باسمه تعالی

جزوه درس مالتی مدیا

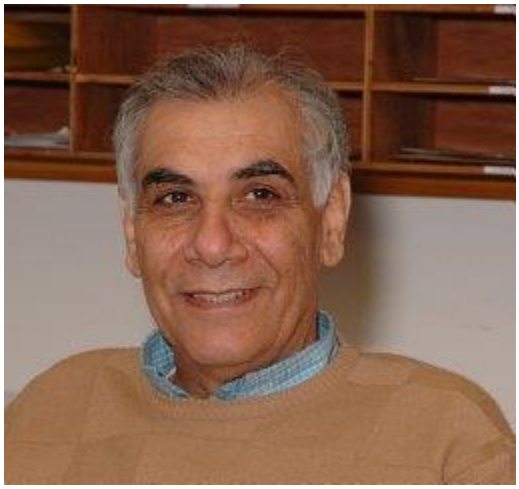
استاد: دکتر محمد قنبری

تهیه کننده: رضا سعیدی نیا

مرجع:

Standard codecs, Image compression to advanced video codecs, 3d edition, Mohammad Ghanbari, Published by The Institution of Engineering and Technology, London, United Kingdom

تقدیم به روح استاد عزیزم پروفسور محمد قنبری



روحش شاد و یادش گرامی

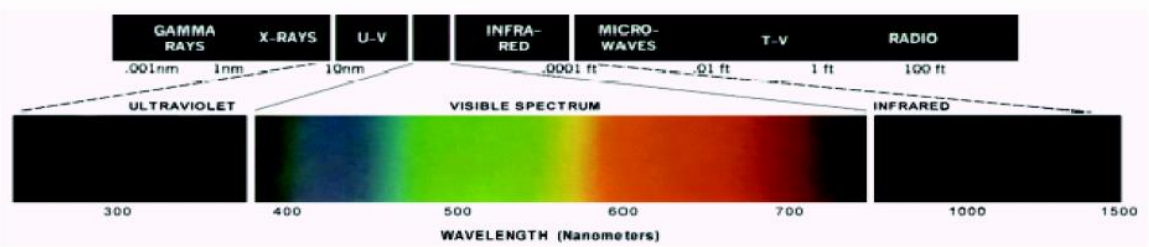
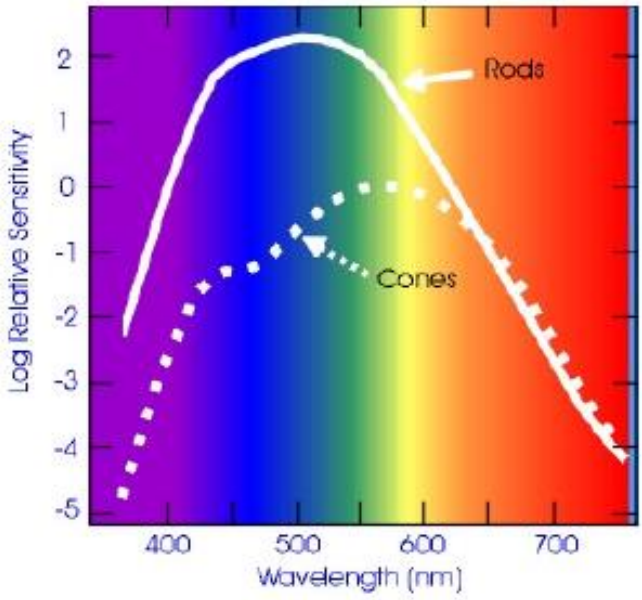
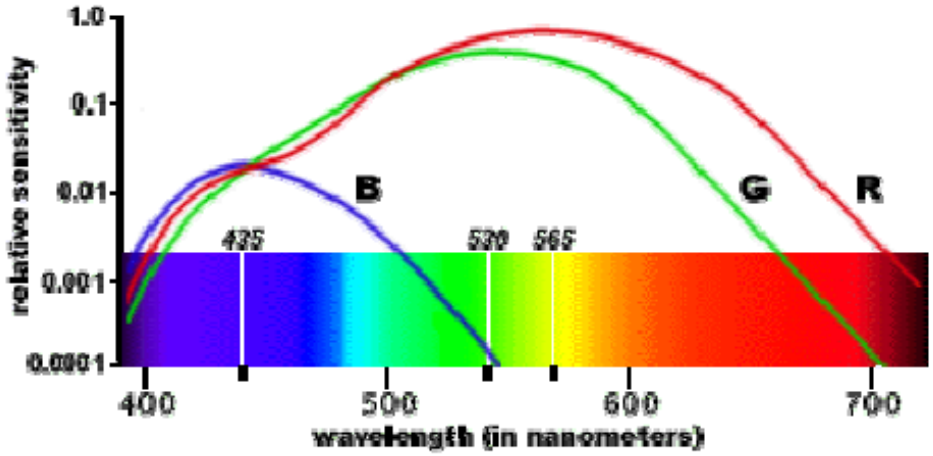


FIGURE 6.2 Wavelengths comprising the visible range of the electromagnetic spectrum. (Courtesy of the General Electric Co., Lamp Business Division.)

این منحنی نور هست و اسپکتروم نوری که ما میبینیم را مشخص می کند. طبق نمودار محدوده زیادی از اسپکتروم را نمی‌بینیم. مساله اصلی در ویدیو چشم هست. و چیزهایی که ما می‌بینیم و چیزهایی که مغز نمی فهمد. این مشخصات چشم در ارسال ویدیو بسیار مهم هست.



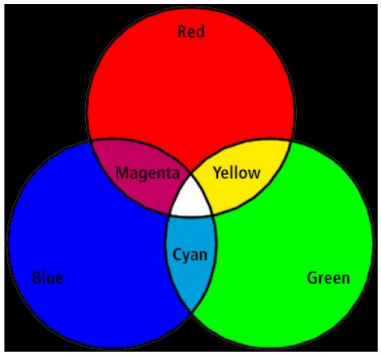
می‌توان خیلی از رنگ ها را با ترکیب سه رنگ اصلی ایجاد کرد. مثلا با ترکیب میزان درستی از RGB رنگ های متنوعی را ایجاد کرد.



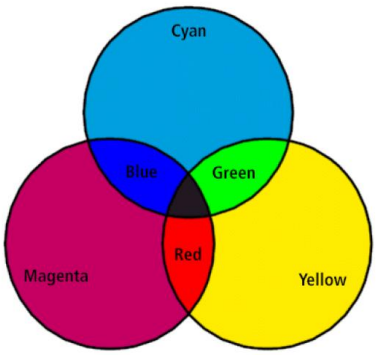
Red

Green

Blue

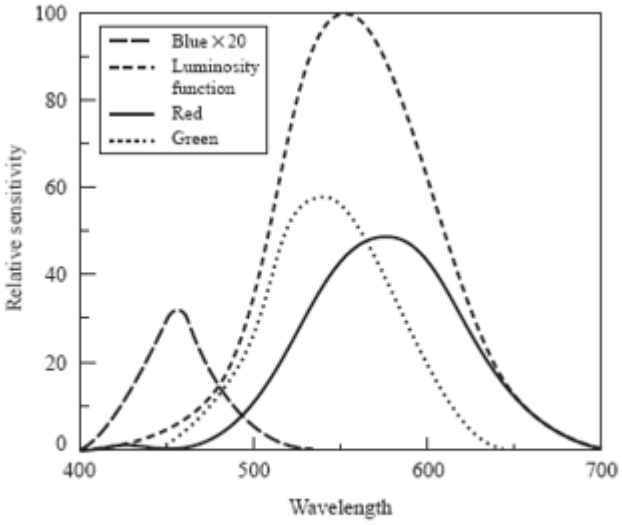


Magenta = Red + Blue
Cyan = Blue + Green
Yellow = Green + Red



Magenta = White - Green
Cyan = White - Red
Yellow = White - Blue

مدل های رنگی RGB, CMY, YIQ در سیستم ها استفاده می‌شود.



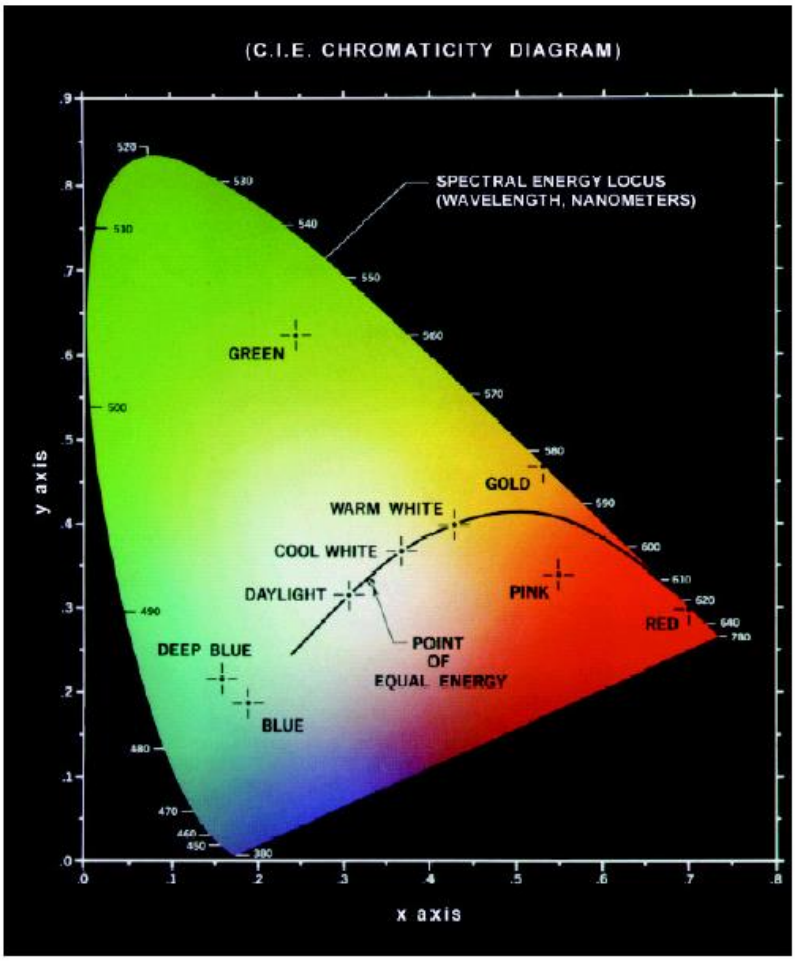
$$C_i = \int C(\lambda) a_i(\lambda) d\lambda, \quad i = r, g, b, y$$

در شکل فوق Luminance بیشترین انرژی را دارد که برای تلویزیون سیاه و سفید استفاده می‌شود. که معادل Y در تلویزیون است. حتی اگر ویدیو رنگی به تلویزیون سیاه سفید ارسال شود باید چیزی را ببینند بنابراین معادل بافت می باشد.

اگر برای رنگ سه رنگ X,Y,Z در نظر بگیریم بطوریکه:

$$x + y + z = 1 \qquad x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z}.$$

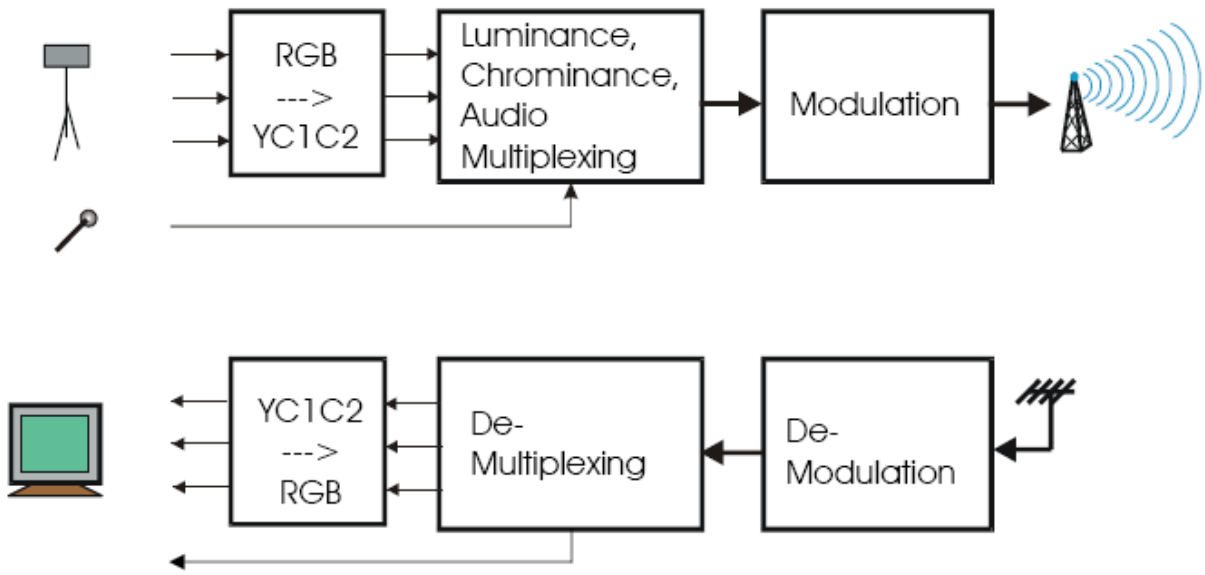
می‌توان رنگ‌های دیگر را از روی آنها ساخت. نمودار CIE سه رنگ پایه X,Y,Z را نشان می‌دهد که ترکیب این سه رنگ انواع رنگ های دیگر را ایجاد می‌کند.



در نمودار CIE هر مثلثی را در نظر بگیریم می توان رنگ های داخل مثلث را با ترکیب آن سه رنگ ایجاد کرد. اندازه بزرگی این مثلث وابسته به تکنولوژی و خواص چشم است. RGB در پردازش مهم است ولی در ارسال مهم نیست چون تغییر شدت باعث تغییر همه رنگ های R, G, B می‌شود یعنی به هم Correlate هستند. از نظر تئوری اطلاعات، اطلاعاتی که می فرستیم نباید به هم correlated باشند. در ویدیو از Y, Cr, Cb استفاده می شود که در فرستنده RGB به YCrCb تبدیل می شود و در گیرنده تبدیل معکوس استفاده می‌شود.

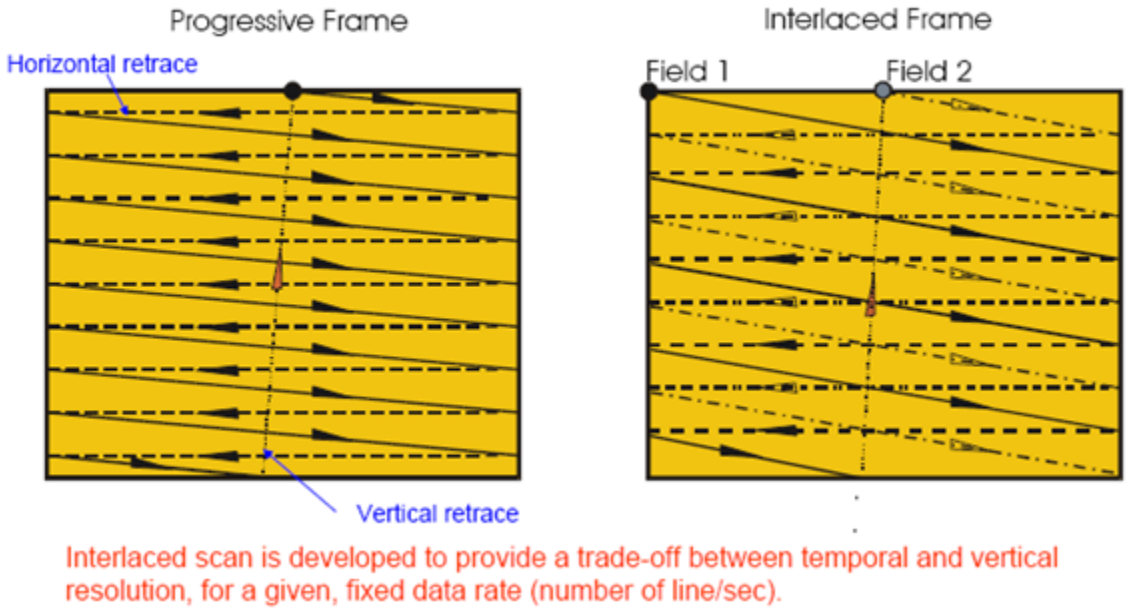
$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.257 & 0.504 & 0.098 \\ -0.148 & -0.291 & 0.439 \\ 0.439 & -0.368 & -0.071 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.164 & 0.000 & 1.596 \\ 1.164 & -0.392 & -0.813 \\ 1.164 & 2.017 & 0.000 \end{bmatrix} \begin{bmatrix} Y - 16 \\ C_b - 128 \\ C_r - 128 \end{bmatrix}$$



YCrCb در قدیم ۸ بیتی بودند و محدوده اعداد -۱۲۸ تا +۱۲۷ را شامل می‌شوند. Cr, Cb با ۱۲۸ جمع می‌شوند و مثبت می‌شوند. اما به Y ۱۶ تا اضافه می‌شود زیرا Y از ۰۰۰۰۰۰۰۰ تا ۱۱۱۱۱۱۱۱ متغیر است اگر یک خطا روی اینها اتفاق بیافتد(rounding error) تغییرات در لبه ها زیاد است. مثلا 00000000 اگر منهای ۱ شود به ۲۵۵ تبدیل می‌شود و سیاه تبدیل به سفید می شود همچنین اگر به 11111111 یک اضافه شود تبدیل به صفر می شود و سفید تبدیل به سیاه می شود. ۱۶ برای پوشش خطا هست. بنابراین ماکزیمم را ۲۳۵ در نظر می گیرند تا با ۱۶ درست شود. (در حالت عادی در صورت بروز خطا زمینه سیاه← سفید، زمینه سفید← سیاه). Y در برگیرنده بافت (texture) (روشنایی) است و Cr, Cb مشخصات رنگی می‌باشند. آنالیز حرکت و جزیئات روی Y انجام می‌شود. حدود ۸۵ درصد اطلاعات در Y و ۷ درصد در Cr,Cb می‌باشد.

رستر اسکن:



در جاروب progressive: n خط را می خواهیم جاروب کنیم از بالا خط به خط حرکت می کنیم و به سمت پایین حرکت می کنیم. هرچه تعداد خط ها بیشتر resolution بیشتر است.

