک استاندارد فشرده سازی مطرح شده است. روش JPEG دارای چهار مد اصلی و مجموعه بسیار زیادی از انتخاب‌های مختلف و قابل تنظیم می باشد و می‌توان به وسیله آنها وضعیت فشرده سازی را کنترل کرد. در مطالب بعدی عنوان خواهد شد، که روش JPEG مبنای روش MPEG در فشرده سازی ویدئو است. اگر بخواهیم مراحل روش JPEG را به اختصار مورد بررسی قرار دهیم: در مرحله اول، مدل رنگ تصویر ثابت ورودی تغییر خواهد کرد. بدین معنی که تصویر ورودی از مدل رنگRGB که یک مدل متداول است به مدل YIQ منتقل خواهد شد. این مدل یعنی مدل YIQ استاندارد مد NTSC است.

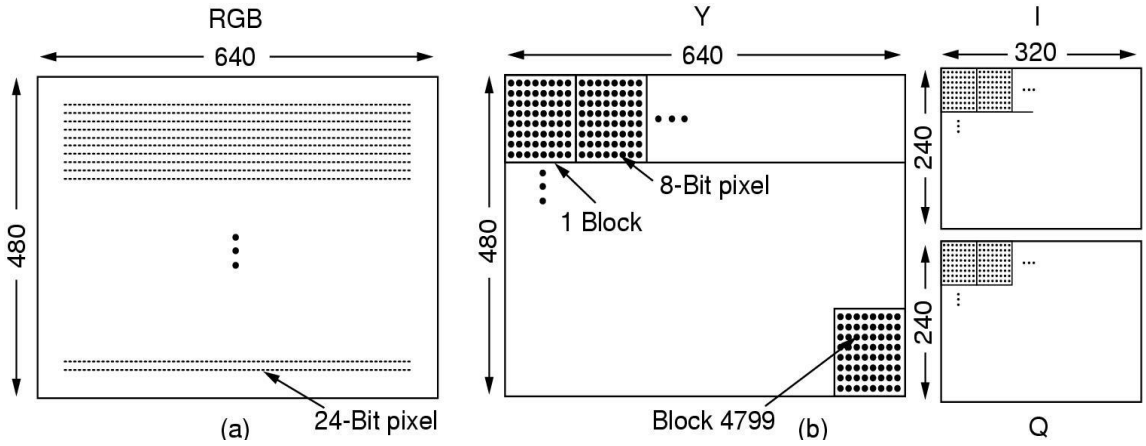
$$R,G,B,Y \in [0,1], \quad I \in [-0.5957, 0.5957], \quad Q \in [-0.5226, 0.5226]$$

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.595716 & -0.274453 & -0.321263 \\ 0.211456 & -0.522591 & 0.311135 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.9563 & 0.6210 \\ 1 & -0.2721 & -0.6474 \\ 1 & -1.1070 & +1.7046 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

برای تبدیل مدل RGB به YIQ می توانیم از تکنیک ساده‌ای استفاده کنیم برای این منظور کافی است ماتریس ۳×۳ ای که در شکل نشان داده شده است را در بردار مربوط به سه رنگ R، G و B ضرب نماییم تا مؤلفه‌های I، Y و Q بدست آید. مولفه‌هایI و Q معمولاً Chrominance رنگ ومؤلفه Y، Luma یا Luminance را نشان می دهد. به غیر از مدل های رنگی RGB و YIQ، می توان از مدل های رنگ دیگری مانند HSI نیز نام ببرد که در مباحث مربوط به پردازش تصاویر رنگی مورد استفاده قرار می گیرد. در هر حال ذکر این نکته نیز خالی از لطف نیست که تبدیل RGB به YIQ یک تبدیل برگشت پذیر است و می‌توان با داشتن مؤلفه های

YIQ مجدداً مؤلفه های RGB را بدست آورد. برای این کار از ماتریس ۳×۳ دیگری که در اینجا نشان داده شده است استفاده می‌کنیم. پس از اینکه تصویر ورودی را به فضای YIQ منتقل کردیم سعی می کنیم که بلاک هایی را در تصویر ایجاد کنیم. پس از اینکه تصویر ورودی را به فضای YIQ منتقل کردیم سعی می کنیم که بلاک هایی را در تصویر ایجاد کنیم. به این شکل که بلاک هایی شامل ٤ پیکسل در تصویر ایجاد کرده و میانگین این ٤ پیکسل در ۲مؤلفه I و Q را جایگزین این ٤ پیکسل خواهیم کرد. این مسئله باعث خواهد شد، سایز تصویر ورودی کاهش یابد. بنابراین اگر فرض کنیم که تصویر ورودی یک تصویر ٤٨٠× ٦٤٠ است. آنگاه با انجام این کار تصویر به یک تصویر ٢٤٠× ٣٢٠ تبدیل خواهد شد. انجام این کار خود نشان دهنده این است که این روش یک روش Lossy است چرا که با داشتن میانگین ٤ عدد، بدست آوردن دقیق آن اعداد لزوماً ممکن نخواهد بود.



