باسمه تعالی

سیستم‌های چندرسانه‌ای (۳۴۲-۴۰)
دانشکده مهندسی کامپیوتر
ترم بهار ۱۳۸۸
دکتر حمیدرضا ربیعی

تکلیف شماره ٦: فشرده سازی ویدئو

۱- مقدمه

در آزمایش قبل، روش های فشرده سازی تصویر ثابت بررسی و آزمایش شد. در این آزمایش، فشرده سازی تصویر متحرک یا ویدئو، پیگیری می شود. جدول یک، نرخ دادهٔ مورد نیاز برای ویدئو دیجیتال در ساختارهای مختلف را نشان می دهد. مشخص است که یک ویدئو به حافظه و یا پهنای باند بیشتری نسبت به تصویر، نیاز دارد. برای مثال یک ویدئو با دقت CCIR 601 (معادل با دقت تلویزیون آنالوگ) با نرخ نمونه برداری رنگ 4:2:0، نرخی برابر ۱۲۸ Mbit/s یا ١٦Mbyte/s دارد. بعبارت دیگر، یک فیلم ۲ ساعته، ۱۱۲ Gbyte حافظه اشغال می کند. یک فیلم CIF با کیفیت کمتر (معادل با دقت VHS یا فیلم های VCD) یک چهارم این مقدار را اشغال می کند. واضح است که در مقایسه با تصویر، روش های مؤثرتری برای فشرده سازی داده ها، برای ذخیره سازی یا ارسال ویدئو مورد نیاز است.

استاندارد کردن الگوریتم های فشرده سازی برای ویدئو، برای اولین بار توسط CCTTT برا کنفرانس تلفنی و تلفن تصویری، انجام شده است. یک گروه تخصصی از CCTTT، (که اکنون بنام ITU-T، The International Telecommunication Unit-Telecommunication Sector نامیده می شود) استانداردی برای تکنیک های فشرده سازی ویدئو برای ویدئو کنفرانس مطرح کرده است. این استاندارد در سال ۱۹۹۰ تهیه شده است، بعنوان H.320 شناخته می شود که قسمت مربوط به کد کردن ویدئو آن، H.261 است. این استاندارد برای ویدئو کنفرانس از خطوط ISDN استفاده می کند که دارای پهنای باند p*64 kpbs است که p=1,2,...,30 می تواند باشد. منبع ویدئو یا CIF (۳۰ fps و ۳۵۲*۲۸۸) یا QCIF (۳۰fpc و ١٤٤*١٧٦) است و ویدئو همراه با صدا باید به اندازهٔ مضربی از ٦٤ kbps فشرده شود.

Video Format	Y Size	Color Sampling	Frame Rate (Hz)	Raw Data Rate (Mbps)
HDTV Over air. cable, satellite, MPEG2 video, 20-45 Mbps				
SMPTT296M	1280x720	4:2:0	24P/30P/60P	265/332/664
SMPTE295M	1920x1080	4:2:0	24P/30P/60I	597/746/746
Video production, MPEG2, 15-50 Mbps				
CCIR601	720x480/576	4:4:4	60I/50I	249
CCIR601	720x480/576	4:2:2	60I/50I	166
High quality video distribution (DVD, SDTV), MPEG2, 4-10 Mbps				
CCIR601	720x480/576	4:2:0	60I/50I	124
Intermediate quality video distribution (VCD, WWW), MPEG1, 1.5 Mbps				
SIF	352x240/288	4:2:0	30P/25P	30
Video conferencing over ISDN/Internet, H.261/H.263, 128-384 Kbps				
CIF	352x288	4:2:0	30P	37
Video telephony over wired/wireless modem, H.263, 20-64 Kbps				
QCIF	176x144	4:2:0	30P	9.1

جدول ۱: فرصت های ویدئو دیجیتال برای کاربردهای مختلف

عموماً برای داشتن کیفیت قابل قبول، یک ویدئو CIF، به ۳۸٤ kbps یا بیشتر و یک ویدئو QCIF به ٦٤kbps یا بیشتر نیاز دارد. بدنبال ایجاد استاندارد H.320، ITU-T، توسعهٔ کار را برای سایر رسانه های انتقالی ادامه داد. از مهمترین آنها، استاندارد H.323 است که برای شبکه هایی که کیفیت سرویس را تضمین نمی کنند، مانند اینترنت، تهیه شده است و نیز استاندارد H.324 که برای خطوط با پهنای باند بسیار کم مثل خطوط تلفن از طریق مودم ۲۸۰۸kbps یا کانال بی سیم، مطرح شده است. استاندارد کد کردن ویدئو در هر دوی این استانداردهای H.323 و H.324، H.263 است که پیشرفت محسوسی نسبت به H.261 بخصوص برای نرخ بیت های پایین دارد. با H.263، یک ویدئو QCIF، با کیفیت معادل یا بهتر از H.261 با نرخ ٦٤kbps، تا حدود ۲٤kbps فشرده می شود.

همزمان با تلاش ITU-T یک گروه تخصصی به نام MPEG، Motion Picture Expert Group، از ISO نیز استانداردهای مختلفی را برای ذخیره سازی، توزیع ارسال و انتشار ویدئو، توسعه دادند. اولین استاندارد این گروه MPEG-1 است که قابلیت فشرده سازی ویدئو و صوت با دقت SIF، (۳۰ fps ۲٤۰*۳٥۲ pels/sec) را تا ۱/٥Mbps با کیفیت خوب، دارد. این استاندارد امکان ذخیره‌سازی و پخش فیلم ها روی CD-ROM را می داد که در آن زمان نرخ آنها به ۱/٥ Mbps محدود می شد.

استقبال از فیلم های MPEG-1 روی CD (VCD یا Video CD) عاملی برای توسعهٔ ویدئوهای دیجیتال شد. وقتی استاندارد MPEG-1 برای اولین بار مطرح شد، سخت افزار پیچیده برای کد و واکد کردن بی درنگ آن مورد نیاز بود. عرضه شدن میکروپروسسور Intel Pentium I، واکد کردن فیلم های MPEG-1 بطور بی درنگ و گرفتن فیلم MPEG-1 از طریق WEB را میسر کرد.