Interlaced: در قدیم تکنولوژی صفحه نمایش ضعیف بود. در این حالت تعداد خط ها را کم کردند تا پهنای باند کم شود و در دو مرحله یک بار field1 و بار بعد field2 را جاروب می کنیم. در تصویر اول filed1 و در تصویر دوم field2 ارسال می‌شود. اینکه در ثانیه چند عکس ارسال شود تا flicker نداشته باشیم. حداقل باید ۵۰ فریم در ثانیه فرستاد. در این حالت ۲۵ فریم از نوع فیلد۱ و ۲۵ از نوع فیلد۲ ارسال میشد برای نصف کردن پهنای باند.. چشم ما به اجسامی که ثابت هستند حساس هست به اجسامی که زیاد و با سرعت بالا تغییر می کنند حساس نیست و جزئیاتش را نمی بینیم. یعنی ۵۰ عکس با نصف جاروب. در سیستم آنالوگ برای نصف کردن پهنای باند استفاده می شد. در دیجیتال به این نیازی نداریم. ولی برای سازگاری بین دیجیتال و آنالوگ نیاز است. در دیجیتال مشکلی نداریم برای ارسال عکس. گاهی در دیجیتال ۵ تصویر در ثانیه ارسال می شود و دیکدر آنها را ۱۰ بار تکرار می کند و تصویر freeze می‌شود. چون صفحه ای که نگاه می کنیم خاموش و روشن میشود. باید تعدادی تکرار شود تا چشمک نزند. در سیستم progressive خط ها به هم نزدیکترند نسبت به interlaced. در progressive ۲۵ فریم با اندازه کامل داریم و در interlaced ۵۰ فریم نصف عکس. فشرده سازی کدام بهتر هست؟ در حالیکه هر دو از لحاظ تعداد خط مثل هم هستند؟ ساده ترین فشرده سازی از تشابه بین پیکسل ها استفاده می کند که در این حالت progressive بهتر است. گاهی هم از تشابه بین فریم ها هم برای فشرده سازی استفاده می‌شود. برای بین فریم‌ها، فشرده سازی interlaced بهتر که دارای سرعت زیاد است و progressive سرعت کمتر است. در بعضی جاها بصورت adaptive کار می شود یعنی در سرعت بالا از progressive و سرعت کم از interlace استفاده می‌شود.

جلسه ۱۳۹۶/۷/۳

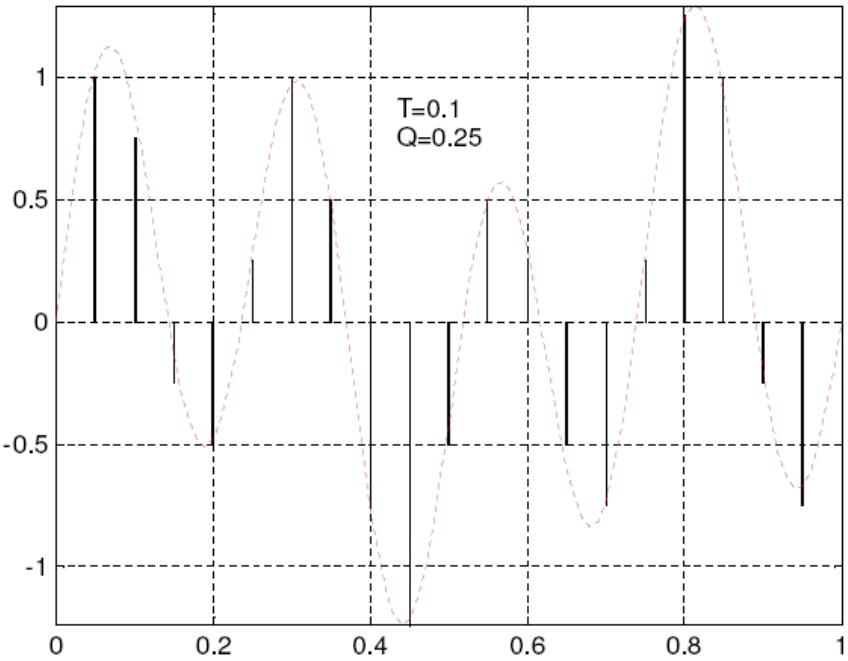
در هر لحظه می‌توان freeze کرد در یک لحظه که نور می‌تابد می‌توان از آن عکس گرفت. Charge coupled Device (CCD)

ویدئو با زمان در تغییر است. در هر ۲۰ میلی ثانیه باید کل صفحه را پویش کرد و خواند تا در یک ثانیه، ۵۰ فریم گرفته شود. در progressive، ۵۰ تا فریم کامل داشت (می توان ۲۵ تا خواند ولی هر صفحه را دوبار نشان داد). در interlaced خواندن نصف خط ها در هر ۲۰ میلی ثانیه.

در گیرنده: میزان RGB مشخص شده و روی صفحه تابیده می‌شود یا بصورت progressive یا .interlaced

در تلویزیون‌های HD، ۱۹۲۰ ستون و ۱۰۸۰ خط داریم در قدیمی ها ۷۲۰ ستون و ۵۷۶ خط داریم. تحقیقات روی ۱۶۰۰۰ پیکسل است. ولی دوربین ۱۶۰۰۰ پیکسلی نداریم.

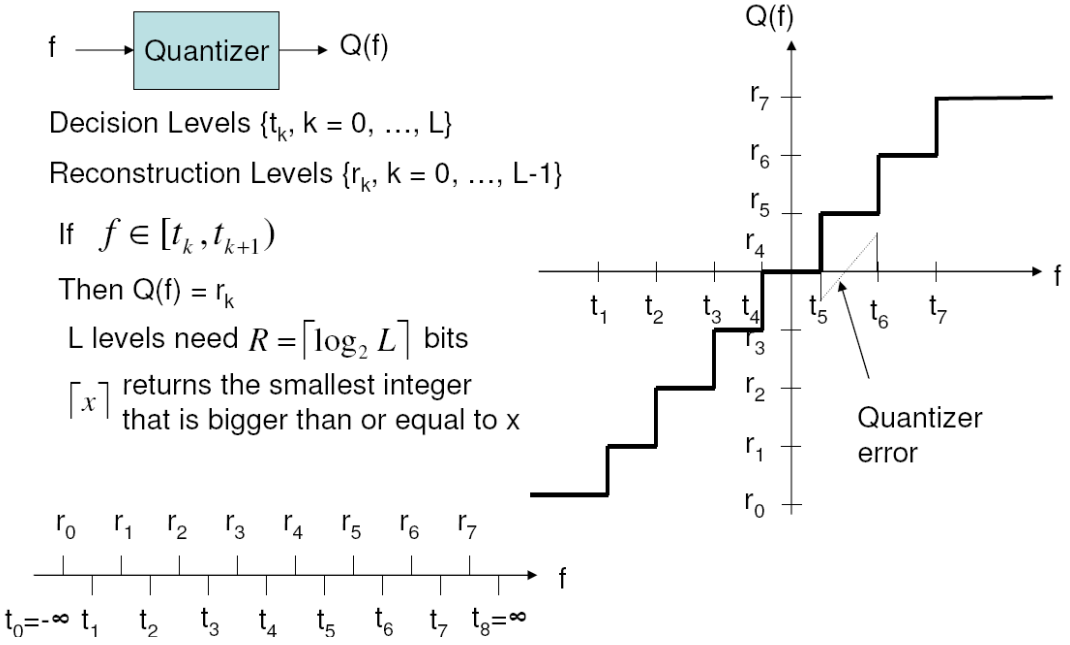
امروزه ویدیوها روی ۱۰ بیت: دو آی سی به جای یک آی سی. ۶ بیت به چپ شیفت داده شود ۱۶ بیت می‌شود، ۶ بیت LSI، صفر است. پردازش ویدئو بین ۰ تا ۲۵۵ است. اگر از محدوده خارج باشد باید محدود شود. YCrCb پردازش می شوند. Y مثبت هست ولی Cr,Cb مثبت و منفی هستند به همین خاطر با ۱۲۸ جمع می شوند تا مثبت شوند. برای luminance، ۱۶ تا اضافه می کنیم برای پوشش خطا. در پردازش RGB اگر از صفر کمتر بودند باید آنها را صفر کنیم و اگر بیشتر از ۲۵۵ بود روند شود. در مدولاسیون سیگنال شیفت داده می‌شود. مثلا همه سیگنال شیفت داده شود به ۱۰۰مگ.



تبدیل سیگنال از آنالوگ به دیجیتال: برای منقطع کردن زمان از sampling استفاده می‌شود. در Sampling طبق تئوری شانون اگر نمونه برداری دو برابر فرکانس باشد هیچ چیزی از دست نمی دهیم.

$G(t)=f(t).S(t)$

$G(F)=F(f).S(t)$



Quantize کردن باعث از بین بردن اطلاعات می شود. مثلا اگر دامنه کوانتایز ۱ باشد و در شکل فوق فاصله t4,t5 اعداد 0.5,+0.5- باشد همه را صفر در نظر می گیریم یعنی ۰.۳ هم صفر هست و ۰.۴ هم صفر هست. سیگنال در مسیر انتقال ضعیف می شود و در گیرنده تقویت می شود که تقویت سیگنال، نویز تولید می‌کند. مزایای دیجیتال storage و transmission است. با کاهش فاصله بین quantizeها می‌توان subjective loss lessکرد. (subjective loss less: یعنی دامنه کوانتایز را طوری در نظر می گیریم که rounding از نظر چشمی تاثیری نداشته باشد).

Uniform Quantization

- Equal distances between adjacent decision levels and between adjacent reconstruction levels
  - $t_l - t_{l-1} = r_l - r_{l-1} = q$
- Parameters of Uniform Quantization
  - L: levels ($L = 2^R$)
  - B: dynamic range $B = f_{max} - f_{min}$
  - q: quantization interval (step size)

    $q = B/L = B2^{-R}$
- Quantization function

$$Q(f) = \left\lfloor \frac{f - f_{\min}}{q} \right\rfloor * q + \frac{q}{2} + f_{\min}$$

Note: $\lfloor x \rfloor$

returns the biggest integer that is smaller than or equal to x

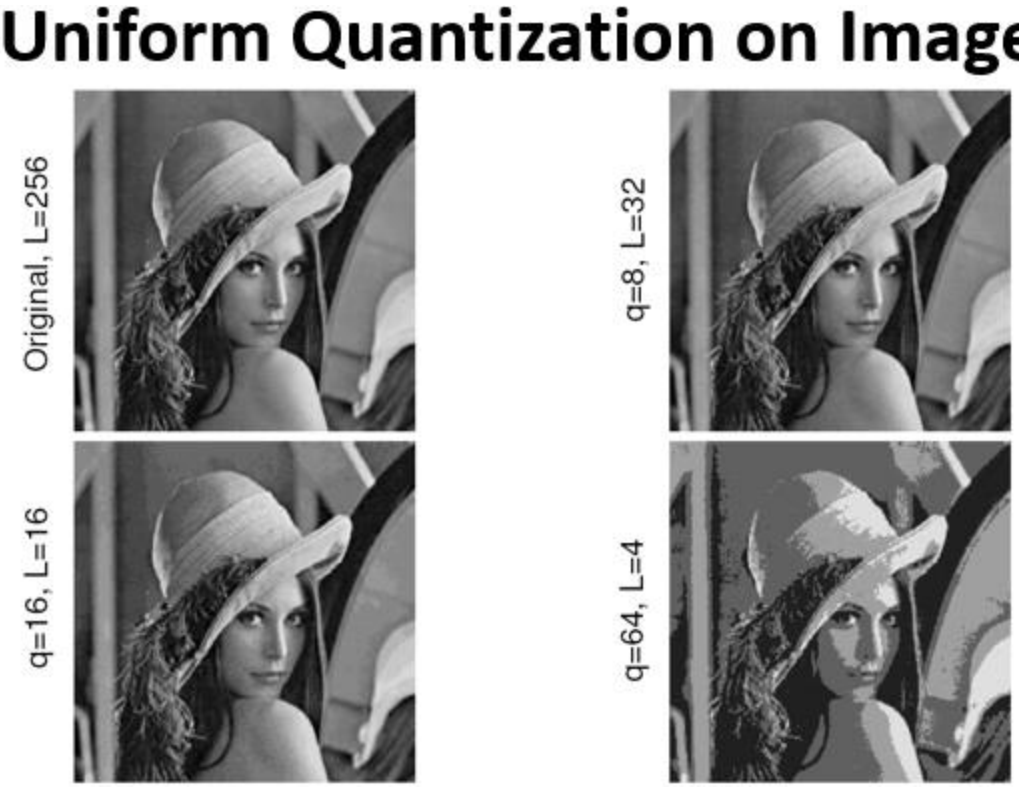
مثال ۱: فرض کنید سیگنالی پیوسته داریم که از خروجی CCD با رنج 0.0-5.0 بدست می‌آید و بخواهیم ۸بیت برای ارسال داشته باشیم. بنابراین $L=2^8=256$ و $Q=5.0/256$ که خروجی بازه ها بصورت زیر خواهد بود [I*q, (I+1)*q). و برای باز سازی از

$rl = l \times q + q/2,\ l = 0,\ldots, 255.$ استفاده می کنیم.

مثال ۲: سیگنال ورودی گسسته هست و شامل ۲۵۶ سطح خاکستری. با چهار سطح L=4 (R=2) داریم: $F_{min}=0,\ f_{max}=256,$

$q=256/4=64$و $q/2=32$ و $Q(f) = \lfloor f / 64 \rfloor \times 64 + 32$

در شکل زیر عکس یک خانم با ۴ سطح مختلف ارسال شده است در بالای سمت چپ که عکس اصلی است ۸ بیت برای ارسال استفاده شده است و $L=256=2^8$. در عکس بالای سمت راست L=32, Q=8 و از ۵ بیت استفاده شده است که مشاهده می شود هنوز کیفیت خوب است. در شکل پایین سمت چپ از $L=16=2^4$ استفاده شده است و Q=16 که هنوز کیفیت خیلی خراب نیست ولی در شکل سمت راست پایین که $L=4=2^2$ و Q=64 کیفیت خراب شده است و به آن contouring distortion می‌گویند.



Sampling را از دوبرابر پهنای باند بیشتر در نظر می گیریم، سوال چند دامنه در نظر بگیریم؟ سیگنالی که کوانتایز می‌کنیم در آن distortion ایجاد می کنیم. یعنی هر عددی در دامنه را به یک عدد نگاشت می‌دهیم و مقداری از آن را از دست می‌دهیم. یعنی مقدار اصلی را نداریم. ما می‌توانیم این error را بگیریم. طریقه محاسبه آن بصورت زیر است:

$$MSE = \sigma_q^2 = E\{(Q(f) - f)^2\} =$$

$$\int_{t_0}^{t_L} (f - Q(f))^2 \, p(f) df = \sum_{l=0}^{L-1} \int_{t_l}^{t_{l+1}} (f - r_l)^2 \, p(f) df$$

که p(f) احتمال تابع چگالی f است. MSE برای یک تصویر دلخواه برابر است با:

$$MSE = \sigma_q^2 = \frac{1}{MN} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (f(i,j) - Q(f(i,j)))^2$$

این یک واحد سنجش هست هر سیستمی که بهتر باشد MSE=mean squared error آن کمتر باید باشد. (M,N ابعاد تصویر است). چون تغییرات با تغییر دامنه خیلی زیاد است و به توان ۲ می رسد بهتر است از لگاریتم استفاده کنیم تا هر ۱۰ تغییر در اعداد یک تغییر ایجاد کند.

SNR=signal to noise ratio :SNR, PSNR

$$SNR(dB) = 10\log\frac{\sigma_f^2}{\sigma_q^2}$$

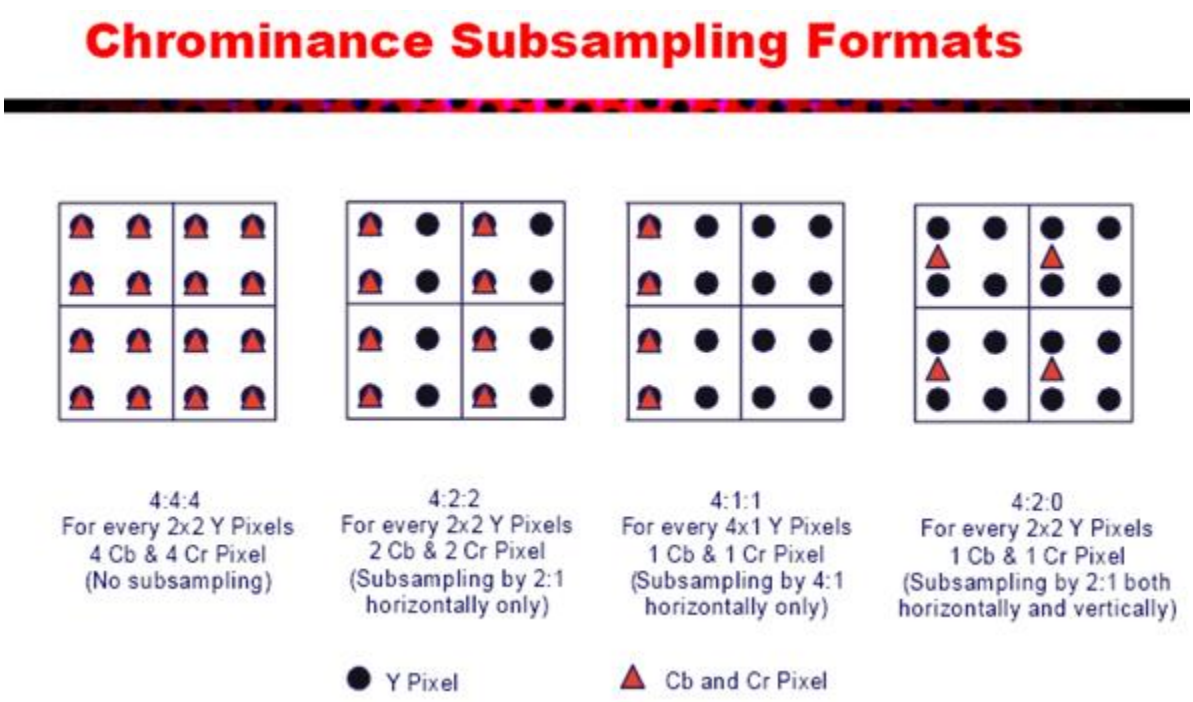
$$\text{Signal Variance} = \text{Signal Energy} = \sigma_f^2 = \frac{1}{MN}\sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(f(i,j)-mean)^2$$

Peak SNR=PSNR: برای تصویر ۸ بیتی peak =255

$$PSNR(dB) = 10\log\frac{255^2}{\sigma_q^2}$$

Log 2=0.3. برای ویدیو S/n= 6n+11و برای ویدیو برای 65دسی بل داریم.n=9bit ➔ 6n= 54➔6n+11=65

برای صدا S/N= 6n+2. مثلا برای صدای 80db داریم: n=13➔ 6n=78➔ 80 =6n+2



در سیگنالی که بصورت YCrCb نشان می‌دهیم Y خیلی مهم است. هر پیکسل دارای یک شدت روشنایی Y است و دارای یک Cr,Cb هست. چون رنگ texture ندارد می‌توانیم پیکسل‌های مشابه را با رنگ مشابهی در نظر بگیریم و کیفیت تغییر نکند یعنی تعداد chrominance را کم کنیم بدین منظور چند فرمت تعریف شده است. مرجع Y است مثلا در فرمت 4:4:4 در هر ۴ پیکسل

۴ تا روشنایی و ۴ تا Cr و ۴ تا Cb داریم در حقیقت فشرده‌سازی نداریم. در 4:2:2 هر ۴ پیکسل 2*2 دارای ۴ تا روشنایی و ۲ تا Cr و 2 تا Cb می‌باشد. یعنی هر دو پیکسل همسایه بطور متوسط یک رنگ می دهد. در فرمت 4:2:0 هر ۲*۲ پیکسل یک Cr و یک Cb دارد که متوسط رنگ ۴ پیکسل است. در فرمت 4:1:1 در هر سطر با ۴ پیکسل یک Cr و یک Cb داریم. اینها فرمت های ضبط هست. 4:4:4 برای عکس های پزشکی است که کیفیت بالایی دارند و دقت مهم است. 4:2:2 در تلویزیون در استودیو استفاده می‌شود. برای ارسال و ویدیو اینترنت و تلویزیون 4:2:0 است. تعداد luminance ثابت است و برای هر پیکسل یک luminance داریم. ولی تعداد Cr,Cb را می توانیم کم کنیم و distortion بیشتری را می تواند تحمل کند زیرا چشم ما زیاد به جزئیات حساس نیست.