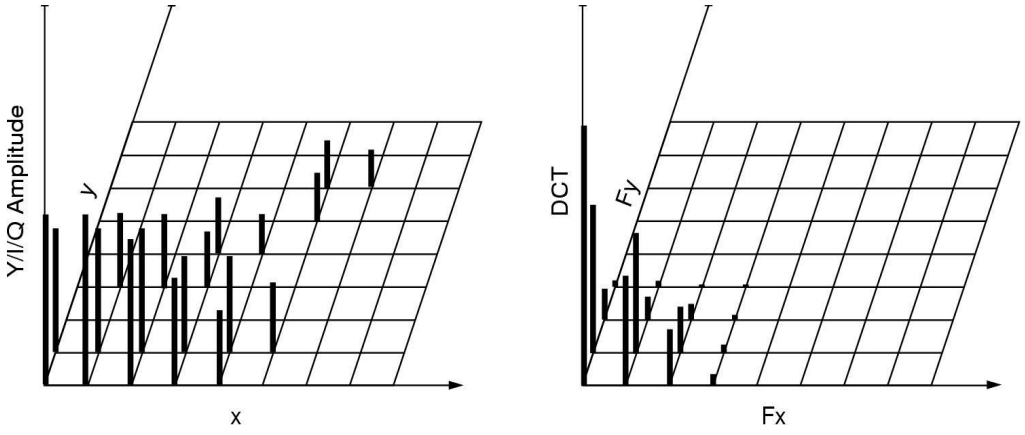
در مرحله دوم مقدار ۱۲۸ را از تمام المان های همه ماتریس ها کم می‌کنیم. در اینجا منظور از همه ماتریس ها، ماتریس y، ماتریس I و ماتریس Q است. سپس در مرحله سوم هر ماتریس به بلاک های ۸×۸ تقسیم بندی می شود. این مسئله در شکل بالا نشان داده شده است. در این شکل می بینید، که یک تصویر ٤٨٠×٦٤٠ ابتدا بلاک بندی شده است و سپس با استفاده از ایده کاهش، تصاویر ٢٤٠×٣٢٠ بدست آمده است و آنگاه بر روی هر یک از این تصاویر که هر تصویر نشان دهنده مقدار یک مؤلفه است بلاک های ۸×۸ ایجاد شده است.

Discrete Cosine Transformation

مرحله چهارم در فشرده سازی JPEG استفاده از تبدیلDCT است. تبدیل DCT یا Discrete Cosine Transformation بر روی هر بلاک اعمال می شود و از نتایج این تبدیل استفاده می‌گردد.

می دانیم که DCT در تئوری یک روش Lossless است. به این معنی که می توان نتایج بدست آمده از DCT را مجدداً به گونه ای با هم ترکیب کرد تا داده ورودی به آن بدست آید، یعنی عمل DCT معکوس بر روی داده ها گرفته شود.

معمولاً در عمل DCT یک روش Lossy است به این دلیل که در روند استفاده از DCT استفاده از Floating Point ها ممکن است اتفاق بیفتد. Floating Point ها یا اعداد ممیز شناور نمی توانند کلیه بازه اعداد حقیقی را بپوشانند و حتماً رزولوشن خاصی برای آنها تعریف شده است. و همین مسئله باعث می‌شود که خطاهای Rounding رخ داده و DCT دقیقاً و صددرصد در محیط دیجیتال برگشت پذیر نباشد.



این مسئله در شکل نشان داده شده است. با توجه به شکل به دلیل خطاهای مربوط به Quantization عمل DCT به طور صددرصد برگشت پذیر نیست. پس از این کار در مرحله پنجم ضرایب DCT را Quantized می کنیم، بدین معنی که همه ضرایب DCT بر یک مقدار خاصی تقسیم می شود.

برای این منظور از یک ماتریس Quantization استفاده می‌کنیم که این ماتریس Quantization مقادیر مختلفی را می تواند بپذیرد. یکی از چالش های مهم یا یکی از نکات مهم در روش JPEG طراحی و داشتن یک ماتریس Quantization مناسب است که مقادیر این ماتریس Quantization لزوماً در استاندارد JPEG قرار ندارد.

DCT Coefficients

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized coefficients

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization table

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

همانطور که در شکل می بینید ماتریس Quantization مورد نظر دارای برخی خواص است. از جمله این موارد این است که هر چه شما از نقطه (۰و۰) ماتریس به سمت نقطه انتهایی حرکت می‌کنید ضرایب مربوط به ماتریس افزایش پیدا کرده است. این امر باعث می شود، هنگامی که مقادیر یا ضرایب DCT بر روی ماتریس Quantization اعمال می‌شود و خروجی به شکل یک Integer دیده شود، آنگاه ضرایب Quantized شده در فرکانس های بالا مقدار صفر بگیرد.

در ضرایب Quantized شده مقادیر صفر زیادی قرار دارد که این مقادیر صفر در فرکانس‌های بالا هستند. و این نکته بسیار مهم است؛ زیرا فرکانس های بالا معمولاً شامل اطلاعاتی هستند که چهارچوب اصلی تصویر را نمی‌سازند، به عبارت دیگر چهارچوب اصلی تصویر معمولاً بر اساس اطلاعات موجود در فرکانس‌های پایین تصویر است و این مسئله باعث می‌شود که حذف ضرایب فرکانس‌های بالا سیستم را آماده فشرده سازی کند، به گونه ای که تغییرات چندانی در تصویر ایجاد نشده و تنها برخی از جزئیات تصویر از بین رفته باشد. بدیهی است که با تغییر مقادیر موجود در ماتریس Quantization ، تعداد ضرایبی که از ضرایب DCT به سمت

صفر تغییر مقدار می دهند افزایش پیدا خواهد کرد. همین امر نشان می‌دهد که روش JPEG قابلیت کنترل سطح فشرده سازی را دارد. یکی از پارامترهایی که معمولاً در روش‌های مبتنی بر JPEG مطرح می شود ارائه ضریبی است که با استفاده از آن سطح فشرده سازی روش تنظیم خواهد شد. بدیهی است که این ضریب اثر مستقیم در مقادیر موجود در جدول Quantization دارد.