بعد از کامل شدن MPEG-1، MPEG روی استاندارد کردن شیوهٔ فشرده سازی ویدئو و صدا، برای کاربردهای انتشار ویدئو با کیفیت بالا، متمرکز شد، با این هدف که ویدئو با دقت CCIR601 (کیفیت تلویزیون) بین ۳ تا ۱۰Mbp فشرده شود. این تلاش منجر به مطرح شدن MPEG–2 شد. ایجاد استاندارد MPEG-2 وقایع مهمی بدنبال داشت: امکان انتشار ویدئو از طریق ماهواره (مانند Direct-TV)، فیلم های DVD و تلویزیون دیجیتال. استاندارد MPEG-2 همچنین دارای امکاناتی برای فشرده سازی ویدئو با دقت HDTV است. همچنین می تواند با دقت MPEG-1 (SIF) نیز کار کند و با MPEG-1 سازگار است.

جدول ۲، استانداردهای فوق برای فشرده سازی سیگنال های چند رسانه ای را جمع بندی کرده است.

Standards	Application	Video Format	Raw Data Rate (Mbps)	Compressed Data Rate (Mbps)
H.320/H.261	Video conferencing/ telephony over ISDN	CIF	37	>=384 Kbps
		QCIF	9.1	>=64 Kbps
H.323/H.263	Video conferencing over Internet	4CIF/ CIF/ QCIF		>=64 Kbps
H.324/H.263	Video over phone lines/ wireless	QCIF	9.1	>=18 Kbps
MPEG-1	Video distribution on CD/ WWW	CIF	30	1.5
MPEG-2	Video distribution on DVD / digital TV	CCIR601 4:2:0	128	3-10
	HDTV	SMPTE296/295	<=700 Mbps	18-45

جدول ۲: استانداردهای کد کردن ویدئو برای کاربردهای مختلف

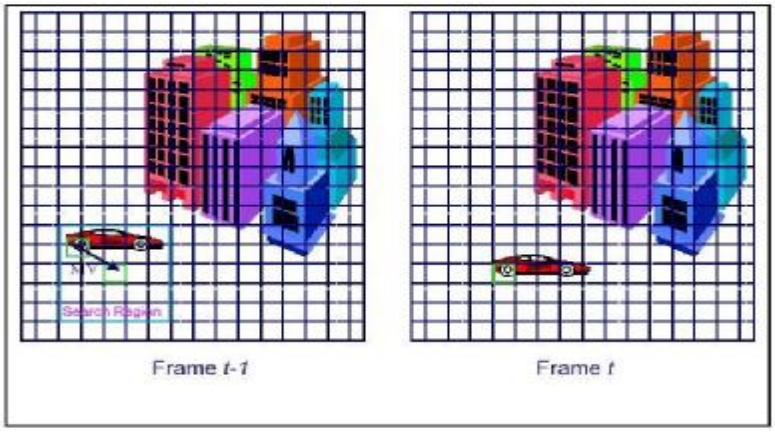
بدنبال MPEG-2، استاندارد دیگری به نام MPEG-4 طراحی شد که هدف آن ایجاد قابلیتی برای دسترسی به اشیاء مختلف و مجزا در تصویر ویدئو بود. ویدئو با روش مبتنی بر شیء کد می‌شود. مثلاً هر شیء بطور جداگانه کد می شود. تلاش گروه MPEG در ادامه برای استاندارد MPEG-7 است. که هدف آن ایجاد استانداردی برای توصیف و قابلیت مرتب سازی است، بگونه‌ای که امکان توصیف محتوای اطلاعات تصویری و صوتی را داشته باشد تا دستیابی به ویدئو دیجیتال را میسر کند.

در این آزمایش، تکنیک های ابتدایی برای فشرده سازی ویدئو، بخصوص تخمین جبران حرکت و مرور استانداردهای H.261، MPEG-1 و MPEG-2 ، دنبال می شود. برای جزئیات بیشتر در این زمینه به [1] مراجعه نمایید.

۲- تئوری ها و الگوهای فشرده سازی ویدئو

۲-۱- تخمین حرکت و جبران حرکت

تخمین و جبران حرکت، پایه و اساس اکثر الگوریتم های فشرده سازی ویدئو هستند. برای جبران حرکت، فرض می کنند که تصویر جاری، تصویری با تغییرات جزیی نسبت به تصویر قبلی است. این فرض، امکان استفاده از تخمین و درونیابی را ایجاد می کند. هنگامی که یک فریم بعنوان مرجع استفاده می شود، مجموعه فریم های دنبالهٔ آن، اختلاف های جزیی با یکدیگر دارند که نتیجهٔ حرکت اشیاء و یا حرکت دوربین است، که در شکل ۱ نشان داده شده است. برای اینکه مقایسهٔ فریم ها، ساده تر شود. یک فریم بطور کامل کد نمی شود، بلکه به بلوکهایی تقسیم می شود و بلوکها بطور مستقل کد می شوند. برای هر بلوک در فریمی که کد می شود (فریم جاری)، بهترین بلوک منطبق، در فریم مرجع در میان تعدادی از بلوکهای انتخاب شده، جستجو می شود. برای هر بلوک، یک بردار حرکت تولید می شود که اختلاف بین مکان آن بلوک و بهترین بلوک مطابق با آن، در فریم مرجع را نشان می دهد. این انتخاب می تواند با یک جستجوی کامل و جامع انجام شود، روشی که بطور مختصر شرح داده خواهد شد.



شکل۱: میزان حرکت مبتنی بر بلوک

در روش درونیابی، بردارهای حرکت، در ارتباط با دو فریم مرجع تولید می شوند، یکی از فریم مرجع قبلی و دیگری از فریم مرجع بعدی، بهترین بلوکهای تطبیق داده شده در هر دو فریم جستجو می شوند و دو بلوک بدست آمده میانگین گرفته می شوند.

۲-۱-۱- الگوریتم تطابق بلوکها برای تخمین حرکت (BMA)

فرض کنید که هر بلوک B_n در فریم f_k، متناظر با یک بلوک در فریم f_{k-1} با موقعیت $\overline{D}_n$ باشد.

که $\overline{D}_n$ بنام بردار حرکت B_n نامیده می شود. تخمین $\overline{D}_n$ می تواند از طریق حداقل کردن خطای تخمین، مانند جمع مجذور خطاها و یا قدر مطلق خطاها باشد.