مثلا یک تصویر M*N با فرمت 4:2:0، تعداد M*N روشنایی Y دارد ولی M*N/4 تعداد Cr,Cb دارد.

Digital Video Formats

Video Format	Y Size	Color Sampling	Frame Rate (Hz)	Raw Data Rate (Mbps)
HDTV Over air. cable, satellite, MPEG2 video, 20-45 Mbps				
SMPTE296M	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				
BT.601	720x480/576	4:4:4	60I/50I	249
BT.601	720x480/576	4:2:2	60I/50I	166
High quality video distribution (DVD, SDTV), MPEG2, 4-10 Mbps				
BT.601	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

جلسه ۹۶/۷/۱۰

اندازه گیری کیفیت ویدئو: Objective assessment(کمّی): باید یک meter داشته باشیم که بتوان کیفیت ویدیو را اندازه گیره کرد.

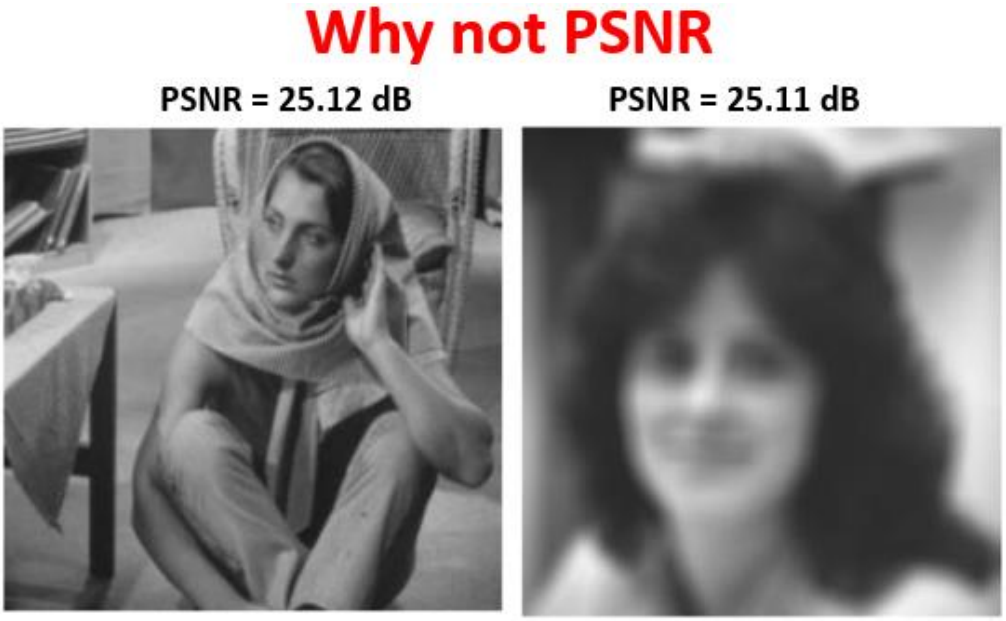
Y_r=reference video.

Y_d= decoded video

$$MSE=\frac{1}{m\times n}\sum_{x=1}^{m}\sum_{y=1}^{n}\left(Y_r(x,y)-Y_d(x,y)\right)^2$$

$$PSNR=10\times log_{10}\left(\frac{255^2}{MSE}\right)dB$$

PSNR: قابل قبول هست؟ مشخص نیست. PSNR واحد سنجش زیاد خوبی نیست. از PSNR نمی‌شود برای Contentهای مختلف استفاده کرد. ولی برای یک عکس مشابه خوب هست. یک عکسی که PSNR=26 دارد از همان عکس با PSNR=25 بهتر است.



Q. Huynh-Thu and M. Ghanbari, 'The scope of validity of PSNR in image/video quality assessment, *Electronic Letters*, **44:13**, (June 2008), pp. 800-801

Structural Similarity Index Measurement (SSIM): SSIM برای کیفیت متوسط خوب هست.

y→decoded ،x→reference

$$SSIM(x,y) = f(l(x,y), c(x,y), s(x,y))$$

SSIM سه مقایسه براساس موارد زیر دارد:

- luminance, l(x,y)
- contrast c(x,y)
- Structure s(x,y)

۱- **Luminance comparison: l(x,y)**
ابتدا میانگین luminance محاسبه می شود

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$$

سپس مقایسه luminance اجرا می‌شود:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

۲- contrast c(x,y):

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right)^{\frac{1}{2}}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

۳- Structure s(x,y)

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$$

۴- محاسبه نهایی SSIM:

$$SSIM(x,y) = [l(x,y]^{\alpha}[c(x,y)]^{\beta}[s(x,y)]^{\gamma}$$

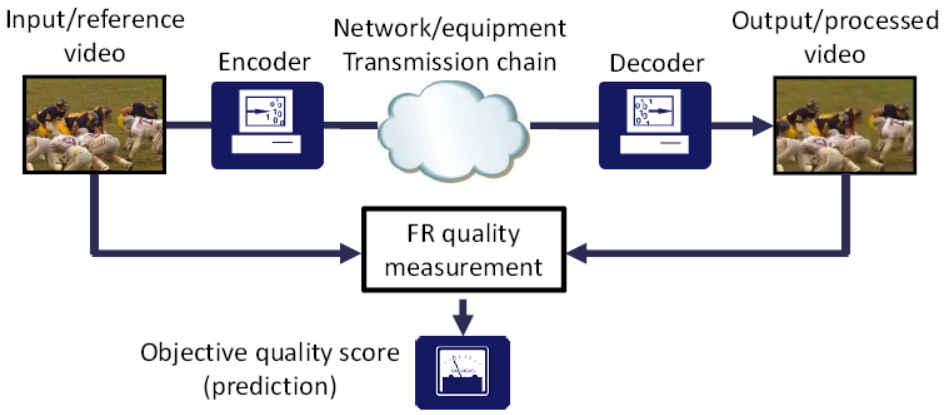
$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2}$$

VQEG: **Video Quality Experts Group** سه روش برای اندازه گیری کیفیت ویدیو معرفی کرد که عبارتند از:

۱- Full reference method(FR) ۲- Reduced Reference method(RR) ۳- No reference method

بیش از ۷۰ درصد ترافیک اینترنت ویدیو هست و تا ۲۰۲۰ به ۸۵ درصد میرسد بنابراین محاسبه کیفیت ویدیو بسیار مهم هست.

Full reference meter: اندازه گیری FR مقایسه را بین یک سیگنال ویدئوی مرجع در سیستم ورودی و سیگنال ویدئوی در سیستم خروجی انجام می دهد.

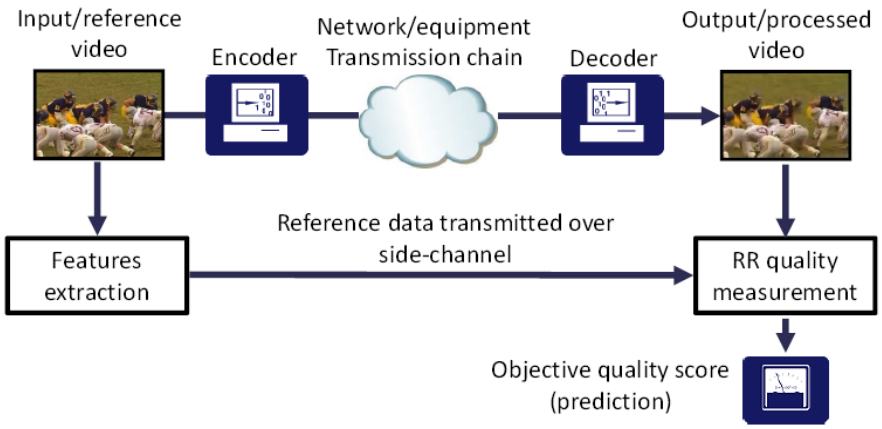


در این روش ویدئوی اصلی را داریم و می خواهیم ویدئو پردازش شده چقدر خوب است. مسائلی مثل delay, jitter همچنین فشرده سازی و packet loss روی کیفیت ویدیو تاثیر دارد. مساله این هست آیا رفرنس را داریم؟ هزینه انتقال ویدئو خیلی بالاست. ولی برای مقایسه انکدرها با هم قابل استفاده است ولی در شبکه مفید نیست.

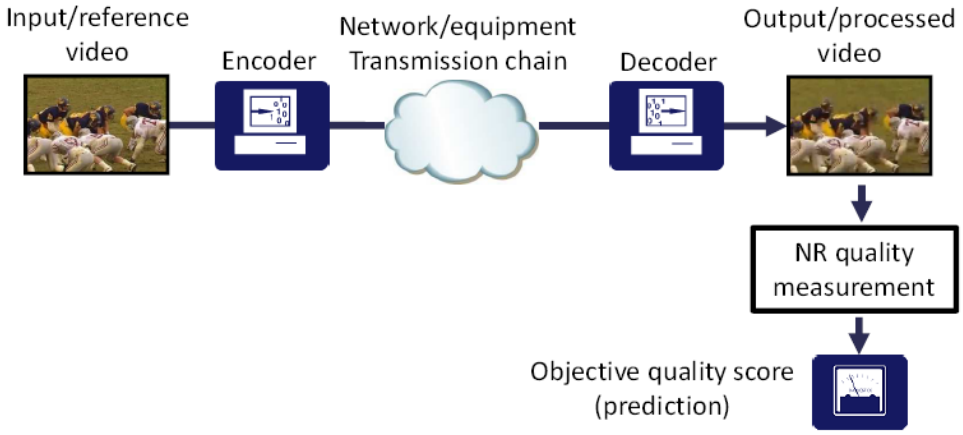
Reduced-reference Method

- در روش اندازه‌گیری کیفیت RR، پارامترهای خاصی (ویژگی‌هایی) از هر دوی سیگنال‌های مرجع و پردازش شده استخراج می‌شوند.
- داده‌های مرجع مرتبط با این پارامترها با استفاده از یک کانال کناری (side channel) به سیستم اندازه‌گیری ارسال می‌شوند.
- سیستم اندازه‌گیری ویژگی‌های مشابهی را استخراج می‌کند تا مقایسه را انجام دهد و یک کیفیت را مقایسه می کنیم.

در این روش ویدیوی اصلی را داریم و پارامترها را همراه با ویدیوی کد شده به گیرنده میفرستیم در گیرنده ویدیو را دیکد می کنیم و همین پارامترها را در آن بدست می‌آوریم و با هم مقایسه می کنیم. (مثلا پارامترهای یک بلاک ۳۲*۳۲). هر چه پارامترهای بیشتری بفرستیم سربار بشتری داریم ولی اندازه گیری ما دقیق تر است. ارسال پارامترها یا بصورت side channel یا بصورت water mark می‌فرستیم.



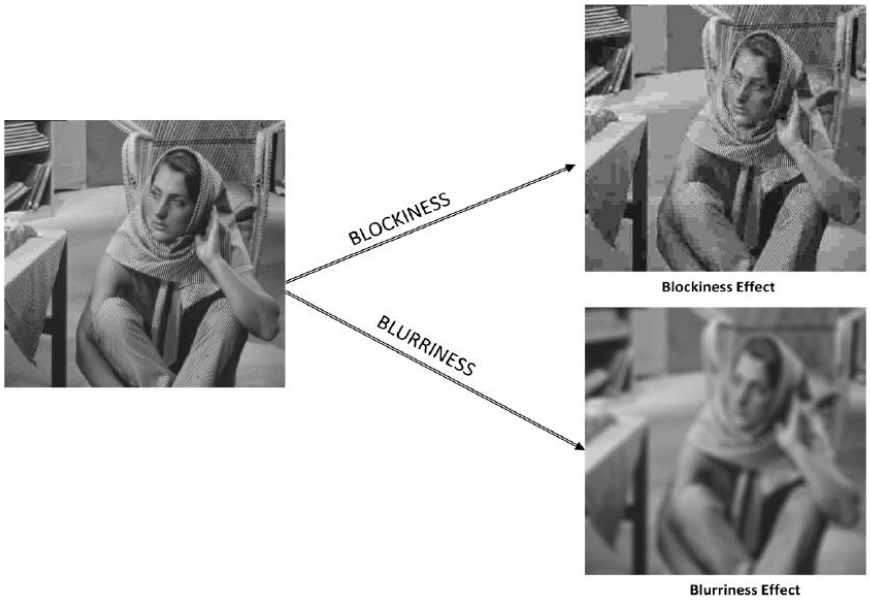
No-reference Method: اندازه‌گیری کیفیت بدون مرجع (NR) فقط ویدئویی پردازش شده را آنالیز می‌کند بدون نیاز به اطلاعات مرجع (کامل یا جزئی)



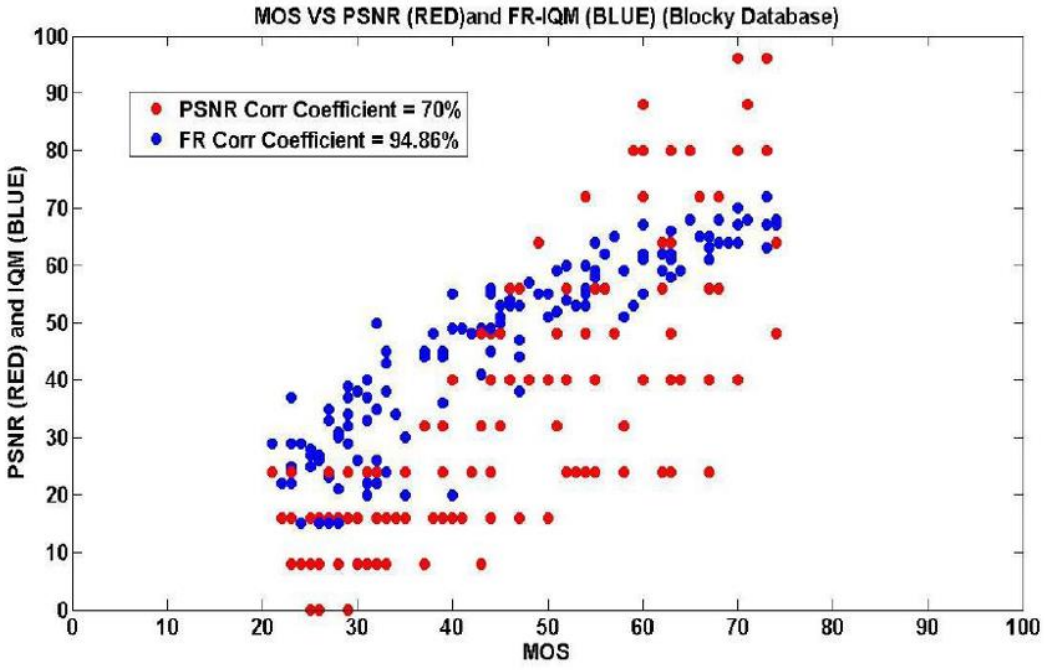
محاسبه کیفیت در این روش خیلی سخت است. چون هیچ رفرنسی نداریم. اولی برای شبکه بدرد نمی‌خورد. اشکال دومی این هست که قبل از ارسال ویدئو روی شبکه باید متادیتای مورد نیاز را بدست آوریم. این روش سوم در شبکه خیلی مناسب هست. Fr از همه بهتر است و NR بدترین است. در جدول زیر سه روش از لحاظ کارایی با هم مقایسه شده‌اند. مقایسه براساس حساسیت آنها براساس پارامترهای مختلف است. اگر یک delay ایجاد شود یا یک پیکسل جابجا شود اندازه گیری آن هم کار مشکلی هست. از لحاظ آماری حساسیت RR مهم هست.

	FR	***RR***	***NR***
Spatial & temporal offset	Very sensitive	Less sensitive	insensitive
False statistics	insensitive	Sensitive	insensitive
Coding Method	Less sensitive	Can be adjusted	Very sensitive
Type of degradation	Less sensitive	Can be adjusted	Very sensitive

بیش از ۹۰ درصد اعوجاجات (distortion) از نوع Blockiness & Bluriness می‌باشد. بنابراین اگر بتوانیم این دو را حدس بزنیم چطور هست می توانیم آن را بسازیم.



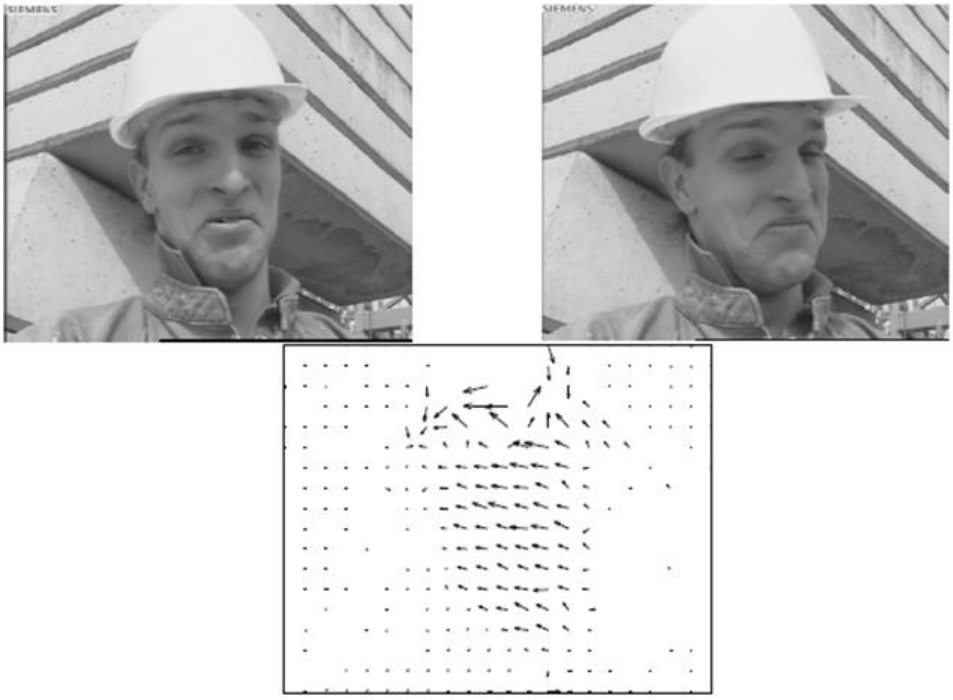
FR vs PSNR



Water marking چیست؟ یعنی اطلاعاتی در داخل خود تصویر مخفی شود بطوریکه دیده نشود. که به آن data hiding می‌گویند. این سیستم خیلی نسبت به فشرده سازی آسیب پذیر (vulnerable) است.

Motion estimation: برای تشخیص اینکه یک شیء در فریم قبلی کجا بوده باید بتوانیم تشخیص بدهیم یک پیکسل در فریم قبلی کجا بوده است.