پس از انجام Quantization و در گام ششم کاری که انجام می شود این است که به ازای هر بلاک (Block) مقدار موجود در سلول (۰و۰) آن Block که پس از گرفتن تبدیل DCT و انجام عمل Quantization بدست آمده است با مقدار تفاوت آن از بلاک قبلی جایگزین می شود.

در مرحله هفتم داده‌ها به صورت خطی نمایش داده می شوند. برای نمایش داده ها به صورت خطی، از تبدیل زیگ زاک یا zig zag Transform استفاده می کنیم.

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

نحوه انجام تبدیل زیک زاک در شکل نشان داده شده است. همانطور که در شکل نشان داده شده است، این تبدیل داده‌ها را به گونه‌ای می‌خواند که چنانچه روند خواندن آنها را دنبال کنیم داده ها همگی در یک خط قرار دارند. پس از اینکه با استفاده از zig zag Transform داده ها خطی شدند آنگاه RLC یا Run Length Coding روی آنها اعمال می شود سپس عملیات huffman Coding را بر روی این مجموعه داده ها اعمال می نماییم و خروجی huffman Coding در واقع مجموعه ضرایبی را می سازد که ما انتظار داریم که این ضرایب یک تصویر فشرده شده را نمایش دهد.

فشرده سازی ویدئو (Video Compression)

هدف، در مبحث Video Compression فشرده سازی ویدئو است. با توجه به مباحث قبل، روش JPEG که برای اعمال بر روی تصاویر ثابت به کار می رود مبنای روش دیگری است که آن روش دیگر برای فشرده سازی ویدئو به کار می رود. روش فشرده سازی ویدئو را به عنوان MPEG می‌شناسیم.MPEG یا Motion Photographic Expert Group یک روش فشرده سازی ویدئو است که ورژن های مختلفی در آن وجود دارد. MPEG1، MPEG2، MPEG4 و حتی MPEG7 ورژن های مختلفی از MPEG هستند که هر یک خواص و توانمندی های خاص خود را دارند.

MPEG1 برای Video Recorder استفاده می شود یعنی در دستگاههای ویدیوئی که کار ضبط ویدیوئی را بر عهده دارند به کار برده می شود و معمولاً کیفیت مناسبی دارد. خروجی MPEG1 معمولاً تصویری با حجم 1.2 Mbps خواهد بود که در استاندارد و Resolution (۳۵۲×۲٤۰) قرار دارد و این رزولوشن بکار گرفته شده در سیستم NTSC است.

MPEG2 روش دیگری است که معمولاً برای ویدیوهای باکیفیت بالا که به صورت Broad Cast به کار برده می شوند استفاده می شود. روش MPEG2، 4-6 Mbps پهنای باند را اشغال می کند. و معمولاً در سیستم های NTSC یا PAL به کار برده می شوند. برای ورژن های جدیدتر MPEG و خصوصیات آنها می‌توانید به کتاب های موجود در این زمینه مراجعه نمایید.

اما نکته ای که به طور کلی در تمام گونه های MPEG رعایت می شود این است که در تمامی این گونه‌ها سعی می شود که از مزایا و سودمندی دو نکته اساسی در تصاویر ویدیوئی استفاده شود. اولین نکته، افزونگی در سطح مکان است، که معمولاً افزونگی است که به ازای هر Still image یا تصویر ثابت در سیستم وجود خواهد داشت. دومین نوع افزونگی، افزونگی در حوزه زمان است و معمولاً در این نوع افزونگی زمانی که کاربرد ویدیوئی دارد سعی می شود ارتباط بین فریم‌های مختلف متوالی یا غیرمتوالی مورد بررسی قرار گیرد.

در حالت اول، اگر بخواهیم MPEG را با یک دید سطح بالا نگاه کنیم، خواهیم گفت که در این حالت Coding یا فشرده سازی هر فریم مستقلاً با استفاده از JPEG صورت می‌گیرد. اما در مبحث مربوط به افزونگی زمانی، فریم های پشت سر هم کسانی هستند که این افزونگی را تعیین می کنند. معمولاً ایده اصلی این نوع افزونگی این است که فریم‌های پشت سر هم غالباً مشابه یکدیگر هستند و این امر باعث بهبود وضعیت فشرده سازی ویدیو خواهد شد.

بنابراین تفاوت اساسی بین JPEG و MPEG بحث Motion compensation یا سازگار کردن حرکت است. چنانچه حالتی را در نظر بگیرید که یک شیئ بسیار ساده و با حرکتی آهسته در یک تصویر را داشته باشید، در این حالت اعمال کردن تنها JPEG بر روی تفاوت هردو فریم، جواب بسیار خوبی تولید خواهد کرد. اما مشکلی که در چنین حالتی روی خواهد داد زمانی خواهد بود که دوربین با استفاده از تکنیک‌های Pan و Tilt و Zoom فضای داده مشاهده شده را تغییر دهد و حرکتی هر چند به صورت مجازی ایجاد کند. MPEG سعی می‌کند که این نوع حرکت‌ها را با یکدیگر سازگار کند.