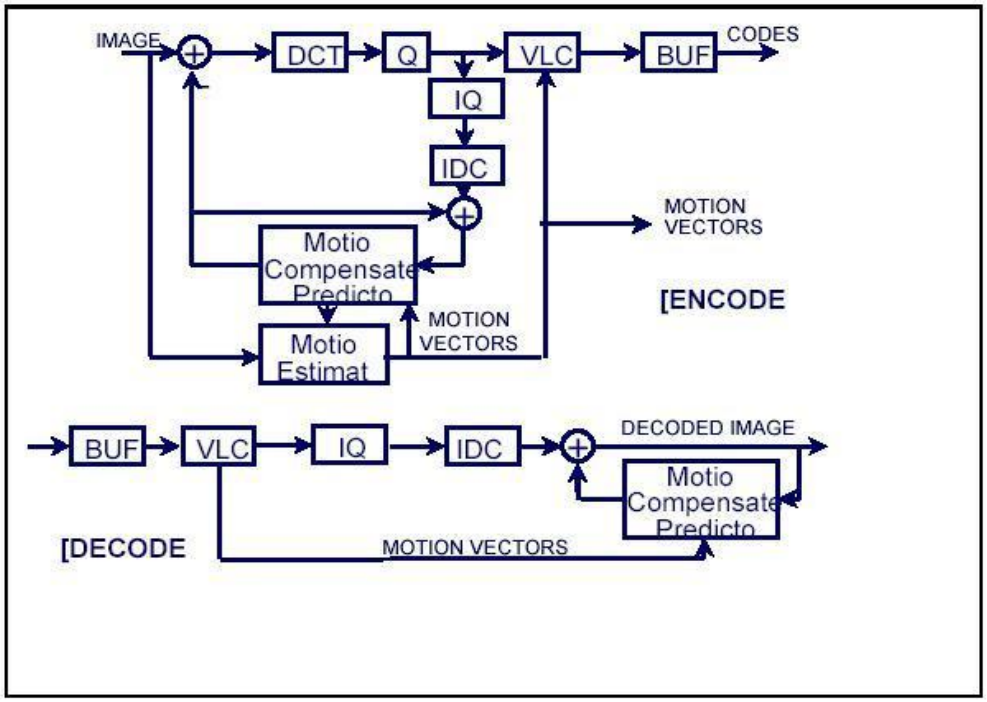
BMA یک روش جستجوی کامل را برای پیدا کردن بلوکی که دارای حداقل خطاست، استفاده می کند. بعبارت دقیق تر، ایـن روش بلوک جاری را با کلیهٔ بلوک های ممکن در یک محدودهٔ از قبل تعریف شده در اطراف مکان فعلی، مقایسه می کند، برای هـر بلـوک تعیین شده، خطای تخمین را محاسبه می کند بعد از بررسی تمام بلوکهای ممکن، بلوکی که حداقل خطا را داشت بعنوان بهترین بلوک مطابق، انتخاب می شود.

حرکت واقعی بین دو فریم ویدئو، عموماً بر اساس بلوکهای ثابت قابل تعیین نیست. بنابراین استفاده از الگوریتم تطابق بلوکهـا بـرای تخمین حرکت و جبران حرکت نمی تواند تخمین های دقیقی ایجاد کند. هنگامی که حرکت حقیقی یک بلوک، یک انتقال ساده نباشد، الگوریتم برای پیدا کردن بلوکی با حداقل خطا، تلاش می کند. شکل ۲ مثالی را از جبران حرکت با استفاده از روش تطابق بلوک هـا، نشان می دهد.

شکل۲: مثالی از جبران حرکت با استفاده از روش تطابق بلوک ها

تصویر بالا سمت چپ، تصویر فریم قبلی است، تصویر بالا سمت راست، فریم جاری است، تصویر پایین سمت راست، میدان حرکتهای تخمین زده شده است (بردارهای حرکت تعیین شده از محل مرکز هر بلوک رسم شده اند.) و تصویر پایین سمت چپ، تصویر تخمین زده شده با استفاده از میدان حرکت را نشان می دهد.

ملاحظه می شود که الگوریتم بطور دقیقی عدم حرکت در پس زمینه را تعیین کرده است در حالیکه بسته بودن چشم با این روش ساده، در شکل تخمین زده شده دیده نمی شود.



شکل ۳: مراحل کد و واکد کردن برای یک ماکروبلاک در یک کد کنندهٔ ویدئو معمولی

۲-۱-۲- کد کردن ویدئو با استفاده از جبران حرکت و کد تبدیل

متداولترین روش کد کردن ویدئو به نام کد کردن block-based hybrid شناخته می شود. در این روش هر فریم به ماکرو بلوک ها (MBs) تقسیم می شود. هر کدام دارای چندین بلوک 8*8 است. هر ماکرو بلوک با استفاده از ترکیبی از جبران حرکت و تبدیل DCT، همانطور که در شکل ۳ نشان داده شده است، کد می شود. هر ماکرو بلوک می تواند در یکی از دو مود، کد شود. در مود Intra یک روش مبتنی بر کد کردن DCT و شبیه به JPEG بطور مستقیم روی هر بلوک اعمال می‌شود. این مود برای هر ماکروبلاک در اولین فریم بکار می رود و بعد بطور متناوب در فریم های زیر دنباله نیز استفاده می شود. در مود Inter، یک بردار حرکت در ابتدا تعیین می شود و روش DCT برای کد کردن خطای میزان حرکت استفاده می شود. خصوصاً، این روش، ماکروبلاک جاری از فریم جاری را با بهترین ماکروبلاک مطابق با آن در فریم قبلی، تخمین می زند (فقط مؤلفه های Luminance) اگر خطای تخمین زدن کمتر از حد تعیین شده از قبل باشد. اختلاف داده ها تعیین نمی شود. در غیر اینصورت خطای تخمین با استفاده از DCT تبدیل می شود و مؤلفه های تبدیل یافته، کوانتیزه شده و با استفاده از روش runlength مشابه با JPEG، کد می شوند. در نهایت، رشته بیت کد شده همراه با اطلاعات بردارهای حرکت کد شده به video multiplex ارسال می‌شوند. همانطور که در توصیف JPEG شرح داده شد، اندازهٔ step کوانتیز کننده، می تواند بر اساس کیفیت تصویر دلخواه و کارایی کد کردن، تغییر کند.

۲-۲- الگوریتم کد کردن ویدئو H.261

در سال ۱۹۹۰، CCTTT مجموعه ای از استانداردهای بین المللی ویدئو کنفرانس، شامل کد کننده ویدئویی H.261 برای سرویس های صوتی تصویری روی ISDN، p*64 kbps (که با استاندارد p*64 نیز شناخته می شوند) تصویب کرد. کاربردهای در نظر گرفته شده برای این استاندارد، تلفن تصویری و سیستم ویدئو کنفرانس است. بنابراین سیستم هایی که با این استاندارد مرتبط هستند

باید قابلیت کد و واکد کردن بی درنگ این استاندارد را داشته باشند. محدودهٔ p از ۱ تا ۳۰ است. برای یک ارتباط ISDN با نرخ پایه، p یک تا ۲ است.