Sample Motion Field



در بخش بزرگی از بلاک ها هیچ حرکتی نداریم ولی در بعضی بلاک ها مثل محدوده صورت حرکت داریم. ۶۰ درصد processing power کدک ها در همین بخش صرف می‌شود یعنی یک بلاک نسبت به قبلی چقدر تغییر کرده است.

Optical Flow Equation

- When illumination condition is unknown, the best one can do it to estimate optical flow.
- Constant intensity assumption -> Optical flow equation

Under "constant intensity assumption":

$$\psi(x+d_x, y+d_y, t+d_t) = \psi(x,y,t)$$

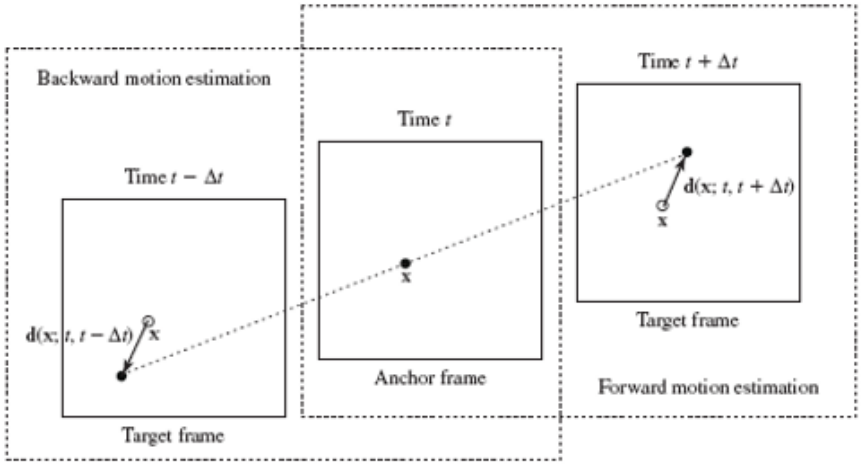
But, using Taylor's expansion:

$$\psi(x+d_x, y+d_y, t+d_t) = \psi(x,y,t) + \frac{\partial \psi}{\partial x}d_x + \frac{\partial \psi}{\partial y}d_y + \frac{\partial \psi}{\partial y}d_t$$

Compare the above two, we have the optical flow equation:

$$\frac{\partial \psi}{\partial x}d_x + \frac{\partial \psi}{\partial y}d_y + \frac{\partial \psi}{\partial t}d_t = 0 \quad \text{or} \quad \frac{\partial \psi}{\partial x}v_x + \frac{\partial \psi}{\partial y}v_y + \frac{\partial \psi}{\partial t} = 0 \text{ or } \nabla \psi^T \mathbf{v} + \frac{\partial \psi}{\partial t} = 0$$

فرض این است که اگر بلاکی جابجا شد، شدت روشنایی آن تغییر نکند. البته این تغییر در ۴۰ میلی ثانیه هست. فرض کنید جسمی در زمان t در مختصات (x,y) باشد و در زمان t+dt در مکان (x+dx, y+dy) باشد. در اینصورت فرمول تغییرات بصورت شکل فوق است. به عبارتی ما زمانی می‌توانیم motion vector را تشخیص دهیم که detail داشته باشیم. یعنی بتوانیم تغییرات (x,y) را ببینیم. هر چه detail تصویر بیشتر باشد دقت بیشتر است.



Anchor frame: $\psi_1(\mathbf{x})$

Target frame: $\psi_2(\mathbf{x})$

Motion parameters: $\mathbf{a}$

Motion vector at a pixel in the anchor frame: $\mathbf{d}(\mathbf{x})$

Motion field: $\mathbf{d}(\mathbf{x};\mathbf{a}), \mathbf{x} \in \Lambda$

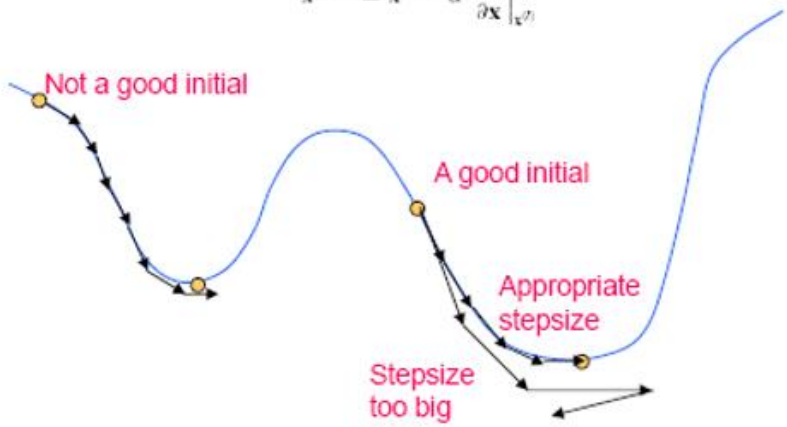
Mapping function:

$$\mathbf{w}(\mathbf{x};\mathbf{a}) = \mathbf{x} + \mathbf{d}(\mathbf{x};\mathbf{a}), \mathbf{x} \in \Lambda$$

Lost concealment: اطلاعاتی در جایی از تصویر گم شده است بتوانیم از بخش دیگری آن را جایگزین کنیم یعنی خرابی از دید چشم پنهان باشد. در اینجا باید motion معلوم باشد تا بدانیم بلاک از کجا آورده شود.

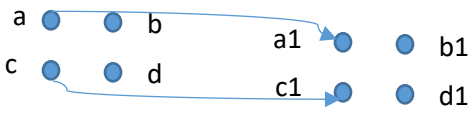
Gradient Descent: فرض کنید می خواهیم motion جسمی را محاسبه کنیم. یک بلاک مثلا ۸*۸ را در یک فریم می گیریم و با بلاک های فریم دیگر مقایسه می کنیم تا حرکت آن را در بلاک دیگر پیدا کنیم. این عمل خیلی complex هست البته روش‌های سریعی هم هست که می‌توان آن را سریعتر بدست آورد.

$$\mathbf{x}^{(l+1)} = \mathbf{x}^{(l)} - \alpha \left.\frac{\partial J}{\partial \mathbf{x}}\right|_{\mathbf{x}^{(l)}}$$

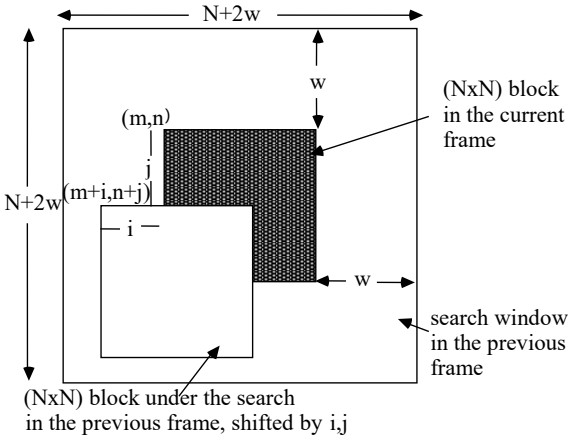


در این روش نقطه شروع بسیار مهم هست که از کجا شروع کنیم که به نقطه مطلوب برسیم.

اگر یک نقطه در مختصات (x,y) باشد و در فریم جدید در نقطه (x+dx,y+dy) باشد در اینصورت (dx,dy) بردار حرکت آن نقطه است. اما یک نقطه در فریم N ممکن است با هزاران نقطه در N+1 تطابق داشته باشد به جای یک پیکسل می‌توان از چند پیکسل استفاده کرد. هر چه بلاک بزرگتر باشد reliable تر هست ولی همه پیکسل‌ها به اندازه هم تغییر نمی‌کنند. فرض کنید (a,b,c,d) چهار پیکسل در فریم قبلی بوده و (a1,b1,c1,d1) معادل آنها در فریم جدید باشند:



برای پیدا کردن چهار پیکسل در فریم جدید که motion آنها برابر با چهار پیکسل در فریم قبلی باشد باید a1,b1,c1,d1 را طوری بیابیم که $\sum(|a-a_1|^2+|b-b_1|^2+|c-c_1|^2+|d-d_1|^2)$ حداقل باشد. برای پیدا کردن motion vector چقدر باید جستجو کنیم؟ اگر به اندازه w پیکسل به اطراف جستجو کنیم در اینصورت Order الگوریتم $(2w+1)^2$ می‌باشد.

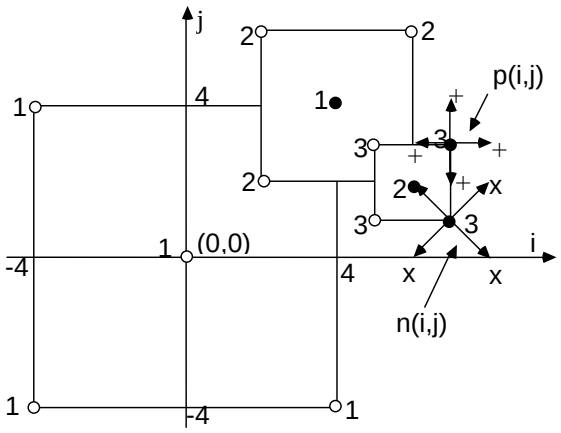


اگر می توانستیم یک پیکسل را پیدا کنیم خیلی عالی بود ولی نمی توانیم هرچه بلاک را بزرگتر کنیم بهتر است اما اگر اندازه بلاک خیلی بزرگ شد حرکت آن representative برای همه پیکسل ها نیست. بلاک هرچه بزرگتر باشد سربار و خرج بیت آن کمتر خواهد شد. اندازه بلاک در خود فریم نیز قابل تغیر است. به این روش که همه جا را جستجو کنیم تا بلاک مورد نظر را پیدا کنیم Exhaustive Block Matching Algorithm(EMBA) می‌گویند. یکی از کاربردهای motion هنگامی هست که در شبکه loss داشته باشیم اگر یک فریم از دست برود از روی فریم قبلی می‌توان با motion vector آن را ساخت. در مخابرات کدک ویدیو با H و صدا با G شروع می‌شود. Streaming مخابرات نیست. مخابرات دو طرفه است. آدرس دهی یکی از مشکلات در پردازش ویدیو هست به همین خاطر اندازه بلاک را توان ۲ می گیرند. یک ماکرو بلاک اندازه ۱۶*۱۶ دارد.

برای افزایش سرعت جستجو می توان از الگوریتم های سریع استفاده کرد. یکی از این روش ها الگوریتم لگاریتمی هست که ابتدا گام را بزرگ می گیریم بعد نقاط گوشه های آن را محاسبه می کنیم بین آنها نقطه ای که نزدیک تر هست را انتخاب کرده سپس در نقطه جدید گام حرکت را نصف می کنیم و این عمل را تکرار می کنیم تا به پیکسل برسیم.

Fast Algorithms for BMA

- Key idea to reduce the computation in EBMA:
  - Reduce # of search candidates:
    - Only search for those that are likely to produce small errors.
    - Predict possible remaining candidates, based on previous search result
  - Simplify the error measure (DFD) to reduce the computation involved for each candidate
- Classical fast algorithms
  - Three-step
  - 2D-log
  - Conjugate direction
- Many new fast algorithms have been developed since then
  - Some suitable for software implementation, others for VLSI implementation (memory access, etc)

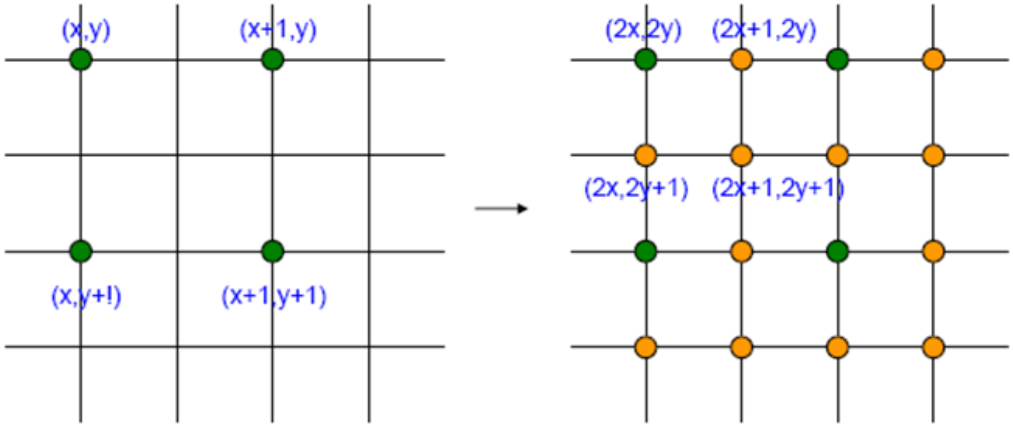


اگر میزان حرکت قبلی را درست بدست آوریم به احتمال ۹۹٪ حرکت بعدی هم به همان میزان خواهد بود. با w=8 می توان در 3- step حرکت را بدست آورد. در هر مرحله ۸ چک داریم.

سوال این هست که چگونه گام حرکت را اعشاری کنیم. دو راه داریم یکی اینکه تصویر را دو برابر کنیم و نقاط جدید را interpolate کنیم.

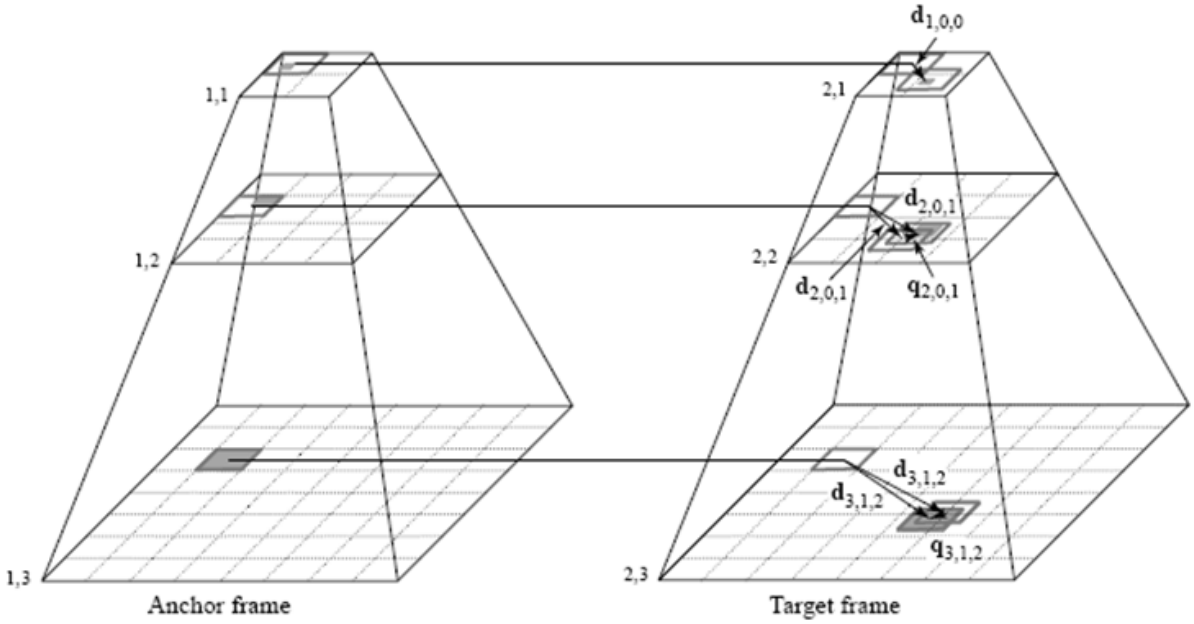
فرض P1,P2 دو پیکسل واقعی است. با میانگین گیری از P1,P2، P_{12}' را می‌سازیم و با میانگین گیری از P1,P_{12}'، P_{112}" را می‌سازیم برای مختصات اعشاری.

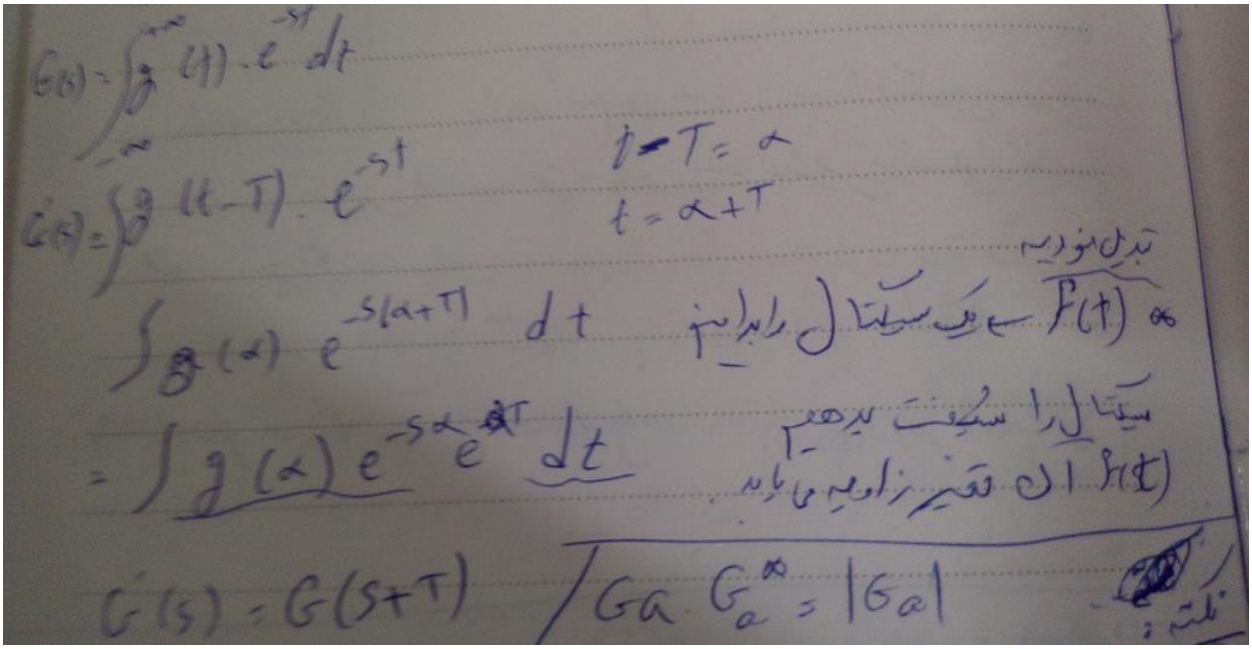
Bilinear Interpolation



O[2x,2y]=I[x,y]
O[2x+1,2y]=(I[x,y]+I[x+1,y])/2
O[2x,2y+1]=(I[x,y]+I[x+1,y])/2
O[2x+1,2y+1]=(I[x,y]+I[x+1,y]+I[x,y+1]+I[x+1,y+1])/4

Hierarchical Block Matching Algorithm (HBMA)





Video Compression: در research، logic مهم است نه فقط نتایج. سرعت و دقت و افزایش ظرفیت خوب هست به شرطی که منطق کار پیدا شود.

Differential Pulse Code Modulation (DPCM)

a-0→a' (quantize). فرستنده: b-a'→(b-a')'

در گیرنده با a' جمع می‌شود و b' را می‌دهد.