روش MPEG2 شامل سه نوع فریم مختلف است که توسط یک برنامه نمایش دهنده به کار گرفته می شود. این سه فریم که در روش MPEG2 وجود دارد عبارتند از:

۱. Intracoded frames
۲. Predictive frames
۳. Bi-directional frames

فریم های Intracoded فریم‌هایی هستند که می‌توانند به صورت محلی با الگوریتم JPEG فشرده شوند. یا به عبارت دیگر این فریم ها به صورت محلی با الگوریتم های مبتنی بر JPEG فشرده می شوند.

Predictive frames فریم های پیش گویی هستند که تفاوت بلاک به بلاک با فریم قبلی را در نظر گرفته و بر اساس آن برنامه دارند.

Bi-directional frames تفاوت بین داده‌های مشاهده شده در فریم قبلی و فریم بعدی را نمایش خواهد داد. بنابراین برای اینکه بتواند این تفاوت را برای فریم‌های بعدی نیز لحاظ نماید حتماً برای پیاده سازی این سه نوع فریم نیازمند ابزارهای مناسبی هستیم.

I frames

I frames فریم‌هایی هستند که به صورت پریودیک در هر ثانیه یک بار و یا در هر ثانیه دوبار در خروجی قرار می گیرند. علت اینکه در روش MPEG استاندارد، فریم های I به حالت پریودیک در هر ثانیه در خروجی قرار می گیرد را می توان در سه مورد بررسی کرد:

اول اینکه اگر همه فریم ها وابسته به آخرین فریم باشد چنانچه کسی اولین فریم را از دست بدهد امکان نمایش و دنبال کردن فیلم برای آن مقدور نیست. دوم اینکه اگر فریمی با خطا در سمت گیرنده دریافت شود و بتوان فریم های بعدی را نیز نمایش داد و سوم اینکه عملیات مربوط به Forward و Backward یعنی جلو رفتن و عقب رفتن در سطح فیلم آسان تر انجام شود. سه مورد فوق را به عنوان سه دلیل اصلی این مسئله می دانند که I frame ها باید به صورت پریودیک داده های مربوط به خود را بر روی کانال خروجی قرار دهند.

P frames

بعد از بررسی I frame ها که به عنوان یکی از اجزای اصلی روش فشرده سازی MPEG مورد بررسی قرار می گیرد، به بررسی P frame ها می‌پردازیم. می توان P frame ها را به عنوان ماکروبلاک های موجود در تصویر شناخت. به عبارت دیگر P فریم ها، ماکروبلاک هایی هستند که معمولاً اندازه آنها ۸x ۸ یا ١٦×١٦ است. یک ماکروبلاک، بلاکی است که سعی می شود مشابه آن در فریم قبلی پیدا شود. به عبارت دیگر P frame یک ماکروبلاک کد شده است که با جستجوی فریم قبلی بدست آمده است. بنابر این هدف از P frame پیدا کردن بردار حرکتی یک بلاک است. به این منظور یک بلاک را در فریم جاری استخراج کرده و سعی می کنیم در فریم قبلی مکان آن بلاک را پیدا کنیم. پیدا کردن مکان یک بلاک در

فریم قبلی ممکن است به صورت انطباق کامل و یا انطباق با تفاوت‌های اندک باشد. در این حالت پس از پیدا کردن بلاک مورد نظر در فریم قبلی بردارهای حرکتی قابل استخراج خواهد بود. آنچه که در استاندارد MPEG باید به آن دقت کرد این است که در این روش استاندارد، به برخی از مسائل توجهی نشده است و یا به عبارت دیگر MPEG مشخص نکرده است که چطور و با چه قابلیتی و در چه فاصله‌ای باید برای پیدا کردن یک بلاک یا متناظر با یک ماکروبلاک در فریم قبلی عملیات جستجو را انجام داد.

بنابراین اگر ماکروبلاک پیدا شود، تفاوت ماکروبلاک و بردار حرکتی آن توسط روشی مشابه روش JPEG فشرده می شود. اما اگر پیدا نشود اتفاقی که خواهد افتاد این خواهد بود که فریم جاری با روش JPEG مشابه یک فریم I فشرده سازی و ارسال خواهد شد.

B Frames

فریم های B را نیز می توان مشابه فریم های P دانست. سعی می کنیم ماکروبلاک را در فریم های قبلی یا فریم های بعدی به کمک فریم های B مدل کرده و ارسال نماییم. بنابراین برای کد کردن یک فریم B، Encoder نیاز دارد که سه فریم decode شده با نام‌های فریم قبلی، فریم جاری و تخمینی از فریم آینده یا فریم بعدی را در اختیار داشته باشد. با توجه به این سه فریم و داشتن اطلاعات مربوط به این سه فریم، فریم B ساخته می شود. فریم B جهت حرکتی یک ماکروبلاک را تعیین می کند تا بتوانیم در سمت دیکدر بازسازی مناسبی از رفتار حرکتی داشته باشیم. یادآوری می‌کنیم که یکی از نکات مهم در کدگذاری MPEG بحث Motion Compensation بود که فریم‌های P و B وظیفه این کار را در این روش بر عهده داشتند.