CCTTT، فرمت های CIF و QCIF را بعنوان فرمت های ویدئویی برای تلفن تصویری در نظر گرفت. ویدئو با فرمت CIF دارای ۳۵۲*۲۸۸ پیکسل برای Y و ۱۷۶*۱٤٤ پیکسل برای $Cr\&C_b$ است. نرخ فریم می تواند بین یک تا ۳۰ فریم در ثانیه باشد. کلیهٔ کد کننده ها باید در سطح QCIF عمل کنند و عملکرد در سطح CIF اختیاری است. با سرعت تقریباً ۳۰ فریم در ثانیه (۲۹/۹۷ حداکثر مقداری است که پشتیبانی می شود.) CIF فشرده نشده نرخ ۳٦/٤٥ Mb/s و QCIF نرخ ۹/۱۱۵Mb/s دارد. برای یک کانال ۱/۵ Mb/s، کاهش قابل توجهی لازم است ولی برای یک کانال p=2 (۱۲۸ kb/s) کاهش ۱: ۲٤، احتیاج است. عموماً، CIF برای ارسال هایی با نرخ بیشتر از ۳۸٤ kb/s (p=٦) توصیه می شود.

کد کنندهٔ H.261 از یک ترکیبی از الگوهای DCT و DPCM با تخمین حرکت، استفاده می کند همانطور که در شکل ۳ نشان داده شده است. هر ماکرو بلاک می تواند به صورت یکی از دو مود Intra یا Inter کد شود. مود Intra برای هر ماکروبلاک در اولین فریم استفاده می شود و بعد متناوباً در زیر دنبالهٔ فریم ها استفاده می شود تا از انتشار خطا که در اثر خطاهای ارسال پیش می‌آیند، جلوگیری کند.

H.261 از یک ساختار داده ای سلسله مراتبی برای کد کردن داده ها استفاده می کند. این ساختار شامل تصویر، گروه بلوکها (GOB)، ماکروبلاک و بلوک می باشد. یک بلوک مجموعهٔ ۸*۸ از پیکسل هاست که می تواند شامل نمونه های y، C_b یا C_r باشد. یک ماکرو بلاک از ٤ تا بلوک Luminance با ابعاد ۸*۸ (y) و دو بلوک Chrominance (C_r & C_b) تشکیل می شود. یک GOB شامل ماکروبلاک ها در چندین سطر متوالی می شود. یک تصویر شامل چندین GOB می‌شود. استاندارد H.261، syntax رشته بیت کد شده را تعریف می کند. هر بلوک شامل مؤلفه‌های DCT(DCTCOEFF) از یک بلوک و یک علامت EOB بدنبال آن می باشد. هر ماکروبلاک شامل داده‌های ٦ بلوک و یک سرآیند ماکروبلاک است. یک GOB از کی سرآیند GOB و ماکروبلاک‌های آن GOB، بدنبال آن ساخته می شود. در نهایت، تصویر شامل یک سرآیند تصویر است که آرایه ای متوالی از GOB ها بدنبال آن می آیند.

۲-۳- استاندارد MPEG-1

استاندارد MPEG-1 به منظور ذخیره کردن یک ویدئو متحرک با دقت SIF (۱۷۶*۱۲۰ pels for C_r & C_b at 30 fps و pels ۳۵۲*۲٤۰ for y) با نرخ ۱/۵Mbps، طراحی شده است.

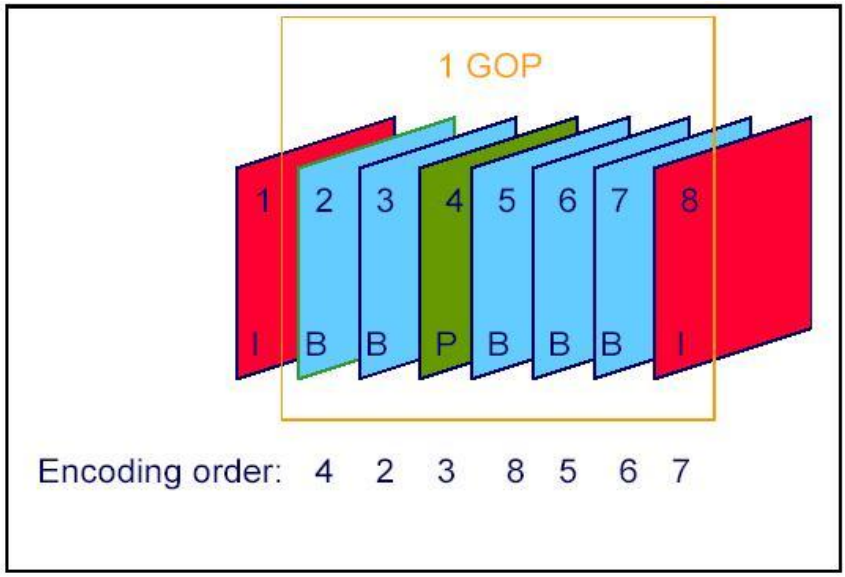
در ادامه نحوهٔ انجام این موارد توسط MPEG-1، شرح داده می شود.

۲-۳-۱- مودهای کد کردن تصویر و جبران حرکت دو جهته

یکی از تفاوت های اساسی بین MPEG–1 و H.261 این است که علاوه بر استفاده از جبران حرکت از فریم قبلی، از فریم بعدی نیز استفاده می کند (که قبلاً کد شده است). بطور کلی فریم جاری می تواند از روی هر دو فریم قبلی و بعدی، تخمین زده شود. این عمل بعنوان تخمین دو بسته در MPEG نامیده می شود. MPEG-1 فریم های یک ویدئو را در سه مود مختلف کد می‌کند، تصویر I، تصویر p و تصویر B. فریم ها به GOP ها تقسیم می شوند بطوریکه هر GOP شامل یک تصویر I و چندین تصویر P و B می باشد. این تقسیم بندی در شکل ٤ نشان داده شده است. عملیاتی که برای تصاویر مختلف انجام می شود با جزئیات بیشتر در ادامه توضیح داده می‌شوند.

Intra picture (I) : یک تصویر I بر اساس خود تصویر (با استفاده از یک روش مشابه با JPEG)، فشرده می شود. همانند الگوریتم JPEG هر بلوک ۸*۸ در یک ماکرو بلوک ۱٦*۱٦، تحت تبدیل DCT قرار می گیرد و مؤلفه های DCT تولید می شوند. این مؤلفه ها کوانتیزه شده و به صورت zig-zag مرتب می شوند تا بهترین runlength از مؤلفه های صفر بدست آید.

runlength و مقادیر غیر صفر با استفاده از روش کد کردن همینگ، کد می شوند. مود تصویر I، برای اولین فریم در هر GOP، برای ایجاد امکان دستیابی تصادفی، استفاده می شود.