Entropy coding: داده‌هایی که بیشتر تکرار می‌شوند را با بیت کمتری می‌فرستد. اما اگر در داده‌ها error داشته باشیم چقدر خطا روی مطالب دریافت شده تاثیر دارد؟ compress را چگونه تا ۲۰۰ برابر افزایش دهیم؟

یک روش این هست که روی بلاک mv را محاسبه کنیم. و برای بلاک بعدی تفاوت mv را با mv قبلی بفرستیم که خود باعث فشرده سازی می‌شود. جلوگیری از انتشار خطا یکی از مسائل مهم در هر کدک هست. Loss less حداکثر سه برابر فشرده سازی می‌توان کرد.

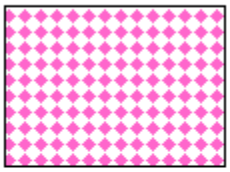
در block based آدرس دهی ساده است، کار روی بلاک بخش می‌شود. Overhead یکی از مسائل مهم در video processing هست. Cost هر کار مهم هست.

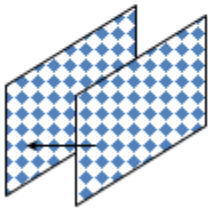
Transformation چگونه باید باشد؟

۱- Linear باشد یعنی برگشت پذیر باشد. عملی که در فرستنده انجام می شود معکوس آن در گیرنده انجام شود.
۲- انرژی را زیاد نکند. (نه از دست بدهیم نه اضافه کنیم).

Principals of compression

- Spatial redundancy reduction (pixels inside a picture are similar)
- Temporal redundancy reduction (Similarity between the frames)
- Statistical redundancy reduction (more frequent symbols are assigned short code words and less frequent ones longer words)





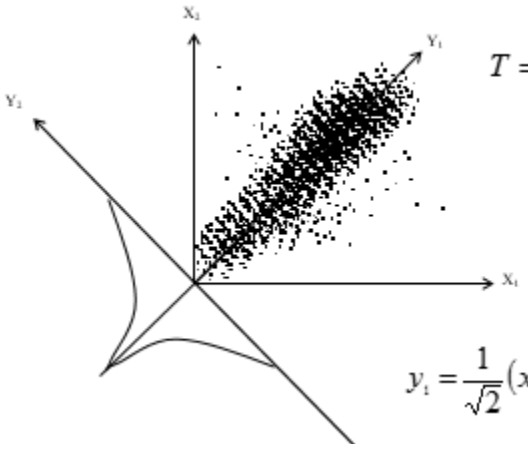
tciifeetekeethe

سه پارامتر داریم که می‌توانیم از آنها برای فشرده سازی استفاده کنیم. اولی این هست که پیکسل های داخل یک تصویر به هم correlated هستند یعنی اگر به یک پارامتر نگاه کنید بقیه را می توانید حدس بزنید. دومی این هست که بین فریم ها هم به هم مربوطند. البته temporal redundancy در image وجود ندارد ولی در ویدیو داریم. سومی این هست که از لحاظ آماری سمبلی که زیاد اتفاق بیافتد را با بیت کمتری بفرستیم و سمبل با فرکانس کمتر را بیت بیشتری بدهیم. از هر سه برای فشرده سازی استفاده می کنیم و همه کدک ها از اینها استفاده می‌کنند.

Spatial redundancy reduction

پیکسل ها داخل یک فریم به هم مربوطند و هر چه فاصله نزدیکتر باشد شباهت بیشتر هست. در گیرنده هم اگر پیکسلی گم شد می توان آن را از روی شباهت بدست آورد. روش های مختلفی برای استفاده از این ارتباط وجود دارد:

- Differential Pulse Code Modulation (DPCM (نمونه برداری می کنیم و quantize می کنیم و ارسال می کنیم و تفاوت ها را ارسال می کنیم. مشکل در انتشار خطا هست اگر خطایی ایجاد شود در تفاوت ها تکرار می‌شود.) در سیستم های circuit switch این خطا کم هست ولی در سیستم packet based امکان از بین رفتن بسته در اثر پر بودن بافرها و .. وجود دارد. همچنین سیستم بی سیم نسبت به خطا خیلی حساس هست. برای بالا بردن نرخ فشرده سازی به جای کار با یک پیکسل روی گروهی از پیکسل ها کار کنیم. هرچه بلاک بزرگتر، سربار کوچکتر. اگر دو پیکسل x1, x2 بغل هم باشند و با هم correlated باشند آنگاه می توان x2 را از روی x1 بدست آورد.



$$T=\begin{bmatrix}\cos 45 & \sin 45\\ \sin 45 & -\cos 45\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

$$[y_1,y_2]=\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}$$

$$y_1=\frac{1}{\sqrt{2}}(x_1+x_2) \qquad \text{and} \qquad y_2=\frac{1}{\sqrt{2}}(x_1-x_2)$$

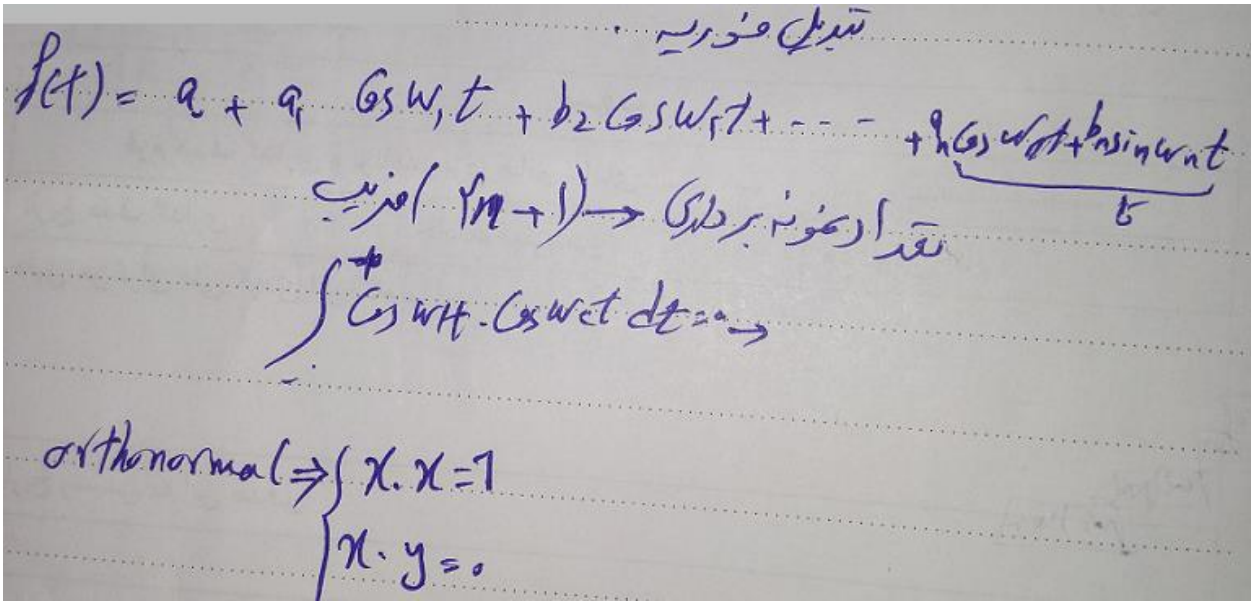
اگر تبدیلی داشته باشیم که Y1Y2 را از روی x1x2 بدست آورد تبدیل باید خطی باشد باید شرایط زیر را داشته باشد:

۱- Y1y2 بیت کمتری از x1x2 نیاز داشته باشد.

۲- X1x2 را بتوان در گیرنده از روی y1y2 بدست آورد.

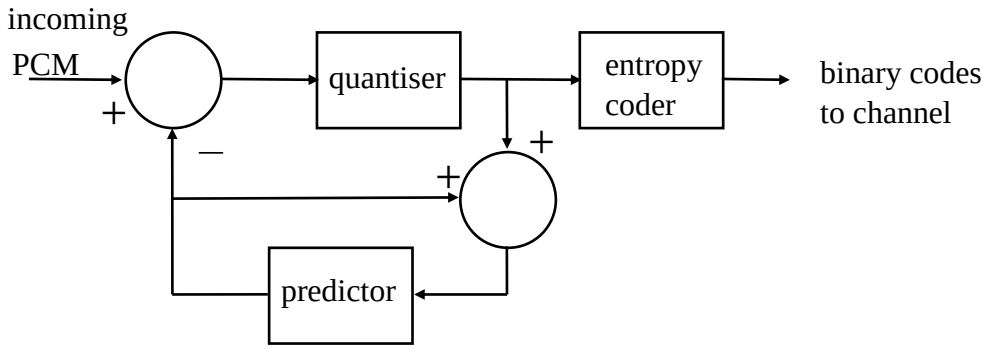
۳- Orthogonal باشد یعنی هر تابع در خودش مقدار دارد ولی در توابع دیگر صفر هست.

۴- Orthonormal باشد. هر تابع در خودش ضرب شود یک می‌شود و در بقیه ضرب شود صفر می‌شود. یعنی انرژی اگر در یکجا متمرکز بشود در دیگری صفر است.

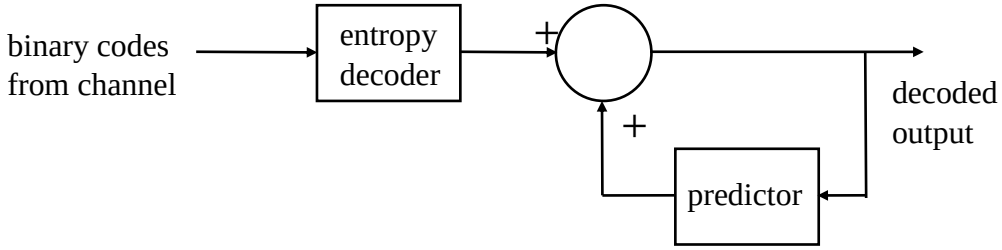


Transform را طوری تعریف می کنیم که انرژی در یک بخش جمع شود و در بقیه جاها صفر شود.

ENCODER

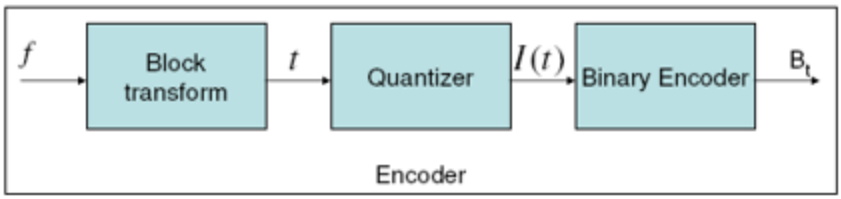


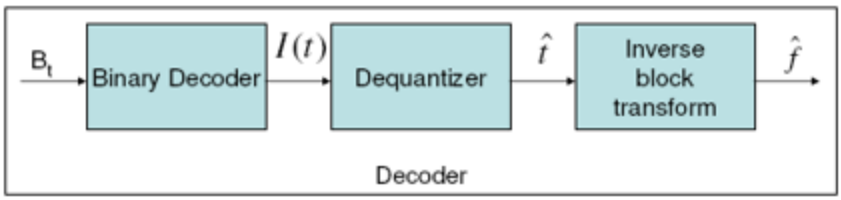
DECODER



- Transform coding
  - Discrete Cosine transform (DCT)
  - Hadamard Transform (HT):
  - Discrete Wavelet transform (DWT)
  - Discrete Sine Transform (DST)
  - Discrete Fourier Transform (DFT)
  - Slant transform

Components in Transform Coding





For block transform coders, we usually use *Unitary (orthonormal) transforms.*

Discrete Cosine Transform (DCT)

- One Dimensional DCT

Forward DCT:

$$F(u) = \sqrt{\frac{2}{N}} C(u) \sum_{x=0}^{N-1} f(x) \cos\left(\frac{\pi(2x+1)u}{2N}\right) \qquad u = 0,1,\ldots,N-1$$

where $C(u) = \sqrt{\frac{1}{2}}$ for $u = 0$

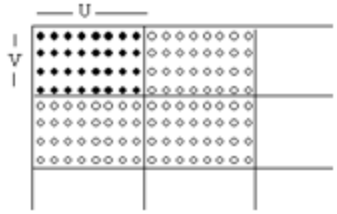
$C(u) = 1$ otherwise

IDCT:

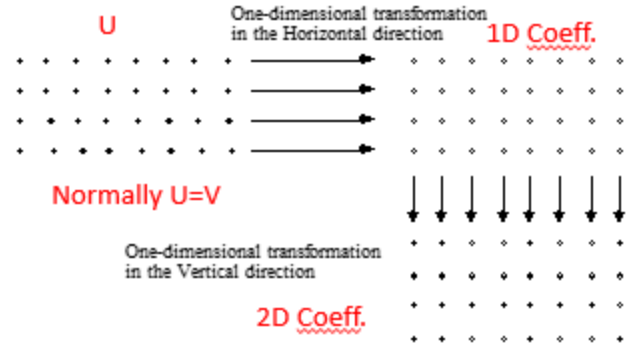
$$f(x) = \sqrt{\frac{2}{N}} \sum_{u=0}^{N-1} C(u) F(u) \cos\left(\frac{\pi(2x+1)u}{2N}\right) \qquad x = 0,1,\ldots,N-1$$

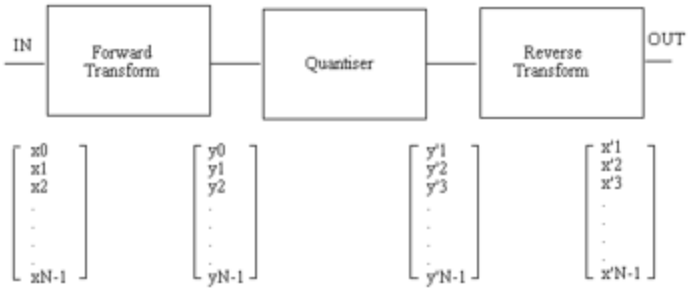
تبدیل دو بعدی: همانطور که پیکسل ها در طول خط correlated هستند در عمودی هم به هم correlated هست.

Two dimensional transform



- A group of U pixels in each line are 1-D transformed.
- This is repeated for V lines.
- A group of V coefficients in the vertical directions are transformed.
- This is repeated for U columns.
- The final output is UV 2-D transform coefficients
- Transform coefficients are quantized for compression
- Compressed coefficients are inverse transformed to reconstruct the image.





در تصاویر کوچک correlation کمتر است در تصاویر بزرگتر مثل HD اندازه بلاک را ۶۴*۶۴ می‌گیرند چون تصویر بزرگ است و correlation هم بیشتر است. در صورتی که کدک های موبایل بلاک را ۴*۴ می گیرند. همیشه دو بعدی از یک بعدی بهتر است. یعنی اگر یک بلاک ۱۶*۱۶ ساده داشته باشیم با پترن مشابه همه ۲۵۶ پیکسل را می توان با یک coefficient نشان داد. یعنی انرژی در یک نقطه تمرکز می‌کند. و می توان texture تصاویر را تجزیه تحلیل کرد و محتواهای مشابه را هم بدست آورد.

البته چون پیکسل ها در زمان هم correlated هستند می توان تبدیل را سه بعدی گرفت ولی در عمل اشکال دارد. چون در دامنه زمان باید برای n فریم بعدی صبر کرد که برای two-way communication مناسب نیست.

DCT and Hadamard 8x8 matrices

Hadamanrd Transform

$$H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}$$ With H0=1

1	1	1	1	1	1	1	1
1	1	1	1	-1	-1	-1	-1
1	1	-1	-1	-1	-1	1	1
1	1	-1	-1	1	1	-1	-1
1	-1	-1	1	1	-1	-1	1
1	-1	-1	1	-1	1	1	-1
1	-1	1	-1	-1	1	-1	1
1	-1	1	-1	1	-1	1	-1

Both transforms are orthonormal, but DCT has a smooth varying basis vector that matches natural images better.

Discrete Cosine Transform DCT:
Basis vectors
COS(kπ(2n+1)/2N)

k&n = 0,.....,N-1
For orthonormality, transform coefficients are divided by $\sqrt{N}$

X= (π/16)

	1	1	1	1	1	1	1	1
√2	cos x	cos3x	sin3x	sin x	- sin x	- sin3x	- cos3x	- cos x
√2	cos2x	sin2x	-sin2x	-cos2x	-cos2x	-sin2x	sin2x	cos2x
√2	cos3x	-sin x	-cos x	-sin3x	sin3x	cos x	sin x	-cos3x
	1	-1	-1	1	1	-1	-1	1
√2	sin3x	-cos x	sin x	cos3x	-cos3x	-sin x	cos x	-sin3x
√2	sin2x	-cos2x	cos2x	-sin2x	-sin2x	cos2x	-cos2x	sin2x
√2	sin x	-sin3x	cos3x	-cos x	cos x	-cos3x	sin3x	-sin x

هدمارد و DCT ارتونرمال هست. چون ضرب هر چیز در خودش یک (۸) می شود ولی در دیگری صفر می شود. هدمارد پترن تصویر نیست ولی DCT هست. چون تغییرات در هدمارد ۱ و -۱ هست یعنی روشن خاموش ولی در طبیعت اینطور نیست.

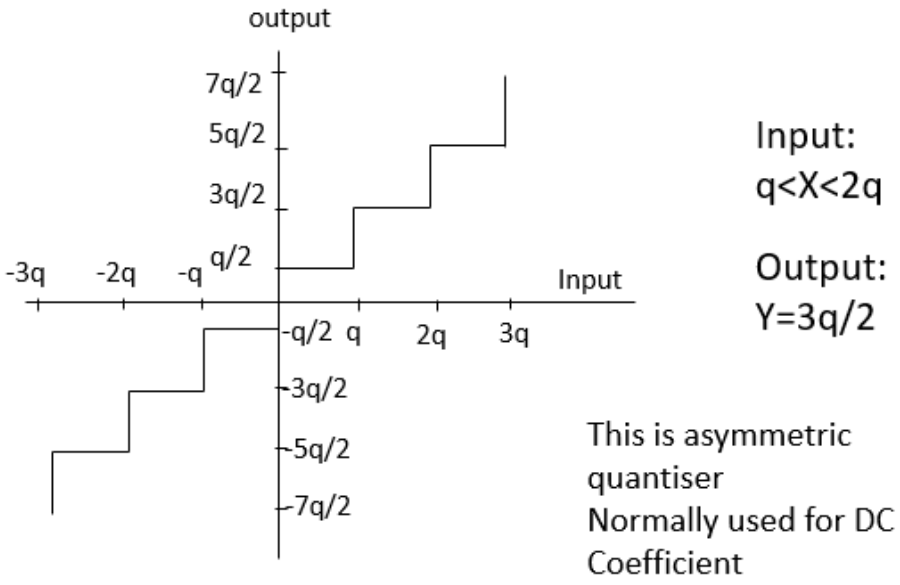
جلسه ۱۳۹۶/۷/۲۹

حساسیت چشم به تغییر در فرکانس های پایین بیشتر است. بنابراین ضرایب فرکانس پایین با q(quantizer) کوچکتر کوانتایز می شود در حالیکه ضرایب فرکانس بالا با q بزرگتر کوانتایز می کند. کوانتایز کردن یعنی نشان دادن اعداد در یک رنج بطوریکه تقریب درست باشد. یک طور گرد کردن است. بنابراین اگر در فرکانس های پایین چشم بهتر کار می کند. می توانیم در فرکانس پایین تقریب دقیق تر داشته باشیم و در فرکانس های بالا ضعیفتر باشد، تقریب را بزرگتر می گیریم:

Step size =2	Step size =10
88→44 90→45 92→46 94→47	80→ 8 90→ 9 100→10 110→11 اعداد سمت راست بیت کمتری می خواهد

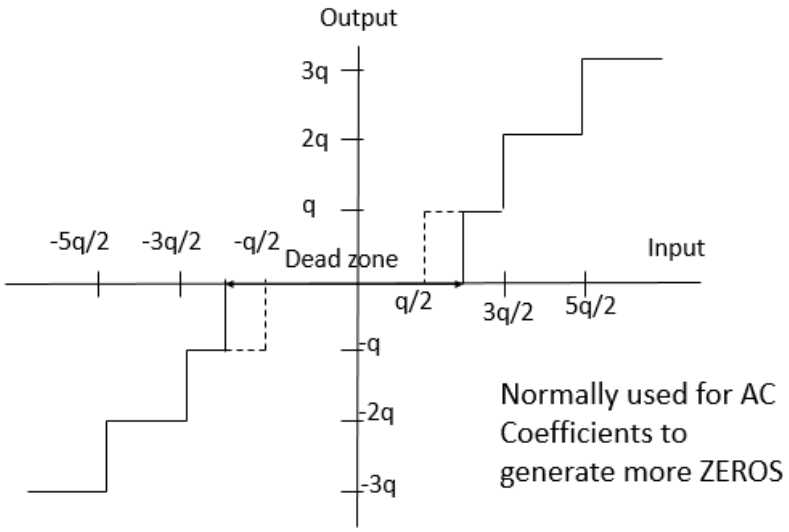
هرچه step size بزرگتر باشد، بیت کمتری برای ارسال نیاز داریم. با quantize همیشه loss داریم. کوانتایز خوب باید subjectively loss less باشد. اگر بخواهیم loss less باشد باید کوانتایز نداشته باشیم. ولی در دیجیتال تقریباً همیشه داریم. کوانتایزایشن باعث فشرده سازی می‌شود هرچه step-size بزرگتر باشد رنج کوچکتر می‌شود و بیت کمتری می خواهد.