پس از شناخت نسبی که از نحوه شناخت Encoding اجزای اصلی سیستم چند رسانه‌ای بدست آوردیم حال به بررسی مسئله زمان بندی در فرایندهای چند رسانه ای می‌پردازیم. در اینجا سعی می کنیم که تکنیک‌ها و الگوریتم‌های زمان بندی که عموماً در سیستم های چند رسانه ای مورد توجه قرار می گیرند را بررسی کنیم. قبل از اینکه فرضیات مربوط به این نوع زمان بندی ها را بیان کنیم، ذکر این نکته ضروری است که سیستم‌های

چند رسانه‌ای را معمولاً با عنوان سیستم‌های بلادرنگ نرم در نظر می گیرند. بنابراین مفاهیم سیستم های بلادرنگ نرم در سیستم های چند رسانه‌ای مورد توجه است و الگوریتم های زمان بندی این نوع سیستم ها در اینجا قابل استفاده خواهد بود. حال به بررسی فرضیاتی که در طول انجام زمان بندی فرایندهای چند رسانه ای حائز اهمیت است می پردازیم. ابتدا فرض می کنیم که ما یک سرویس دهنده ویدیو داریم که شامل خصوصیات زیر است:

۱- تعداد مشخص و ثابتی از فیلم ها را می تواند نمایش دهد.

۲- تمامی این فیلم‌ها از نرخ فریم ثابتی برخوردار هستند.

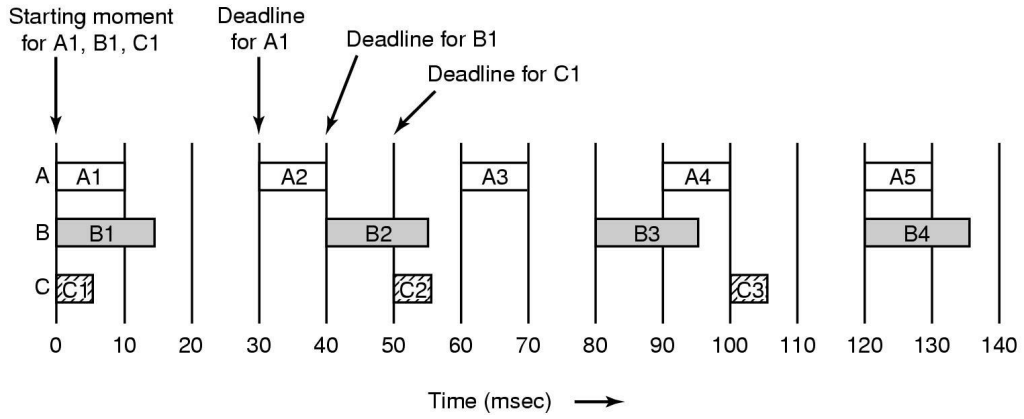
۳- Video resolution آنها یکسان است.

٤- نرخ ارسال داده‌ها و سایر پارامترهای آنها نیز یکسان است.

حال برای زمان بندی یک چنین سیستمی یک الگوریتم کارای زمان بندی ساده به صورت زیر تعریف می‌شود. به ازای هر فیلم، یک فرایند پردازشی داریم، این فرایند داده‌های یک فریم را می‌خواند و آنها را به سمت کاربر ارسال می کند و همه فرایندها اهمیت یکسان دارند. فرایندها مقدار کار یکسانی در هر فریم انجام می دهند. فرایندها وقتی فریم جاری خود را به پایان رساندند یعنی آن را خوانده و ارسال کردند، متوقف می شوند.

اولین تکنیک زمان بندی که برای زمان بندی فرایندهای این گونه می‌توان عنوان کرد، تکنیک زمان بندی RR یا Round Robin است. در زمان بندی RR نیاز به یک مکانیزم Timing یا زمان دهی برای تصمیم گیری و حصول اطمینان از اینکه هر فرایند با نرخ مناسب و در مدت زمان مناسب فعالیت خود را انجام داده است می‌باشد. به این معنی که در روش RR هر فرایند در قالب پروتکل FIFO زمان بندی می شد. اما بر خلاف پروتکل FIFO که یک روش انحصاری است، RR یک روش غیرانحصاری بود که هر فرایند تنها یک برش زمان، CPU را در اختیار داشت. برای اجرای الگوریتم RR نیاز به یک ساعت ضروری است و معمولاً با استفاده از یک تکانه ساعت می‌توان اندازه برش زمانی که در اختیار هر فرایند می‌باشد را اندازه‌گیری نمود. و

فرایندها با تکنیک FIFO زمان بندی شده و به صورت ترتیبی با جلو رفتن کلاک ساعت انجام می شوند. در این تکنیک معمولاً اگر فرایندی کامل شود از سیستم خارج می‌شود و اگر فرآیند تنها عملیات مربوط به یک فریم را انجام دهد فرایند معلق می شود. همچنین در این سیستم معمولاً فرض می کنیم که تعداد فرایندها به حد کافی کوچک است به صورتی که زمان بندی آنها به صورت Round Robin بر پارامترهایQOS آنها تأثیر منفی نمی گذارد. اگر سیستمی با چنین مدلی داشته باشیم زمان بندی Round Robin زمان بندی مناسبی است اما این زمان بندی و این مدل اصولاً یک مدل قابل اعمال در محیط واقعی نیست، یعنی در محیط واقعی چنین مدلی قابل استفاده نیست. به دلیل اینکه معمولاً در یک محیط واقعی تعداد کاربران، اندازه فریم ها ورزولوشن‌های مختلفی وجود دارد. لذا وقتی که تعداد فرایندهای مختلفی در سیستم وجود داشته باشد این بدین معنی خواهد بود که فرکانس‌های مختلفی در سیستم برای هر فرایند قابل تصور است و میزان کاری که هر فرایند انجام می‌دهد متفاوت خواهد بود. در نتیجه باید زمان مرگ فرایندها نیز با یکدیگر یکسان نباشد. بنابراین اگر بخواهیم در یک مدل واقعی بحث کنیم و یک مدل عمومی‌تر را ارائه دهیم عنوان خواهد شد که در یک مدل عمومی فرایندهای مختلفی وجود دارند که برای دریافت پردازنده با یکدیگر رقابت می‌کنند، هر یک از آنها کار خود را انجام می دهند و هر یک از آنها خط مرگ خود را دارند. در نتیجه فرکانس رخداد هر یک از این فرایندها متفاوت خواهد بود. در این مرحله، در مورد سیستم‌های Multimedia که شامل صوت و تصویر هستند بحث می‌کنیم . اینگونه سیستم ها دارای فرایندهای پریودیک هستند که در یک پریود مشخص اجرا می‌شوند، اما فرض می‌کنیم که فرکانس یا پریود هر فرایندی مشخص است. حال که چنین شد یعنی به ازای هر فرایندی Dead line به صورت پریودیک در نظر گرفته شد، می توان عنوان کرد که این سیستم یک سیستم soft - real timeاست. بنابراین تکنیک های real time در اینجا قابل استفاده است.