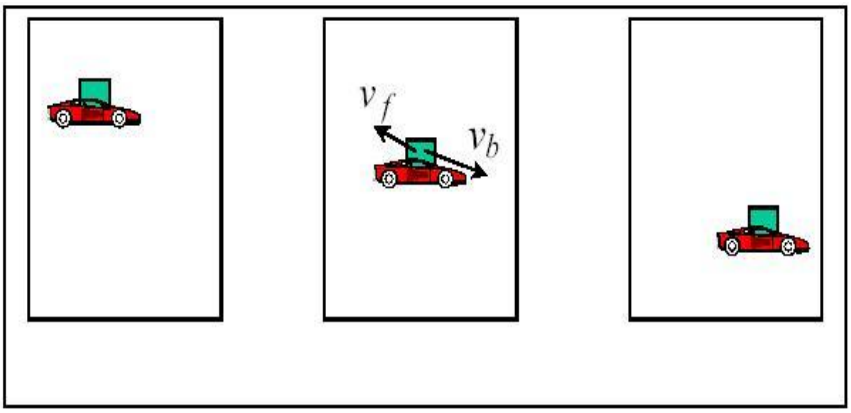


شکل۴: ساختار GOP در MPEG-1

(P) Unidirectional Predicted Pictures : این مود از تکنیک جبران حرکت (شبیه به روش استاندارد H.261) برای فشرده سازی، استفاده می کند. هر ماکروبلاک از تصویر P یا I قبلی برای تخمین زدن یک بردار حرکت استفاده می کند. خطای تخمین زدن با استفاده از DCT، تبدیل شده و مؤلفه های DCT، کوانتیزه و runlength می شوند.

(B) Bidirectionally predicted picture : این مود یکی از سه نوع روش جبران حرکت را برای هر ماکرو بلاک بکار می برد.

جبران حرکت پیش رونده، جبران حرکت پس رونده و جبران Interpolative، جبران حرکت پیش رونده از تصاویر I و P قبلی (مثل روش تصاویر P) استفاده می کند. جبران حرکت پس رونده از اطلاعات تصویر بعدی استفاده می کند که ماکرو بلوک جاری با بهترین بلوک منطبق شده در تصاویر I یا P بعدی، تخمین زده می شود. جبران Interpolative، از متوسط گیری بین بهترین بلوکهای منطبق شده در تصاویر قبلی و بعدی، استفاده می کند. روش جبران حرکت دو جهته در شکل ۵ نشان داده شده است.



شکل ۵: تخمین و جبران حرکت دو جهته

از آنجا که کلیهٔ فریم ها در یک GOP می توانند بدون اطلاع از GOP قبلی، واکد شوند، یک GOP واحد پایه ای است برای دستیابی تصادفی. یک fast forward می تواند با واکد کردن تنها تصاویر I و یا تصاویر I و P انجام شود. یک fest rewind نیز می تواند با واکد کرد تنها تصاویر I انجام شود.

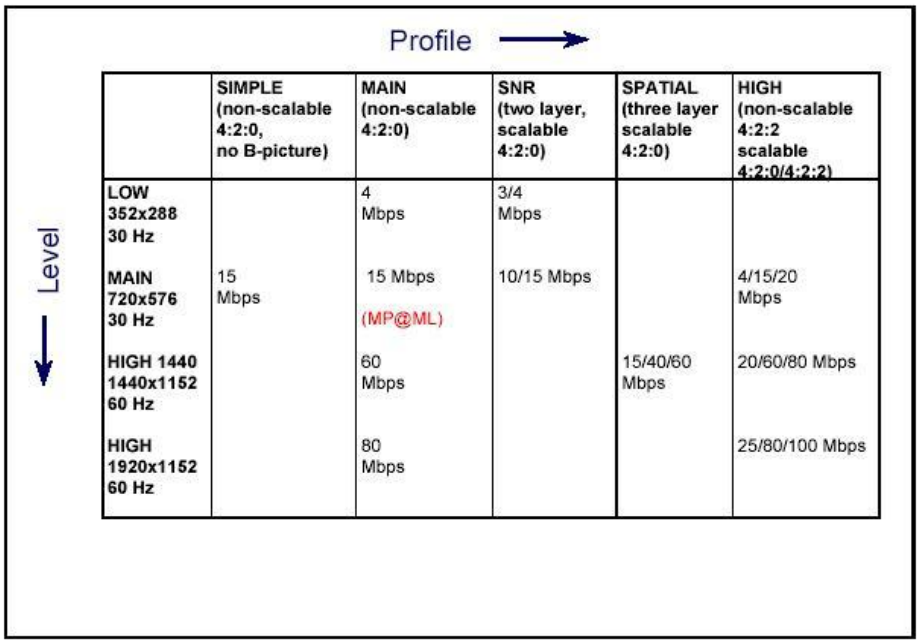
۲-۳-۲- تخمین حرکت با دقت Half-Pel

اختلاف دیگر بین MPEG-1 و H.261 این است که بردارهای حرکت در MPEG-1 با دقت half-pel تخمین زده می شود. یعنی یک جستجوی کامل، با افزایش نیم pel انجام می شود نه با افزایش به اندازهٔ عدد صحیح. تخمین حرکت به این صورت دقیق تر است و امکان کاهش خطای تخمین را می دهد. ولی باز هم به درونیابی فریم قبلی برای تولید نمونه ها در مکان half-pel احتیاج است که این خود به محاسبات بیشتر نسبت به حالت تخمین حرکت با دقت integer-pel، نیاز دارد.

۲-۴- استاندارد کد کردن ویدئو MPEG-2

همانطور که در مقدمه اشاره شد، استاندارد MPEG-2 برای کد کردن ویدئو CCTR601 مطرح شده است. روش اولیهٔ کد کردن در MPEG-2، مشابه روش MPEG-1 است، با ساختار GOP مشابه که هر ماکروبلاک با استفاده از تبدیل مستقیم DCT (مود I)، با تخمین یک طرفه (مود P) و یا با تخمین دو طرفه (مود B) کد می شود. گذشته از دقت spatial بیشتر، اختلاف اساسی بین ویدئو CCTR601 و CIF/SIF، استفاده از interlacing در ویدئو CCTR601 است. این سبب می‌شود که پردازش فشرده سازی، بطور قابل توجهی پیچیده شود. بنابراین روش های خاصی برای کنترل کردن تصاویر interlaced، مطرح شده است که عملکرد جبران و تخمین حرکت و DCT را تغییر می دهد. جزئیات بیشتر در [۱] یافت می شود.