Uniform Quantisation



با تغییر دامنه کوانتایزر می توان بیت را کنترل کرد. کوانتایزری خوب است که محدود اعداد کوچک را صفر قرار دهد هدف تعداد صفرها زیاد شود. Dead zone بزرگ شود.

Uniform Quantiser with Dead band zone



Jpeg block 8*8→ 2D DCT

Distortion=(1/64)∑(تفاوت پیکسل به پیکسل)^2

Default Normalization Matrix in JPEG

Normalization Matrix

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

Actual step size for C(i,j): Q(i,j) = QP*M(i,j)

Example: Quantized Indices

÷ Previous matrix elements

```
>>dctblock =dct2(imblock)
dctblock=
 31.0000  51.7034   1.1673 -24.5837 -12.0000 -25.7508  11.9640  23.2873
 113.5766   6.9743 -13.9045  43.2054  -6.0959  35.5931 -13.3692 -13.0005
 195.5804  10.1395  -8.6657  -2.9380 -28.9833  -7.9396   0.8750   9.5585
  35.8733 -24.3038 -15.5776 -20.7924  11.6485 -19.1072  -8.5366  0.5125
  40.7500 -20.5573 -13.6629  17.0615 -14.2500  22.3828  -4.8940 -11.3606
   7.1918 -13.5722  -7.5971 -11.9452  18.2597 -16.2618  -1.4197  -3.5087
  -1.4562 -13.3225  -0.8750   1.3248  10.3817  16.0762   4.4157   1.1041
  -6.7720  -2.8384   4.1187   1.1118  10.5527  -2.7348  -3.2327   1.5799
```

Quantized coefficients ratios to their quantizer step sizes give the indices

```
>>QP=1;
>>QM=Qmatrix*QP;
>>qdct=floor((dctblock+QM/2)./(QM))
qdct =
   2   5   0  -2   0  -1   0   0
   9   1  -1   2   0   1   0   0
  14   1  -1   0  -1   0   0   0
   3  -1  -1  -1   0   0   0   0
   2  -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
```

Only 19 coefficients are retained out of 64

Example: Quantized Coefficients

%dequantized DCT block

>> iqdct=qdct.*QM

iqdct=

32	55	0	-32	0	-40	0	0
108	12	-14	38	0	58	0	0
196	13	-16	0	-40	0	0	0
42	-17	-22	-29	0	0	0	0
36	-22	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

Multiple of indices to the step size results in quantized coefficients values to be used for inverse transform

Note that many coefficients become zero after quantization!

Original DCT block

dctblock=

31.0000	51.7034	1.1673	-24.5837	-12.0000	-25.7508	11.9640	23.2873
113.5766	6.9743	-13.9045	43.2054	-6.0959	35.5931	-13.3692	-13.0005
195.5804	10.1395	-8.6657	-2.9380	-28.9833	-7.9396	0.8750	9.5585
35.8733	-24.3038	-15.5776	-20.7924	11.6485	-19.1072	-8.5366	0.5125
40.7500	-20.5573	-13.6629	17.0615	-14.2500	22.3828	-4.8940	-11.3606
7.1918	-13.5722	-7.5971	-11.9452	18.2597	-16.2618	-1.4197	-3.5087
-1.4562	-13.3225	-0.8750	1.3248	10.3817	16.0762	4.4157	1.1041
-6.7720	-2.8384	4.1187	1.1118	10.5527	-2.7348	-3.2327	1.5799

Example: Reconstructed Image

%reconstructed image block

>> qimblock=round(idct2(iqdct))

qimblock=

58	68	85	79	61	68	67	38
45	38	39	33	22	24	19	-2
21	2	-11	-12	-13	-19	-24	-27
-8	-19	-31	-26	-20	-35	-37	-15
-31	-17	-21	-20	-16	-39	-41	0
-33	3	-1	-14	-11	-37	-44	1
-16	32	18	-10	1	-16	-30	8
3	54	30	-6	16	11	-7	23

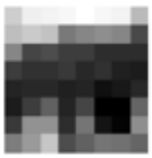
Original image block

imblock=

54	68	71	73	75	73	71	45
47	52	48	14	20	24	20	-8
20	-10	-5	-13	-14	-21	-20	-21
-13	-18	-18	-16	-23	-19	-27	-28
-24	-22	-22	-26	-24	-33	-30	-23
-29	-13	3	-24	-10	-42	-41	5
-16	26	26	-21	12	-31	-40	23
17	30	50	-5	4	12	10	5

Compressed

Compare pixel wise



Original

Temporal redundancy reduction: پیکسل ها بین فریم های متوالی در یک مکان نیز خیلی correlate هستند. در بخش های ثابت تصویر بدون تغییر هستند. بجای انتقال بلوکی از پیکسل ها، مقادیر motion compensatedآنها transform و کوانتایز می شوند. در تبدیل هرچه تعداد ضرایب کمتر باشد تصویر بلوکه تر می شود. هر چه بیشتر باشد بهتر هست. تفاوت بین فریم ها

برای communication مهم است نه streaming. بطور خلاصه: ۱-در داخل فریم quantizer ۲- بین فریم motion compensation ۳- Entropy coding

جلسه ۱۳۹۶/۸/۱

Entropy coding: سمبل با تکرار زیاد را با بیت کم کد می کنیم و سمبل با تکرار کم را با بیت زیاد. که این منجر به کد با طول متغیر variable length code (VLR) می‌شود.

$$bits \text{ assgned to symbole i with prob } p_i = \log_2 \frac{1}{p_i} = -\log_2 p_i$$

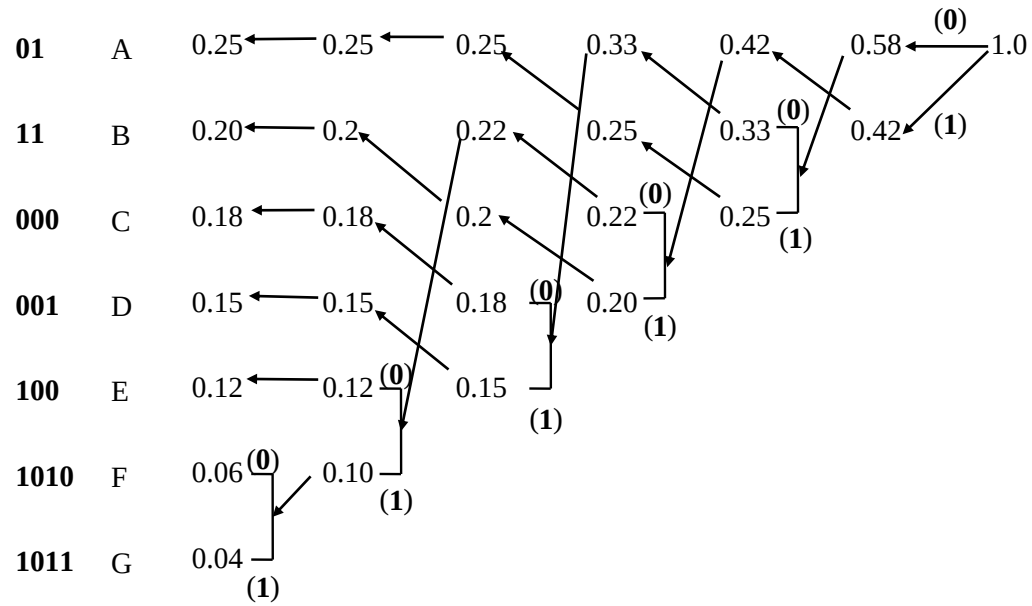
$$\text{Average bit rate} = \quad H(x) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

یک روش تخصیص کد هافمن هست. روش کار به این صورت هست که:

۱- همه سمبل ها را براساس ترتیب احتمال وقوعشان rank گذاری کنید.

۲- بطور متوالی دو سمبل را با کمترین احتمال ادغام کنید تا یک سمبل ترکیبی جدید بدست آورید. سپس براساس ترتیب آنها را مجدد rerank کنید: این یک درخت تولید می‌کند که هر گره، احتمال تمام گره‌های تحتانی آن می‌باشد.

۳- تا هر برگ یک مسیر trace کنید. در هر محل اتصال دو گره اگر از پایین عبور کردید ۱ و اگر از بالا عبور کردید ۰ قرار دهید. مثلا برای حرف A یک مسیر به ریشه وجود دارد که فقط از محل اتصال دو گره می گذرد و در اولی از بالا (ریشه) و دومی از پایین عبور می کند پس کد ۰۱ برای آن بدست می‌آید. یا حرف C تا ریشه از محل سه اتصال می گذرد که هر سه اتصال از بالا عبور می کنند پس کد آن ۰۰۰ می شود. یا G از ۴ اتصال می گذرد و کد آن ۱۰۱۱ می شود.



هافمن بهینه نیست. چون تعداد بیت های اختصاصی صحیح هست و اعشاری ندارد. چگونه به پیکسل عدد کمتر از ۱ بیت نسبت بدهیم؟ بصورت بلوکی اختصاص می‌دهیم به جای تکی. برای مثال فوق با توجه به بیت های بدست آمده هر سمبل و درصد تکرار سمبل میانگین بیت هافمن چقدر است؟

$$0.25\times 2+0.20\times 2+0.18\times 3+0.15\times 3+0.12\times 3+0.06\times 4+0.04\times 4=2.65 \text{ bits}$$

اما انتروپی بیت ها چقدر است؟

$$-\begin{pmatrix}0.25\log_2 0.25+0.2\log_2 0.2+0.18\log_2 0.18+0.15\log_2 0.15+\\0.12\log_2 0.12+0.06\log_2 0.06+0.04\log_2 0.04\end{pmatrix}=2.62 \text{ bits}$$

۲.۶۲ انتروپی سمبل ها هست. که حداقل تعداد بیتی هست که بطور میانگین می شود به سمبل ها اختصاص داد. که هافمن ۰.۰۳ بیشتر از آن است و بنابراین هافمن بهینه نیست. مشکل آن این هست که حداقل بیت اختصاصی ۱ است. که با گروه کردن سمبل ها می توان یک کد اختصاص داد تا بیت اختصاصی کمتر از یک بیت شود.

Arithmetic coding: ایده‌ی پایه کد ریاضی استفاده از یک مقیاس بین ۰ و ۱ برای نمایش سمبل ها می‌باشد. این تابع چگالی احتمال تجمعی همه سمبل هایی است که به ۱ اضافه می‌شوند. بازه و تعداد بیت ها با بزرگ شدن پیام کوچکتر می‌شوند. سمبل های مشابه طول پیام را کمتر از سمبل های نامشابه کاهش می‌دهند.

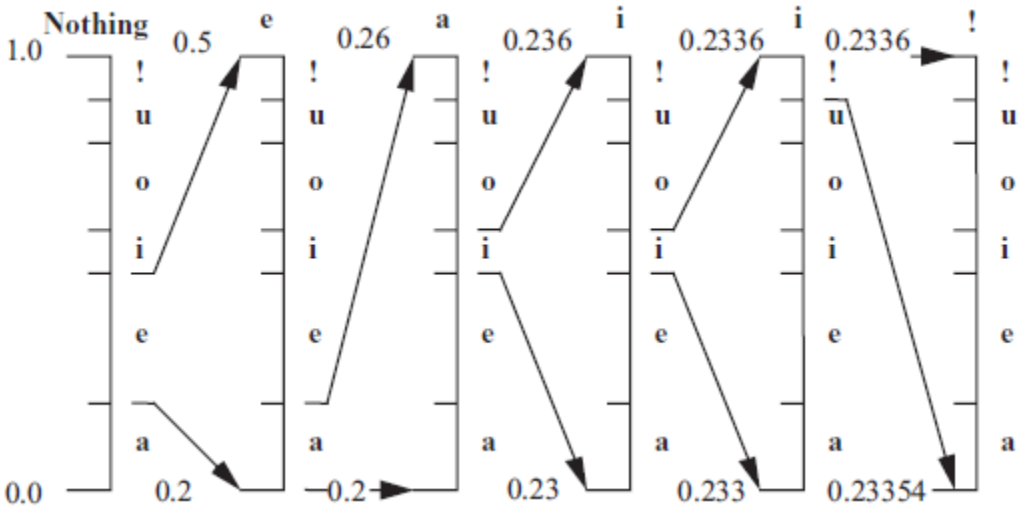
برای توضیح چگونگی کارکرد کد ریاضی از کد ریاضی با مدل ثابت استفاده می‌کنیم. حروف الفبای {a,e,i,o,u,!} را در نظر بگیرید و فرض کنید که احتمالات آنها بصورت جدول زیر باشد:

Symbol	Probability	Range
a	0.2	[0.0, 0.2)
e	0.3	[0.2, 0.5)
i	0.1	[0.5, 0.6)
o	0.2	[0.6, 0.8)
u	0.1	[0.8, 0.9)
!	0.1	[0.9, 1.0)

وقتی احتمال هر سمبل شناخته شد، به هر سمبل یک رنج بین [0,1) اختصاص داده می‌شود. با ارزش‌ترین بخش یک پیام کد شده ریاضی اولین سمبلی است که باید رمزگذاری شود. فرض کنید که پیام eaii! باشد. اولین حرفی که باید کد شود e می‌باشد. چون دامنه e، [0.2,0.5) می باشد کد این پیام نیز در این رنج قرار خواهد گرفت. کاراکتر بعدی که باید رمزگذاری شود a می‌باشد که در دامنه 0-0.2 می‌باشد. چون این اولین حرفی نیست که رمزگذاری می‌شود، این دامنه باید در داخل دامنه [0.2, 0.5) قرار گیرد. چون این محدوده دارای طول 0.3=(0.5-0.2) می‌باشد و حرف a باید در بازه [0, 0.2) در این رنج قرار گیرد و =0.3*(0.2-0) 0.06 بنابراین محدوده آن [0.2,0.26) خواهد بود. حرف بعدی که باید رمزگذاری شود e می‌باشد که باید در محدوده [0.2,0.26) کد شود. چون محدوده i، [0.5,0.6) می‌باشد. فاصله 0.06=(0.26-0.2) هست که 0.03=0.06*0.5 و 0.006=0.06*0.1 و در رنج [0.23, 0.236) قرار خواهد گرفت. به همین ترتیب کد بصورت زیر ساخته خواهد شد.

	New character	Range
Initially		[0, 1)
After seeing a symbol	e	[0.2, 0.5)
	a	[0.2, 0.26)
	i	[0.23, 0.236)
	i	[0.233, 0.2336)
	!	[0.23354, 0.2336)

شکل دیگر روند تبدیل بصورت زیر می باشد.



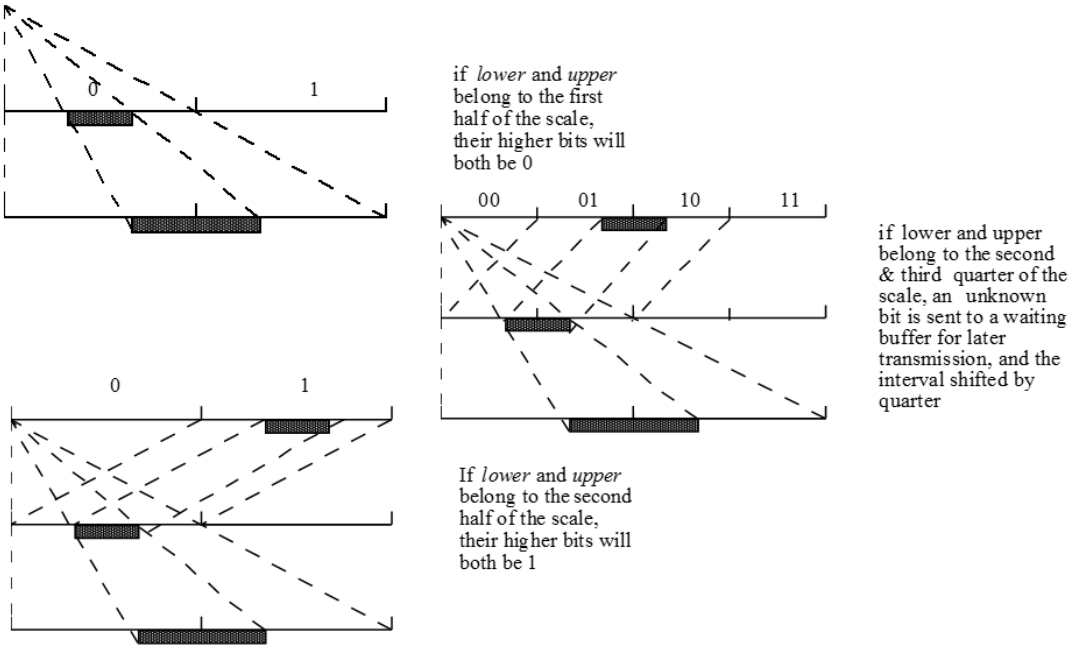
با این شماتیک رمزگذاری، حال می‌خواهیم روش رمزگشایی آن را بررسی کنیم. فرض کنید یک عدد x=0.23355 در دامنه 0.23354 <= x < 0.2336 ارسال شود دیکدر از همین دامنه احتمالات مشابه انکدر استفاده می کند و روال مشابهی را انجام می دهد. با تو جه به اینکه عدد در دامنه [0.2,0.5) می باشد، e از عدد 0.23355 استخراج می‌شود. کد جدید بصورت (0.23355-0.2)/(0.5-0.2)=0.11185 بدست می‌آید. که در محدوده [0, 0.2) قرار می‌گیرد که دامنه a است و a استخراج می‌شود. کد جدید بصورت (0.11185-0.0)/(0.2-0.0)=0.55925 بدست می‌آید که در دامنه [0.5,0.6) قرار دارد و حرف i استخراج می شود و کد جدید (0.55925-0.5)/(0.6-0.5)=0.5925 بدست می‌آید و چون در دامنه [0.5,0.6) هست حرف بعدی i استخراج می شود و کد جدید (0.5925-0.5)/(0.6-0.5)=0.925 بدست می‌آید که در بازه [0.9,1) قرار دارد و برابر با ! یعنی حرف پایان می باشد. در کل بدست آوردن کد از فرمول زیر در دیکدر تبعیت می‌کند.