به عنوان مثال به این شکل دقت کنید در این شکل ما سه فرایند A,B,C را در اختیار داریم که هر سه فرایند، فرایندهای پریودیک است. هر سه فرایند به طور همزمان در لحظه صفر آماده اجرای عملیات مربوط به خود می‌باشند. نکته‌ای که این شکل آن را به خوبی نشان می دهد این است که در یک فرایند پریودیک معمولاً خط مرگ مربوط به یک task را به عنوان نقطه ای در نظر می‌گیرند که Task دوم فرایند یا به عبارت بهتر Task بعدی فرایند، وارد سیستم می‌شود. بنابراین اگر خط مرگ task، A1 از فرآیند A را در نظر بگیریم با توجه به شکل و با توجه به پریود A که هر 30msec یک بار تکرار خواهد شد Dead line، A1 زمان ۳۰ خواهد بود. همین اتفاق برای فرایند B و Task B1 می‌افتد از آنجا که B دارای پریودی معادل هر 40msec یکبار است، بنابراین خط مرگB1، ٤٠ خواهد بود. همانطور که می دانید فرکانس عکس پریود است. بنابراین اگر B دارای پریود ٤٠ است فرکانس آن ۲۵ بار در ثانیه خواهد بود. همین در مورد C نیز صادق است و بنابراین خواهیم دید که گرچه این فرایندها همگی با هم آغاز شده اند اما Taskهای بعدی آنها نسبت به یکدیگر اختلاف زمان آغاز خواهند داشت. حال که می توان برای این فرایندها از تکنیک های Real time استفاده کرد، به چند نکته از سیستم های Real time اشاره میشود. با توجه به مباحث مربوط به سیستم های Real time می دانید که فرایندهای Real time قابل زمان بندی هستند اگر رابطه زیر برای آنها برقرار باشد: یعنی اگر داشته باشیم:

$$\sum_i \frac{C_i}{P_i} \leq 1$$

C_i زمان پردازشی لازم برای یک فرایند و P_i پریود آن فرایند است.

به عنوان مثال در شکل قبل زمان پردازشی فرایند A ، 10msec و پریود آن 30msec بود. نکته دیگر در سیستم های Real time این است که در این نوع سیستم ها فرایندها می‌توانند انحصاری یا غیرانحصاری باشند یعنی الگوریتم زمان بندی ممکن است یک فرایند را قبل از اینکه task مربوط به آن تکمیل شده باشد cpu را از وی بگیرد و یا ممکن است که این کار را انجام ندهد. از سوی دیگر الگوریتم های Real time می‌توانند ایستا یا پویا باشند. یعنی زمان بندی آنها می تواند به صورت پویا یا ایستا انجام شود، یعنی بر اساس یک معیار از قبل تعیین شده و یا بر اساس معیاری که در زمان اجرای فرایندها ممکن است تغییر کند.

الگوریتم RMS (Rate Monotonic Scheduling)

یکی از متداول ترین الگوریتم‌های زمان بندی سیستم‌های Real time را با عنوان Rate Monotonic Scheduling یا RMS می شناسیم. RMS برای فرایندهایی به کار می رود که شرایط زیر را دارا هستند.

۱- هر فرایند پریودیک باید در داخل پریود خود تکمیل شود.

۲- شماره فرایند، وابسته به هیچ فرایند دیگری نیست.

۳- هر فرایند در هر بار اجرا به زمان CPU یکسان نیاز دارد. به عبارت دیگر Taskهای مختلف یک فرایند نیاز یکسانی به CPU دارند و مقدار زمان اجرای Taskها متفاوت نیست.

٤- هر فرایند غیر پریودیک، Deadline ندارد.

٥- قبضه کردن فرایند بصورت آنی و بدون سربار اتفاق می افتد.

لذا اگر فرایندهایی باشند که این خاصیت ها را داشته باشند استفاده از الگوریتم RMS برای آنان امکان پذیر است.