MPEG-2 می تواند فرمت های ویدئو با سطوح رزولوشن مختلف را هندل کند. این استاندارد دارای profile های مختلفی است که قابلیت های بیشتری را در اختیار می گذارد. بحث فوق تنها به profile اصلی در سطح اصلی (mp@ml) اشاره دارد. یک سیگنال HDTV با استفاده از profile اصلی در سطح بالا کد می شود. یک توسعهٔ مهم دیگر MPEG-2 در مقایسه با MPEG-1، scalability profile است که یک ویدئو را قادر می سازد که بصورت یک لایهٔ اصلی و یک لایهٔ تکمیلی، کد شود. لایهٔ اصلی، کیفیت پایه را ایجاد می کند و لایهٔ تکمیلی، هنگامیکه به لایهٔ اصلی افزوده می شود، می تواند کیفیت را بهبود دهد. یک ویدئو کد شده با MPEG-2، با استفاده از مود scalability می تواند روی شبکه هایی با پهنای باندهای مختلف برای گیرنده هایی با قابلیت دریافت با دقت های spatial مختلف، منتقل شود. شکل ٦، profile های مختلف و سطح مختلفی که توسط MPEG2 پشتیبانی می شوند را خلاصه می کند.



	SIMPLE (non-scalable 4:2:0, no B-picture)	MAIN (non-scalable 4:2:0)	SNR (two layer, scalable 4:2:0)	SPATIAL (three layer scalable 4:2:0)	HIGH (non-scalable 4:2:2 scalable 4:2:0/4:2:2)
LOW 352x288 30 Hz		4 Mbps	3/4 Mbps		
MAIN 720x576 30 Hz	15 Mbps	15 Mbps (MP@ML)	10/15 Mbps		4/15/20 Mbps
HIGH 1440 1440x1152 60 Hz		60 Mbps		15/40/60 Mbps	20/60/80 Mbps
HIGH 1920x1152 60 Hz		80 Mbps			25/80/100 Mbps

شکل ۶: سطوح و profile های مورد حمایت MPEG-2

۲-۵- سایر استانداردهای تجاری فشرده سازی ویدئو

۲-۵-۱- تکنولوژی Intel's Indeo

ویدئو Indeo یک تکنولوژی نرم افزاری است که توسط Intel Architecture Labs تولید شده و اندازهٔ فایل های ویدئو دیجیتال فشرده نشده را از ۵ تا ۱۰ برابر کاهش می دهد. این تکنولوژی در محصولاتی مانند Microsoft's video for windows و Apple's Quicktime گنجانده شده است.

تکنولوژی Indeo، از چندین نوع تکنیک فشرده سازی "Lossy" و "Loos less" استفاده می‌کند. تکنولوژی Indeo، ویدئو را بطور همزمان با دریافت آن از طریق video capture board، فشرده می کند بنابراین دادهٔ فشرده نشده نیازی به ذخیره شدن روی دیسک را ندارد. ویدئو آنالوگ دریافتی از یک دوربین ویدئو، VCR، یا دیک لیزری، با هر نوع فرمت استانداردی مانند NTSC، از طریق video capture board مانند یک Intel smart video Recorder board، به فرمت دیجیتال تبدیل می شود.

روش Indeo شامل مراحل زیر است (تمامی آنها الزامی نیستند):

۱- نمونه برداری yuv، برای کاهش مساحت پیکسل به یک مقدار رنگ متوسط.

۲- اختلاف پیکسل و فشرده سازی زمانی، برای کم کردن داده از طریق ذخیره کردن تنها اطلاعاتی که بین پیکسل ها یا فریم ها تغییر کرده است. (این با کوانتیزه کردن اختلاف فریم از روش کوانتیزاسیون برداری انجام می شود.)

۳- کد کردن run-length برای فشرده کردن شاخص های کلمهٔ کد.

٤- کد کردن variable – content، برای کاهش یک مجموعهٔ متفاوت از اطلاعات به تعداد ثابتی بیت.

فایل ویدئو دیجیتال شده با اطلاعات صدا، طبق یک فرمت استاندارد، مثل Microsoft's AVI یا Apple's Quicktime ترکیب می شود و روی دیسک سخت ذخیره می شود. فایل ترکیب شده می تواند برای پخش شدن یا تصحیح شدن، توزیع شود. برای پخش

کردن، فایل باید به قسمت‌های ویدئو و صدا تجزیه شود و ویدئو از طریق یک تعداد روش (عکس عملیات فشرده‌سازی)، واکد شود تا نمایش پیکسل های دیجیتال واقعی مربوط به ویدئو دیجیتال فشرده شده، میسر شود.

سه عامل مؤثر در کارایی عبارتند از:

۱) سرعت میکروپروسسور، ۲) اندازهٔ پنجرهٔ playback بر حسب پیکسل و ۳) نرخ فریم

کوچکتر بودن پنجرهٔ playback باعث می شود که تصاویر ویدئو، طبیعی تر باشند. میکروپروسسور پنجره های playback بزرگتر و نرخ های فریم بیشتر را حمایت کند.

تکنولوژی Indeo، scalable است، یعنی نرخ های سریعتر فریم برای مشتریان با قدرت پردازش بیشتر را فراهم می کند.

Apple's Quicktime -۲-۵-۲

محصول Quicktime با هزینهٔ کم، برای سیستم های end-user desktop، ویدئو تمام متحرک را به ارمغان آورد.Apple ، عمل فشرده سازی و عکس آن را به طور نرم افزاری پیاده کرد. کوانتیزه کردن برداری یکی از روش های فشرده سازی نرم افزاری است که در Quicktime موجود است. این روش امکان داشتن ویدئو با دقت، ۲٤۰*۳۲۰ تا ۳۰ frame/sec بدون کمک سخت افزار را فراهم می‌کند. نسبت های فشرده سازی که با این روش بدست می آیند بین ۲۵ تا ۲۰۰ است.