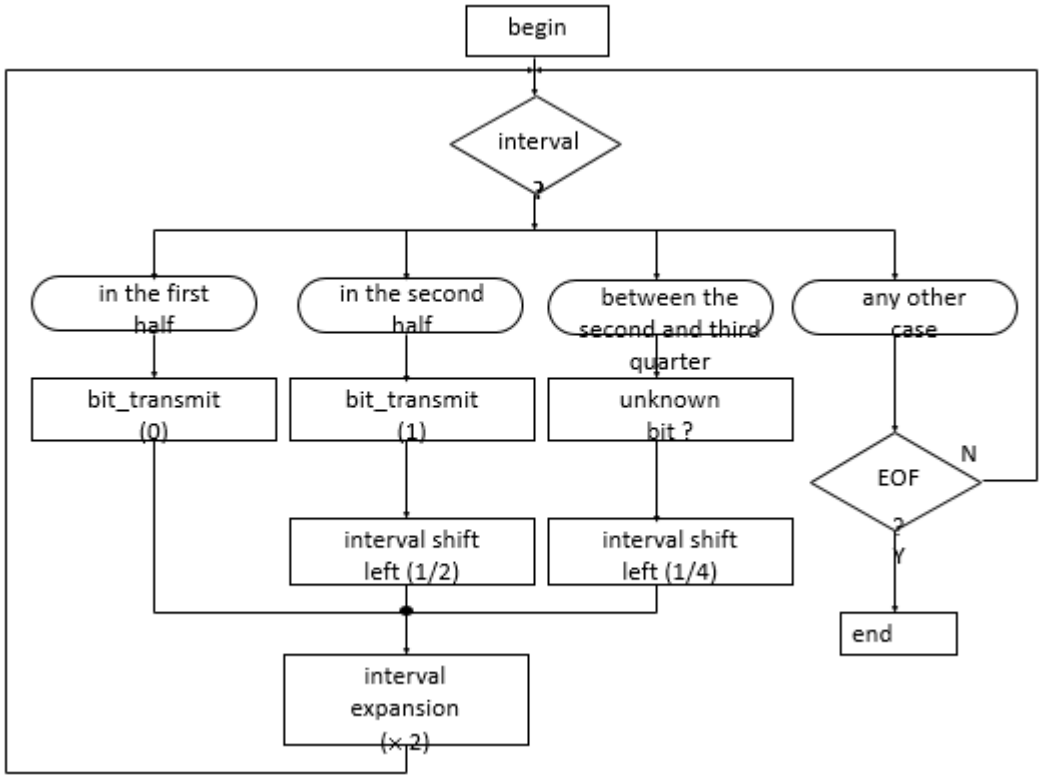
$$R_{n+1} = \frac{R_n - L_n}{U_n - L_n}$$

بزرگترین اشکال این هست که عدد دریافتی در دیکدر دارای خطا باشد. این خطا روی تمام پیام تاثیر خواهد داشت. در هافمن هم به خاطر اینکه کد دارای طول متغیر است در صورت خطا در داده دریافتی روی کل داده منتشر می‌شود. ولی اگر طول متغیر نبود مثل PCM که ۸ بیتی باشد با یک بیت خطا فقط یکی از ۸ بیتی ها خراب می‌شود. دشمن اصلی single bit error هست و از اصول مخابرات هست که یک خطا را correct می کنیم با بیت های parity.

Binary arithmetic coding: در بخش قبل دیدید که وقتی تعداد سمبل ها زیاد می‌شود رنج پیام کوچکتر می‌شود. اگر ما به کدینگ سمبل‌های بیشتری ادامه دهیم ممکن است رنج نهایی از محدوده دقت هر کامپیوتری کمتر شود. برای حل و فصل این مساله می توان با کدینگ باینری کار کرد.

Binary Arithmetic Coding

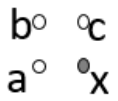




نکات: ۱- entropy coding چه هافمن یا کدینگ ریاضی ایجاد نرخ متغیر می‌کند بنابراین هرکدام از بیت ها به خطا برود دیکدر به خطا می رود و دیکدر باید ریست شود. ۲- در هافمن اگر احتمالات بالا باشد ما هنوز یک بیت می فرستیم. اما در باینری اگر احتمال بیشتر از 0.5 شود هیچی نمی‌فرستیم. پس از هافمن بازدهی بیشتری دارد.

سوال آیا می شود entropy coding را هنوز هم بهبود داد. که به آن context adaptive می‌گویند. که احتمال سمبل ها را براساس محتوا تغییر دهیم.

Example:
Context based arithmetic coding

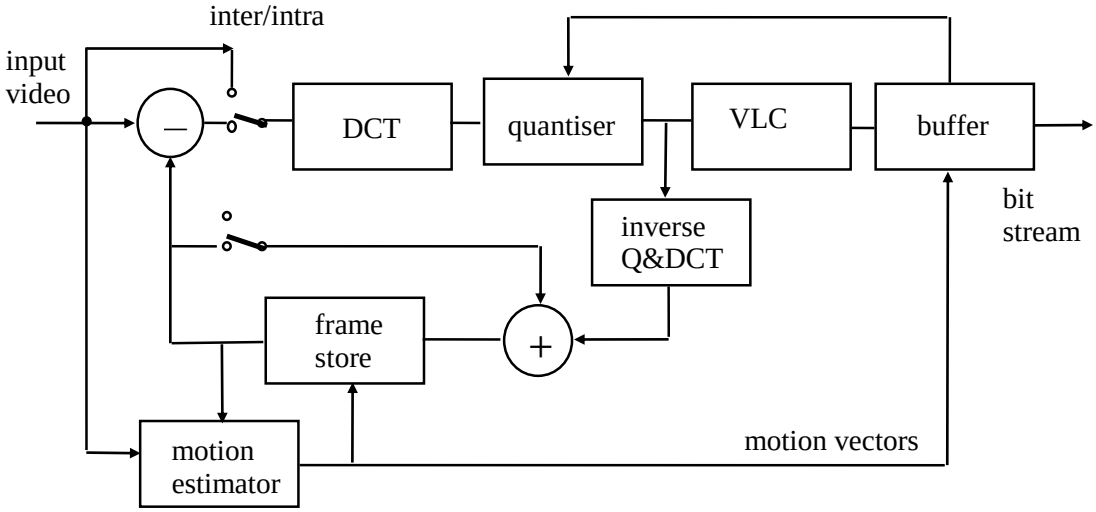


$$Context = 2^2 c + 2^1 b + 2^0 a = 4c + 2b + a$$

موقعیکه در محاسباتمان در کدینگ خطا می کنیم، باید پنالتی بدهیم. که نیاز به مصرف بیت زیادی هست.

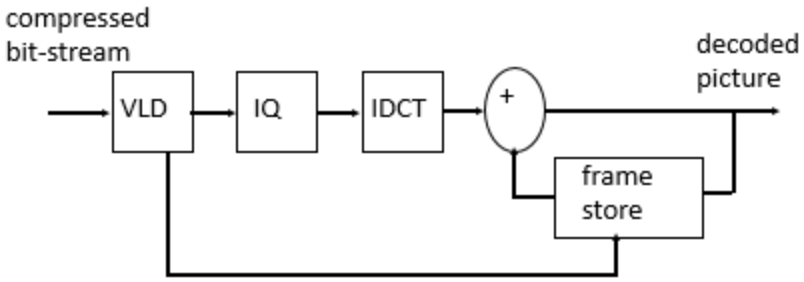
A generic video codec: یک کدک عمومی چه چیز هایی دارد.؟

۱- یک انکدر ویدیو سه مفهوم فشرده سازی را استخراج می کند (spatial, temporal, entropy)
۲- برای کاهش تاخیر انکدینگ DPCM استفاده می‌شود.
۳- Motion compensation برای کاهش تفاوت زمانی پیکسل ها استفاده می شود.



یک بلاک لومینانس می گیریم.frame store یک تصویر قبلی را داخلش دارد. تفاوت فریم را با فریم ذخیره شده را می گیریم و DCT می‌گیریم. سپس کوانتایز می‌کنیم. سپس VLC(variable length code) می‌گیریم و ارسال می‌کنیم. Distortion های ناشی از تفاوت را باید از بین ببریم که با inverse Q&DCT انجام شده و با جمع با فریم اصلی خرابی اصلاح می شود. بافر برای نگهداری داده‌ها استفاده می شود. اگر خروجی تغییر سرعت داشته باشد بافر انها را تنظیم می‌کند. اگر ورودی بافر خیلی بیشتر از خروجی بافر باشد در اینصورت quantizer step size را می بریم بالا تا بیت کاهش یابد و اگر سرعت ورودی بافر کمتر از خروجی شود quantizer step size را کم می کنیم و بیت را بالا می بریم.

Generic Decoder

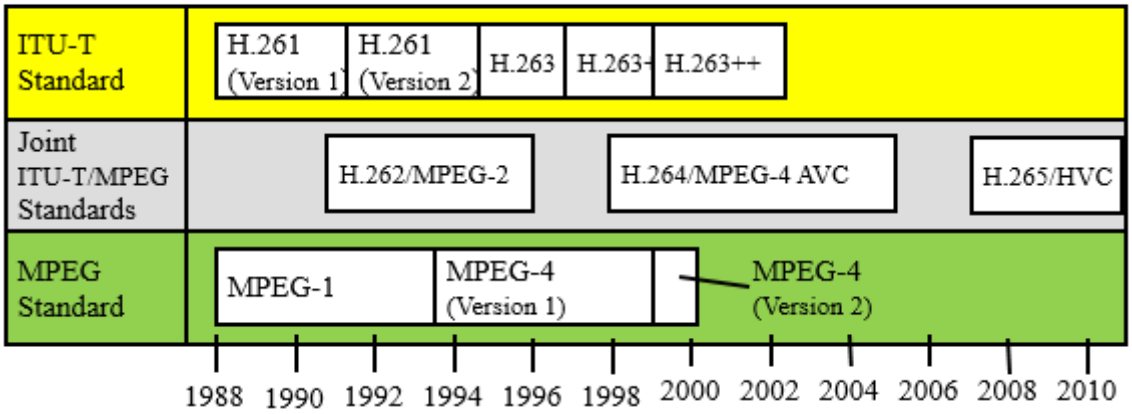


Main problem:

Propagation of channel errors

جلسه ۱۳۹۶/۸/۱۳

Standard codecs



برای فشرده سازی ۱- از spatial correlation استفاده می کنیم و دیدیم که بهترین transformation دوبعدی (DCT) هست چون پیکسل ها هم بصورت افقی و عمودی با هم correlated داریم. البته همزمان ۲- بعد زمانی Temporal redundancy reduction هم داریم که بدرد ما نمی خورد که نیاز به استفاده از فریم های مختلف و بنابراین ذخیره فریم دارد و ذخیره هر فریم 40ms می باشد که به درد 2-way communication نمی‌خورد. چون delay ایجاد می کند. به جاش تفاوت فریم ها را می گیریم اگر بتوانیم جبران حرکت بکنیم.(motion estimation) بنابراین حرکت را هم بدست می آریم و تفاوت می گیریم. به جای کار با پیکسل از گروهی از پیکسل ها استفاده می کنیم. که سربار را کم می کند (سربار آدرس دهی). Quantizer باعث فشرده سازی می‌شود و ۳- از entropy coding یعنی VLC استفاده می کنیم تا بیت های کمتر را برای سمبل های با فرکانس تکرار بیشتر بدهیم. بعضی موقع ها هم لازم هست که خود فریم را کد کنیم نه تفاوت آن را. این باعث خواهد شد جلو انتشار خطا را بگیرد.

دلایل ورودی با نرخ متغیر به بافر: ۱- VLC باعث ایجاد بیت متغیر می شود. ۲- تفاوت گرفتن گاهی ممکن هست هیچ خروجی نداشته باشد مثلا وقتی تغییری بین دو فریم نداریم. ۳- کوانتایزر با تغییر دامنه بیت متغیر تولید می کند. ۴- کد کردن inter/intra متغیر هست. برای تنظیم خروجی بافر مورد نیاز هست یعنی ورودی بافر با rate متغیر هست ولی خروجی آن را smooth می کند که به آن smooth buffer می گویند. بدی بافر بالا بردن end to end delay می باشد. که خروجی روی دو نوع کانال قرار می گیرد ۱- fix bit rate ۲-variable bit rate. (circuit switch or packet switch).

بافر باید اینقدر بزرگ باشد که سرریز نکند مثلا در H.264 تا ۴ ثانیه فریم یعنی ۱۰۰ فریم را ذخیره می‌کند. تنظیم سرعت ورودی را می توان با تنظیم quantizer step size یا دستور قطع ارسال دیتا انجام داد. چرا تصویر در گیرنده freeze می شود؟ وقتی که یک فریم چند بار در گیرنده repeat می‌شود.

گروه های استاندارد:

۱- ITU-T: در مخابرات دوطرفه اولین مساله end-to-end delay است. H مخابرات هست

۲- Mpeg: اصل بر broadcast هست (یک طرفه هست). پردازش و فشرده سازی خیلی مهم هست.

همکاری بین MPEG و ITU می شود Joint MPEG/ITU

JM نمونه آزمایشگاهی H.264 می‌باشد. اولین کدک مخابراتی H.261 بود. P time 64➔p x 64

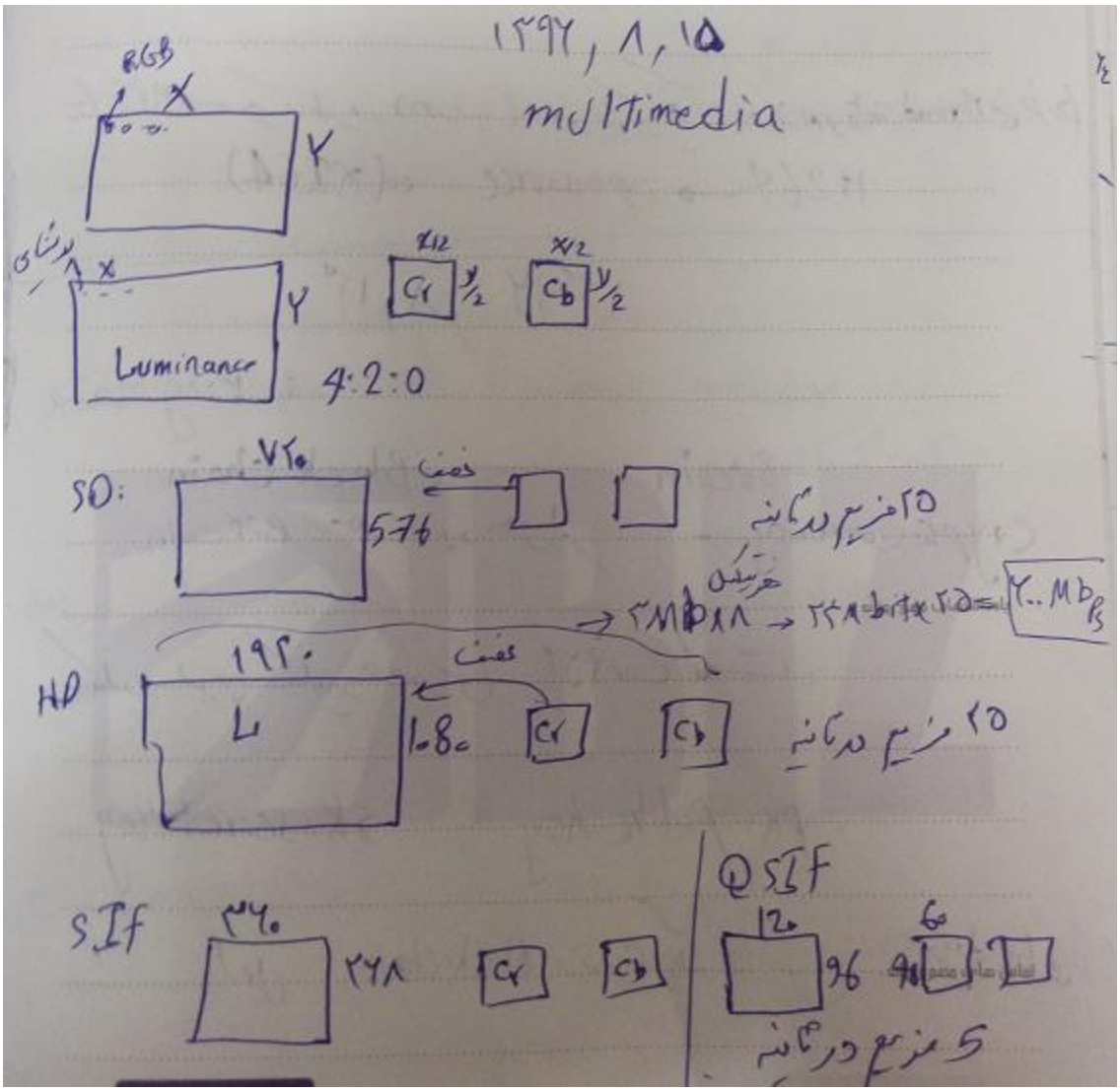
تفاوت بین کدک های مختلف بر اساس پارامترهای زیر است:

All standard codecs follow the generic interframe codec of: DCT/DPCM/MC/VLC. Their main differences lie on the way these elements are employed:

- Block transform length and type
- Block size for Motion estimation and its precision
- Methods of VLC
- Quantisation
- Coding of quantised transform coefficients
- Addressing of data
- Preventing error propagation
- Various types of coding each frame

جلسه ۱۳۹۶/۸/۱۵

اولین قدمی که ما با فشرده سازی انجام می‌دهیم بازی با ابعاد تصویر می‌باشد. لومینانس همیشه برابر با ابعاد تصویر هست ولی وابسته به فرمت رنگ ابعاد Cr,Cb روی x,y نصف می شود. بنابراین برای ارسال بیت کمتر یکی از روش های ساده کوچک کردن ابعاد تصویر می‌باشد. بعد روی تصویر انتخابی فشرده سازی انجام می‌شود.