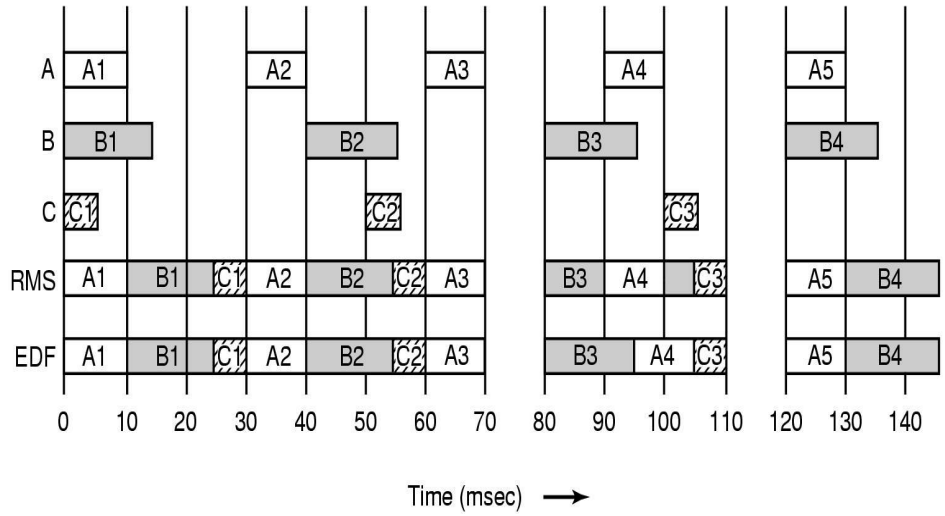
در این الگوریتم هر فرایند یک پریود ایستا دارد. پریود فرایند برابر با فرکانس آن فرایند است. به عبارت دیگر الگوریتم، CPU را از فرایندی را که در حال حاضر کنترل CPU را بر عهده دارد خواهد گرفت، اگر فرایندی با اولویت بیشتر وارد سیستم شود. این موارد کلیات الگوریتم RMS را نشان می‌دهد. الگوریتم RMS، الگوریتمی است که مزایا و معایب خاص خود را دارد و اگرچه در سیستم های Real time قابل استفاده است اما در بعضی موارد زمان بندی‌های انجام شده توسط الگوریتم RMS زمان بندی خوبی نیست.

الگوریتم EDF (Earliest Deadline First)

یکی دیگر از الگوریتم های زمان بندی که در سیستم های Real time به شدت مورد استفاده قرار می‌گیرد و از الگوریتم های معروف این سیستم هاست الگوریتم EDF یاEarliest Deadline First است.

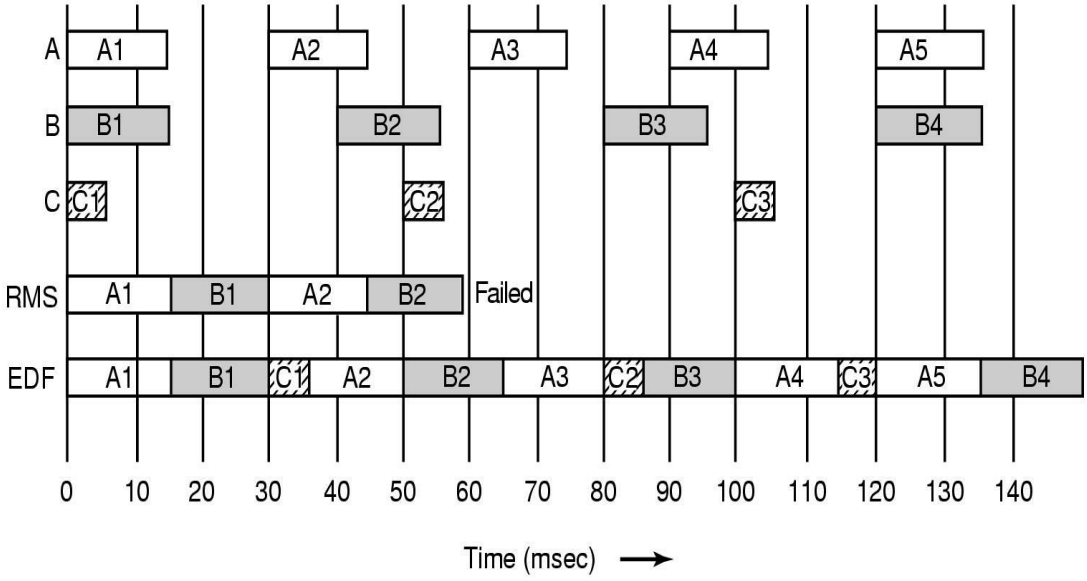
این الگوریتم برخلاف الگوریتم RMS یک الگوریتم پویا است که قابلیت انحصاری بودن را دارد و ورژن غیر انحصاری آن نیز وجود دارد. الگوریتم فرایندی را زودتر اجرا خواهد کرد که Dead line آن نزدیکتر باشد. به این معنی که وقتی یک فرایند جدید وارد سیستم می شود الگوریتم Dead line آن را بررسی می کند و ممکن است CPU از فرایندی که در حال حاضر CPU را در اختیار گرفته شود کنار گذاشته شود و به فرایند جدید داده شود، چنانچه Dead line فرایند جاری از Dead line فرایندی که تازه وارد شده است دورتر باشد. این حالت ورژنی از الگوریتم EDF است که به صورت preemptable عمل می‌کند اما می دانیم که ورژن non preemptableآن به این شکل عمل نخواهد کرد، و حتماً اجرای یک فرایند را کامل می کند و سپس برای انتخاب فرایند بعدی بر حسب Dead line تصمیم خواهد گرفت.