Microsoft AVI -۳-۵-۲

همانند Quicktime، هدف از Microsoft AVI، ایجاد ویدئو با دقت کم و هزینهٔ کم روی desktop است. بر خلاف Quicktime، که قسمتی از سیستم عامل است، AVI بعنوان یک ماژون سطح مجزا، تعریف شده است. AVI بعنوان یک راه حل تنها نرم افزاری، طراحی شده است تا روی مانیتورهای VGA و Super VGA، ویدئو را نمایش دهد. دقت AVI، هنگامیکه با دقت VCR در محدودهٔ ۳۲۰ خط یا بیشتر مقایسه می شود، عموماً کمتر از یک سیگنال تلویزیون عادی است. یک خصوصیت مهم Scalability, AVI آن است.

کارایی تحت AVI، بستگی به سخت افزار مورد استفاده در لایهٔ زیرین آن دارد. AVI شامل چندین الگوریتم نرم افزاری فشرده سازی و عکس آن است. برخی از این الگوریتم ها برای حرکت بهینه شده اند در حالیکه برخی دیگر برای ویدئوهای ثابت بهینه شده اند.

AVI چندین dialog box برای انتخاب اندازهٔ پنجره ها، نرخ فریم، کیفیت و الگوریتم فشرده سازی در اختیار قرار می دهد. کیفیتی که با AVI به دست می آید، با کیفیت Quick Time قابل مقایسه است. با وجودی که بر اساس تکنولوژی های متفاوتی عمل می کنند، در ظاهر خیلی شبیه یکدیگر هستند.

Intel's DVI -۴-۵-۲

Intel's Digital Video Interface یک استاندارد سخت افزاری است. از آنجا که استانداردهای جدیدتری بطور نرم افزاری پیاده سازی شده اند (و DVI اهمیت خود را به عنوان استاندارد تا حدودی از دست داده است)، جزئیات این روش در اینجا مطرح نشده است.

افرادی که به این استاندارد علاقمند هستند می توانند یک نسخه از استاندارد را تهیه و جزئیات آن را مطالعه نمایند.

۳- آزمایش

۱- با فرض داشتن دو فریم از یک دنبالهٔ ویدئو، برنامه ای بنویسید که بردار حرکت را برای اولین بلوک ۱٦*۱٦ (برای مثال بلوک (۰و۰) و (۱۵و۰) و (۰و۱۵) و (۱۵و۱۵) در دومین فریم پیدا کند. دو فریم را بخوانید از توابع () fopen و ()fread استفاده کنید، پارامترهای B,A، BK-size، BK-location، S-start، S-end را بعنوان ورودی تابع ()EBLK که در پیوست A آمده است. در نظر بگیرید. EBLK تابعی است که می تواند بردار حرکت را برای بلوک مشخصی پیدا کند. می‌توانید محدودهٔ جستجو را بطور ثابت ۱٦- و ۱٦ در نظر بگیرید.

۲- برنامه ای را که در قسمت ۱ نوشته اید تکمیل کنید طوریکه بردارهای حرکت را برای تمام بلوک ها در فریم یک ۱ به ۲ (بردار ۲ بعدی برای ذخیره سازی کلیهٔ بردارهای حرکت نیاز است یکی برای mv-x و دیگیری برای (mv-y)، بدست آورد. میدان حرکت را با استفاده از تابع() quiver رسم کنید.

۳- برنامه ایی که در قسمت ۲ نوشته‌اید را تکمیل کنید طوریکه تصویر تخمین زده شده از فریم دوم را با کپی کردن بلوک مربوط از فریم اول (تعیین شده توسط بردار حرکت)، در فریم دوم، بدست آورید. دو فریم اصلی، فریم دوم تخمین زده شده، تصویر خطا (قدر مطلق خطای پیکسل) بین دو فریم اصلی، تصویر خطا بین فریم دوم تخمین زده شده و فریم دوم اصلی را نمایش دهید. میدان حرکت را با کمک دستور quiver در مطلب نمایش دهید. همچنین مقدار PSNR از تصاویر خطای اصلی و جدید را محاسبه کنید.
PSNR بصورت زیر تعریف می شود:

$$PSNR = 10\log_{10}\frac{255^2}{\sigma_e^2}(\text{dB}), \quad \text{where } \sigma_e^2 = \sum_{m,n} e^2(m,n)/(m*n),$$

که در آن e(m,n) مقدار خطا در پیکسل (m,n) است.

کلیهٔ مشاهدات خود را در مورد تصویر تخمین زده شده، میدان حرکت، اختلاف بین تصاویر خطا و مقادیر PSNR بنویسید. چه نوع artifact در تصویر تخمین زده شده مشاهده می کنید؟ علت آن چیست؟

۴- بخش ۱ تا ۳ را با ۳ اندازه بلوک دیگر و پنجره‌های جستجوی متناسب با آن‌ها تکرار کنید. تغییر اندازه بلوک را تحلیل کنید.

۵- برنامهٔ encode.m در پیوست B را بخوانید. این برنامه توابع شما در قسمت ۳ را فرا می‌خواند که بنام getprediction() نامیده شده است. این برنامه تبدیل DCT را روی تصویر خطا انجام می دهد. تبدیل DCT ۸*۸ را روی هر بلوک در تصویر خطای تخمی زده شده، اعمال کنید. چند مؤلفهٔ اول DCT در هر بلوک را نگه دارید و بعد تصویر خطای تخمین را که بدست می آید به فرم تخمین زده شده اضافه کنید تا تصویر بازسازی شده را بدست آورید. PSNR بین تصویر بازسازی شده و تصویر اصلی محاسبه می شود تا کیفیت تصویر کد شده بررسی شود. لطفاً در مورد هر خط دستوری، توضیح دهید که نشان دهد وظیفهٔ آن خط دستور چیست؟ (برای هر دو قسمت comment دقیق بگذارید)

٦- حداقل تعداد مؤلفه هایی که لازم است تا نگهداری شوند تا نتیجهٔ رضایت بخشی از نظر تشخیص چشم بدست آید (می توان از محاسبهٔ PSNR نیز استفاده کرد که PSNR بیشتر از 30 dB لازم است.) را پیدا کنید. همچنین این روش کد کردن DCT را بطور مستقیم برای فریم اصلی دوم بکار برید، DCT ۸*۸ را روی هر یک از بلوکهای فریم اصلی دوم اعمال کنید و چند مؤلفهٔ اول DCT را نگه دارید. تعداد حداقل مؤلفه های مورد نیاز را که نتیجهٔ رضایت بخش می دهند، پیدا کنید و حداقل مؤلفه های لازم در این دو حالت را مقایسه کنید. فرض کنید که کد کردن تعداد یکسانی از مؤلفه های DCT در هر بلوک، تعداد بیت های یکسانی را مصرف خواهد کرد، کدام روش کیفیت بهتری را با نرخ بیت یکسان خواهد داشت؟

۴-گزارش (موعد تحویل ۱۰ روز از زمان بارگزاری)

۱- کدهای matlab و کلیهٔ نتایج تصاویر میانی و نهایی خود را تحویل دهید. ارائه مشاهدات و توضیح آن ضروری است.

۲- در یک سیستم کد کردن ویدئو معمولی، ابتدا تخمین جبران حرکت انجام می شود و بعد تصویر خطای تخمین زدن، با استفاده از DCT، کد می شود. این بر اساس این فرض است که استفاده از کدDCT روی تصویر خطا، نسبت به اعمال DCT روی تصویر اصلی بطور مستقیم، احتیاج به نرخ بیت کمتری دارد.

آیا این فرض صحیح است؟ آیا می توانید با استفاده از یک برنامهٔ matlab، محاسباتی انجام دهید که صحت پاسختان را مشخص کند.

(نتایج بدست آمده در قسمت های ۵ و ٦ در قسمت آزمایش را مقایسه کنید.)

۳- تحلیل مناسبی برای تاثیر اندازه پنجره‌ها، اندازه بلوک‌ها و تعداد ضرایب DCT حذف شده در کیفیت و سرعت پردازش بیاورید.

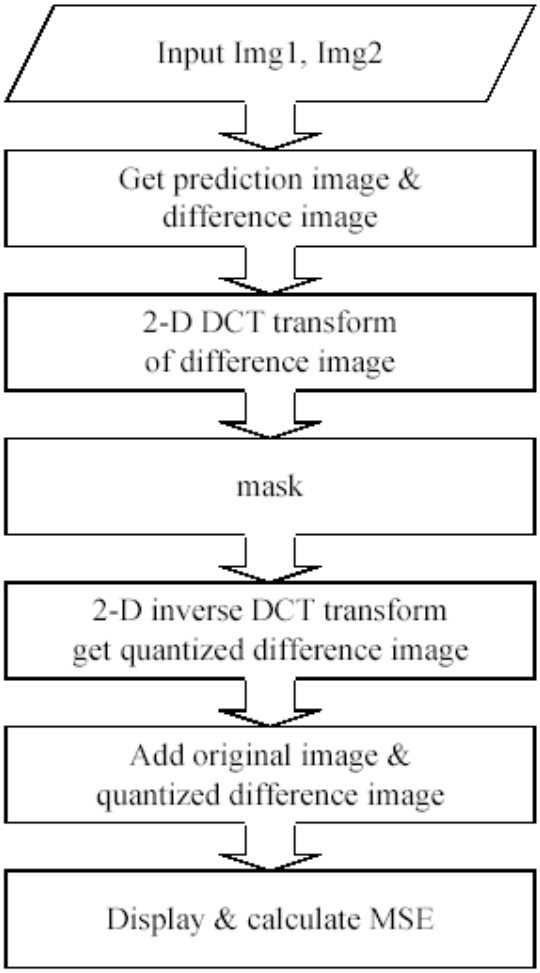
۵- مراجع

[1] A. N. Netravali and B. G. Haskell, " Digital Pictures -- Representation, Compression, and Standards ", 2nd ed., Plenum Press, 1995.

[2] Y. Wang, J. Ostermann and Y.-Q. Zhang, "Video Processing and Communications", Prentice Hall, 2002.

۶- ضمائم:

ضمیمه الف:



بلوک دیاگرام تابع encode.m

```
function [mv_x, mv_y]=EBLK(A,B,Bk_size,Bk_location,S_start,S_end)

%EBLK compute Motion Vector just for one block in Frame B from Frame A
%The size of block in Frame B is Bk_size(1) by Bk_size(2)
%The location of the first pel(upper-left corner) of the block in Frame B
%is (Bk_location(1),Bk_location(2))
%The Search Field in Frame A is from (S_start(1),S_start(2)) to
(S_end(1),S_end(2))
%The return MV is in mv_x and mv_y

%Bk_size, Bk_location, S_start and S_end are 1-by-2 vector. The first element
%represents x direction component. The second represents y direction component.

%Exapmle: To compute the MV for one block in Frame B. The Block size is 16 by
16.
%The block location is (100,200). Search Range in Frame A is from(-16 -16) to
(16 16).
%You need to figure out the values for S_start and S_end.
%Therefore, Bk_size(1)=16, Bk_size(2)=16
%       Bk_location(1)=100, Bklocation(2)=200
%       S_start(1)=84, S_start(2)=184
%       S_end(1)=116, S_end(2)=216
%       [mv_x, mv_y]=EBLK(A,B,Bk_size,Bk_location,S_start,S_end)

tx=Bk_location(1)
ty=Bk_location(2)
Nx=Bk_size(1);
Ny=Bk_size(2);
Block_B=B(tx:tx+Nx-1,ty:ty+Ny-1);

%Initial mv_x & mv_y
mv_x=0;
mv_y=0;

%Initial error
error=255*Nx*Ny;
error=255*Nx*Ny;

for sx=S_start(1):S_end(1)
  for sy=S_start(2):S_end(2)
    temp_error=sum(sum(abs(Block_B-A(sx:sx+Nx-1,sy:sy+Ny-1))));
    if temp_error < error
      error=temp_error;
      mv_x=tx-sx;
      mv_y=ty-sy;
    end;
  end;
end;
```

ضمیمه ب:

Appendix B

```
% function []=encode(file,n)
% file: read from saved workspace
% get anchor, track and error image from the saved workspace
% Img1: anchor frame
% Img2: track frame
% Img3: predicted frame
% error: Img3-Img2,prediction error
% n: number of coeffs to be kept in the error image
% Example: encode(10);

function []=encode(Img1,Img2,n)

% get prediction error and predicted image
[Img3,ImgErr]=getprediction(Img1,Img2);

% apply DCT to error frame
y=blkproc(ImgErr,[8,8],'dct2');
% only keep n coefficients
yy=blkproc(y,[8,8],'mask',n);
% get idct
yq=blkproc(yy,[8,8],'idct2');
% restore
yq=yq+Img3;

% plot the original frame
subplot(2,1,1);
colormap(gray(256));
image(Img2');
title('original track frame');
set(gca,'XTick',[],'YTick',[]);

% reconstruct the frame
subplot(2,1,2);
colormap(gray(256));
image(yq');
set(gca,'XTick',[],'YTick',[]);
% calculate PSNR
error=Img2-yq;
PSNR=10*log10(255^2*352*240/sum(sum(error.^2)));
temp=sprintf('quantized:PSNR=%5.2fdB,n=%d',PSNR,n);
title(temp);
truesize;
```