برای این که رفتار دو الگوریتم (RMS) و (EDF) را با یکدیگر مقایسه کنیم به شکل زیر دقت نمائید:



در اینجا فرض می کنیم که سه فرآیند،A,B,C به صورت پریودیک در حال اجرا می باشند. فرآیند A فرآیندی است که در زمان صفر آغاز شده هر ۳۰ میلی ثانیه یک بار اجرا می شود. فرآیند B فرآیندی است که در زمان صفر آغاز شده است و ۲۵ بار در ثانیه اجرا می شود و فرآیند C نیز ۲۰ بار در ثانیه اجرا می‌شود در حالی که زمان آغاز آن برابر با صفر است. الگوریتم (RMS) با توجه به خصوصیات آن اولویت بیشتری بهtask های فرآیند A خواهد داد و لذا ابتدا A1 را اجرا کرده و سپس B1 وc1 اجرا می‌شوند و به همین ترتیب فرآیندها اجرا خواهند شد. نحوه انجام عملیات مربوط به الگوریتم (EDF) نیز تا حدی شبیه (RMS) است. یعنی در مورد (EDF) نیز ابتدا A1 اجرا شده، سپس B1 و C1 و بعد آن A2، B2 و C2اجرا خواهند شد. به عبارت دیگر تا زمان ۷۰ این دو الگوریتم، زمانبندی مشابهی را انجام می دهند. اما در زمان ۷۰ و پس از آنکه در لحظه-ی ۸۰ زمانبندی مجدد انجام می شود، الگوریتم (RMS) ابتدا B3 را اجرا می کند اما به محض اینکه در زمان ۹۰ فرآیند A4 وارد شد، کنترل از B3 خارج شده و فرآیند A4 اجرای خود را آغاز می کند، و مجدداً در لحظه ی ۱۰۰ فرآیند B3 یا به عبارت بهتر وظیفه B3 از فرآیند B فعالیت خود را تکمیل می کند. اما این اتفاق در اینجا در مورد (EDF) نخواهد افتاد و خواهیم دید که (EDF)، B3 را تکمیل کرده و سپس A4 را اجرا می کند، چرا که می دانیم خط مرگ B3در اینجا ازخط مرگ A4 نزدیکتر است و خط مرگ B3 زمان ۱۲۰ بوده در حالی که A4 نیز خط مرگ مشابه دارد. بنابراین بهتر است که اولویت در این مرحله در اختیار وظیفه‌ای باشد که در حال حاضر CPU را در اختیار خود قرار داده است.

به عنوان مثال دیگری از نحوه اجرای دو الگوریتم (RMS) و (EDF) به مثال زیر دقت کنید:



در این مثال فرآیند A مشابه فرآیند A در مثال قبلی است، یعنی فرآیندی است که هر ۳۰ میلی ثانیه یک بار تکرار می شود و زمان پردازش آن در حدود ۱۵ است. فرآیند B نیز در اینجا هر ٤۰ میلی ثانیه یک بار اجرا می‌شود، اما فرض کرده ایم که زمان اجرای آن برابر ۱۵ است، یعنی در مقایسه با مثال قبلی، زمان پردازشی یک Task از فرآیند A افزایش یافته است و فرآیند C نیز هر ۵۰ میلی ثانیه یک بار اجرا می‌شود، در حالی که زمان اجرای هر وظیفه از آن برابر ۵ می باشد. حال الگوریتم (RMS) کماکان اولویت را به فرآیند A خواهد داد؛ لذا ابتدا Task A1 اجرا می‌شود، و پس از آن Task B1 اجرا می‌شود. متأسفانه با توجه به زمان بندی انجام شده در لحظه ی ۳۰ مجدداً Task A2 از فرایند A قابل اجراست، بنابراین الگوریتم (RMS) با توجه به اولویت بالای A، CPU را مجدداً در اختیار A2 قرار می‌دهد و پس از آن B2 را زمان بندی می‌کند و همین روند ادامه خواهد یافت. به عبارت دیگر در اینجا مدعی می‌شویم که (RMS) یک الگوریتم شکست خورده است؛ چرا که

با توجه به مقادیر، هیچ گاه فرآیند C زمان بندی نمی‌شود، اما الگوریتم (EDF) این مشکل را ندارد. در شکل چگونگی نحوه ی زمانبندی الگوریتم (EDF) را مشاهده می‌کنید. این الگوریتم هر سه فرآیند را زمانبندی می‌کند و هیچ فرآیندی دچار Starvation نشده است.

اگر بخواهیم RMS را با EDF مقایسه کنیم می توان گفت که الگوریتم RMS تضمین می کند که حتماً عملیات را به درستی انجام دهد و تمام فرایندها را زمان بندی کند درصورتیکه داشته باشیم

$$\Sigma \frac{C_i}{P_i} \leq m(2^{1/m} - 1)$$

M در این رابطه تعداد فرآیندهاست. الگوریتم (EDF) با توجه به پویا بودن آن، الگوریتم پیچیده‌تری است و همواره کار می‌کند، اگرچه ممکن است در شرایطی برخی از Taskها نیز miss شود. در یک Video server می‌توانایین قاعده کلی را بیان نمود که اگر کارایی CPU کوچکتر از حد مطرح شده برای RMS است، آنگاه از RMS به دلیل آنکه سربار کمتری دارد استفاده کرد و حد مطرح شده برای RMS همان رابطه‌ی مطرح شده در فوق است. در غیر اینصورت استفاده از الگوریتم EDF توصیه می‌شود. به این دلیل Video server این خاصیت را دارد که زمان انجام فرآیندها وابسته به سخت افزار است. بنابراین چنانچه Video server قابلیت های خوبی داشته باشد، زمان انجام فرایندها بر روی آن کاهش خواهد یافت. بنابراین امکان اینکه رابطه ی مزبور برآورده شود و شرایط لازم را داشته باشد فراهم خواهد شد.