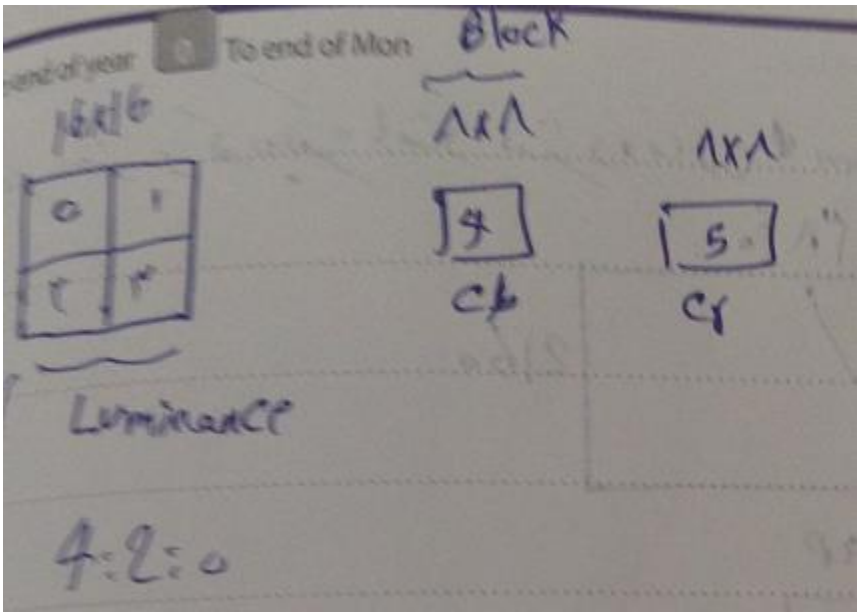

۱۳۹۲, ۸, ۱۵
multimedia
RGB
Luminance
4:2:0
SD:
HD
SIf
QSIf
Ultra HD
map
96
16

در آمریکا ۳۰ فریم در ثانیه ارسال می‌شود و به جای آن تعداد خطوط فریم کم می‌شود. 720*576→720*480

این در ارسال یک طرفه دچار مشکل نمی‌کند ولی در ویدیو کنفرانس دچار ایراد می‌کند. (2-way communication).

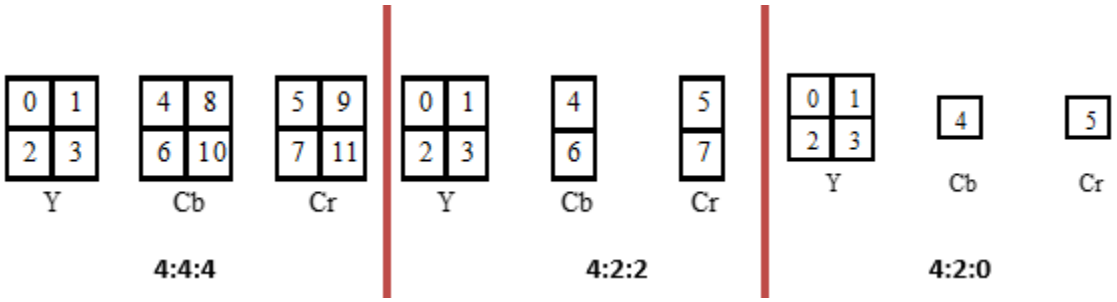
CIF: 360*288*30

در ارتباط دوطرفه باید فرمت های اروپایی و US به هم تبدیل شوند که باعث deform شدن تصویر خواهد شد. ۹۹٪ تلویزیون با فرمت 4:2:0 استفاده می‌شود. در فرمت 4:2:0 فرمت ماکروبلاک بصورت زیر هست که ۶ فریم ۴*۴ برای Y,Cr,Cb استفاده می‌شود. یک ماکرو بلاک ۱۶*۱۶ است و بلاک ۸*۸ است.



برای تخمین motion از ماکروبلاک استفاده می‌شود پیکسل برای luminance هست هر دو L یک chrominance دارد بنابراین اگر در L حرکت ۴ پیکسل باشد در Cr, Cb حرکت ۲ پیکسل هست. تصمیمات براساس L هست زیرا در یک ویدئو ۸۵٪ اطلاعات در L می‌باشد، texture در L است و حرکت در L است. هرچه به L اعمال شود باید به Cr,Cb اعمال شود. ۲ پیکسل در L برابر با یک پیکسل در C است. یعنی جابجایی ۲ پیکسل در L برابر یک پیکسل در C است. در generic video codec به ماکروبلاک کار داریم نه به ابعاد تصویر (HD, SD, CIF, SIF فرق اینها در تعداد ماکروبلاک آنها می‌باشد). ورودی کدک ماکرو بلاک ۱۶*۱۶ می‌باشد motion vector با مقایسه با frame store بدست می‌آید برای لومینانس برابر با MV جابجا می کنیم و برای Cr Cb نصف آن. DCT، quantize، VLC روی 8*8 انجام می شود. Inverse Q&DCT برای حذف distortion است. به همین خاطر به آن Local decoder می‌گویند. فیدبک بافر به quantizer برای تغییر step size هست تا بایت کمتری generate شود. کم باعث تغییر کیفیت ویدیو می‌شود به همین دلیل بافر را خیلی بزرگ می‌گیریم که تغییرات خیلی بزرگ نباشد. البته می توان چند فریم را اصلا کد نکرد که در گیرنده یک فریم فریز می‌شود و کیفیت پایین می‌آید.ورودی بیت بافر تغییر می کند که وابسته به محتوا است و خود وابسته به دو عامل texture, motion می باشد. هرچه تصویر texture داشته باشد بیت بالا می رود هرچه حرکت سریعتر باشد بیت بالا می‌رود خود intra کد کردن هم بیت را بالا می‌برد. برای constant bit rate حتما باید بافر داشته باشیم. برای packet network هم داشته باشیم بهتر هست چون برای روتر هم burst of packet خوب نیست. بهتر هست

سرعت دریافت بسته ها به یکباره تغییر نکند که به آن smoothing buffer می‌گویند. ما می توانیم بافر نگذاریم و با استفاده از virtual buffer نیز step size را کنترل می‌کنند. ترکیب motion vector و macro block وابسته به نوع کانال انتقال است. در circuit switch با هم مالتی پلکس می‌شوند در packet می‌توانند جدا باشند. (بافر خیلی بزرگ= افزایش end to end). در audio, video خطرناک ترین چیز overhead است.



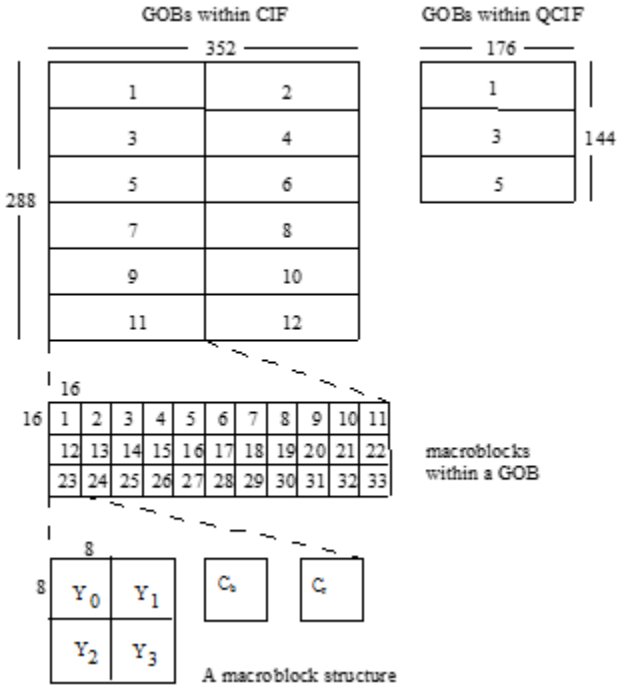
Bit rate variation

- Constant Bit Rate (CBR)
  - Quantiser step size and even frame rate may change to adapt the bit rate to channel rate
  - Video quality is variable
  - Normally a complex structure is used to regulate the bit rate
- Variable Bit Rate (VBR)
  - Quantiser step size is nearly constant, generating almost constant quality picture
  - Difficult to adapt to channel rate, but is suitable for packet Switched Network applications (e.g. Internet)
  - No need for bit rate regulation (coded is simple)

H.261

- The first hybrid/DCT/DPCM codec is H.261
- Designed for video conferencing at px64 Kbit/sec, using Circuit Switched telephone network
- All pictures are interframe coded (difference between successive frames)
- Problems:
  - Error propagation
  - In case of errors, it needs updating

H.261 اولین کدکی هست که ترکیب DCT, DPCM هست که DCT تفاوت بین ضرایب coefficient ها هست و DPCM تفاوت بین فریم‌ها را از بین می‌برد سرعت آن p64x Kbps می‌باشد. H.261 برای کنفرانس ویدیویی بود و برای circuit switch استفاده می‌شد. همه فریم‌ها interframe کد می‌شوند (یعنی تفاوت بین فریم‌های متوالی). فقط زمانی که کدک بفهمد کد کردن اینترا بهتر هست آن را اینترا کد می کند و بیش از ۹۵ درصد فریم ها اینترفریم هستند. مثلا زمانی که texture عوض شود کد کردن اینترا یعنی خود فریم ممکن است بهتر از اینترفریم باشد. مشکل آن انتشار خطا بود و در صورت بروز خطا نیاز به بروزرسانی داشت. که اینجا دو مساله وجود دارد: ۱- جلو انتشار خطار را برای بعدی ها بگیریم ۲- خطای موجود را تصحیح کنیم. روش جلوگیری از انتشار خطا چیست؟



یکی از این راهها این هست که تصویر را به چند قسمت تقسیم کنیم. که تصویر به ۱۲ قسمت تقسیم شده است و به هر کدام از آنها GOB(Group of macroblocks) می‌گویند. دلیل؟ اگر errorی بوجود آمد می‌توانیم در شروع این GOBها سیستم را ریست کنیم. یعنی هر GOB ۳۳ ماکروبلاک دارد. بعد از کد کردن GOBی یک، GOBی دو کد می‌شود. MB کوچکتریم واحد کدینگ ویدیو می‌باشد.

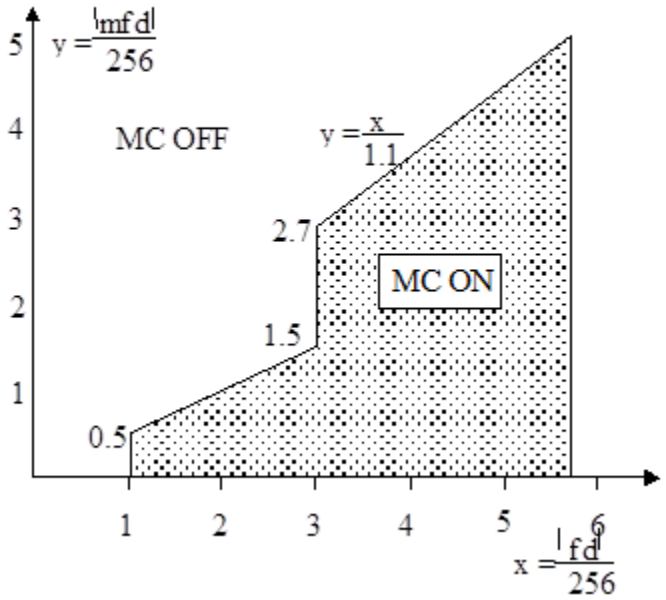
جلسه ۱۳۹۶/۸/۲۰

افزایش تعداد ریست سربار را بالا می برد چون هر ریست یک بیت پترن خاصی هست که در بعضی کدکها تا ۵۰ بیت می شود. تصمیمات با luminance گرفته می‌شود چون بیشتر انرژی در اینجا هست همچنین texture و روشنایی. چون transformation ۸*۸ هست، و بلاک ۱۶*۱۶ هست این عمل برای ماکرو بلاک ۴ بار برای Y0,Y1,Y2,Y3 و دو بار برای Cb,Cr تکرار می کنیم. برای جلوگیری از خطای در زمان intra کد می کنیم (بین فریم ها). با تقسیم تصویر به چند قسمت اجازه نمی دهیم error در خود فریم پخش شود. کوانتایز خودش باعث distortion می شود به همین خاطر باید تصحیح شوند تا از انتشار distortion در ادامه جلوگیری شود. برای کد شدن باید تصمیم بگیریم که آیا خود فریم کد شود؟ یا تفاوت آن؟ در جدید هر دو

انجام می‌شود بین آنها هر کدام بهتر باشد و بیت کمتری داشته باشد انتخاب می‌شود که به processing power ربط دارد. انرژی یعنی standard deviation.

Motion compensate همیشه انرژیش کمتر است. Frame difference: تفاوت بلاک با همان بلاک در فریم قبلی. از نظر فرستادن همیشه بدون motion بهتر است اما با motion انرژی بیشتری دارد. Motion estimation خیلی گران هست و ۶۰ درصد توان پردازش خرج آن می‌شود. همه بلاک ها motion compensate نمی شوند.

Motion compensation decision

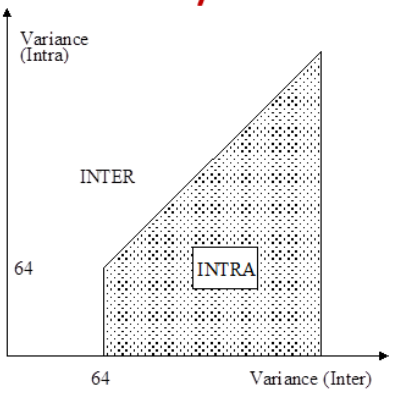


Not all blocks are motion compensated

The one which generates less bits are preferred.

قبل از اینکه کد کنیم تصمیم بگیریم کدام کار را انجام دهیم. اما در کدکهای جدید هر دو انجام می شود و بهتر انتخاب می‌شود. بطور کلی اگر سیستمی پیچیده باشد دو راه داریم که با پیچیدگی مبارزه کنیم. ۱- پردازش موازی ۲- ساده سازی. یا ترکیب هر دو را استفاده کنیم. یک بلاک ۱۶*۱۶ را هم MC می کنیم هم نمی کنیم و انرژی آنها (واریانس) را مقایسه می کنیم و کمتر را انتخاب می کنیم.

Inter/Intra decision



Not all blocks are interframe coded

Intracoded blocks can reset errors

At least 3 MB in each frame is intracoded (forced updating)

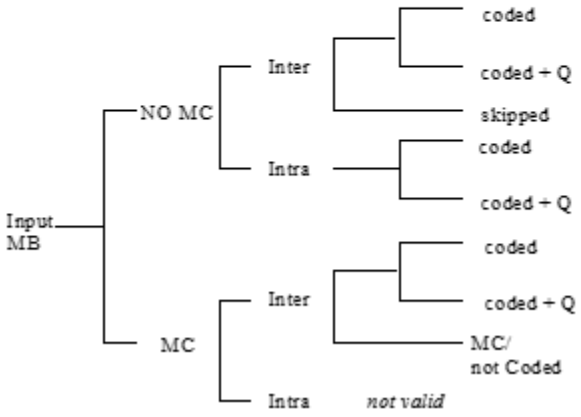
شاید intra کد کردن خود تصویر بهتر باشد (هنگام تغییر زیاد). Intra بیت اگر کم باشد بهتر است. همچنین رفرش می شود و کمک به جلوگیری از انتشار خطا می‌کند.

Other types of macroblocks

- *Inter coded*: interframe coded macroblocks with no motion vector or with a zero motion vector.
- *MC coded*: motion compensated MB, where the MC-error is significant and needs to be DCT coded.
- *MC not coded*: these are motion compensated error MBs, where the motion compensated error is insignificant. Hence there is no need to be DCT coded.
- *Intra*: intraframe coded macroblocks.
- *Skipped*: if all the six blocks in a macroblock, without motion compensation have an insignificant energy they are not coded. These MBs are sometimes called skipped, not-coded or fixed MBs. These types of MBs normally occur at the static parts of the image sequence. Fixed MBs are therefore not transmitted and at the decoder they are copied from the previous frame.

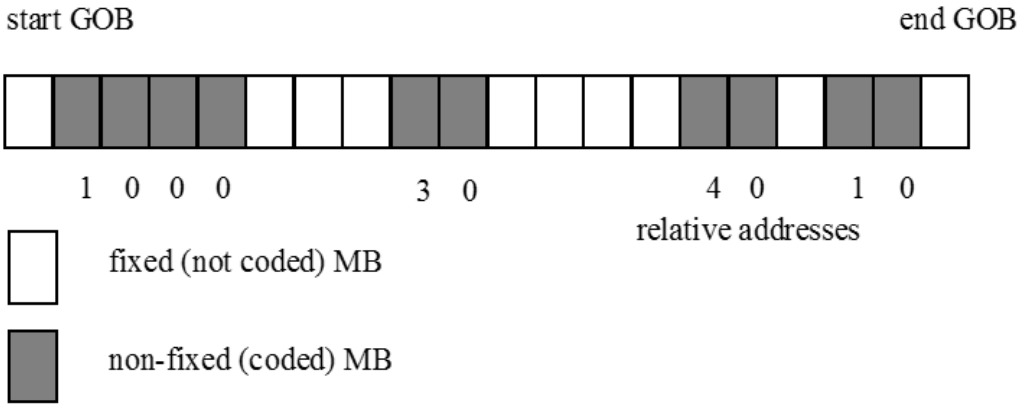
MB Types

- Since the quantiser step sizes are determined at the beginning of each GOB or row of GOBs, they have to be transmitted to the receiver. Hence the first MBs have to be identified with a new quantiser parameter. Therefore we can have some new macroblock types. In general they are:



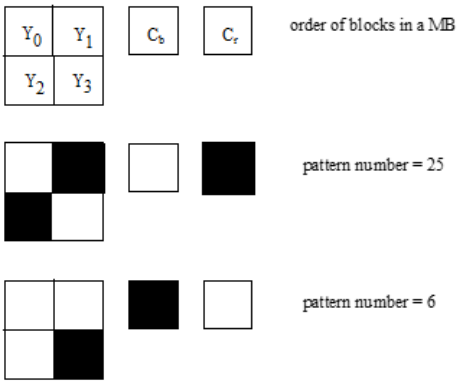
آدرس دهی ماکرو بلاک ها:

Addressing of macroblocks



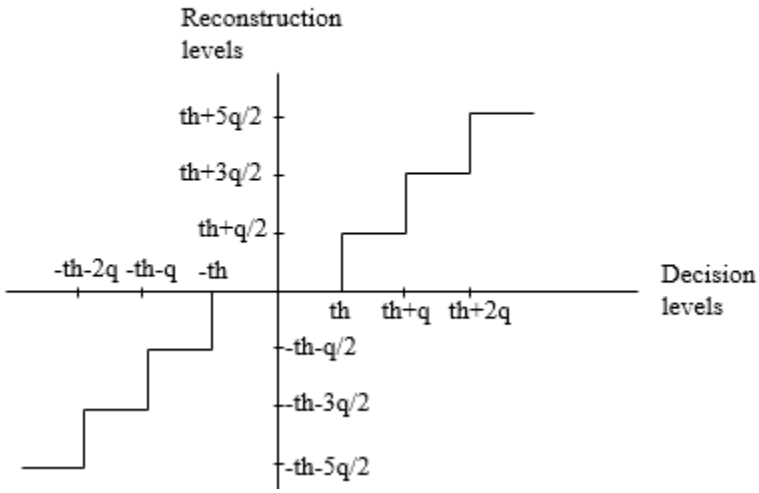
Addressing of blocks

$$attern\_number = 32Y_0 + 16Y_1 + 8Y_2 + 4Y_3 + 2C_b + C_r$$



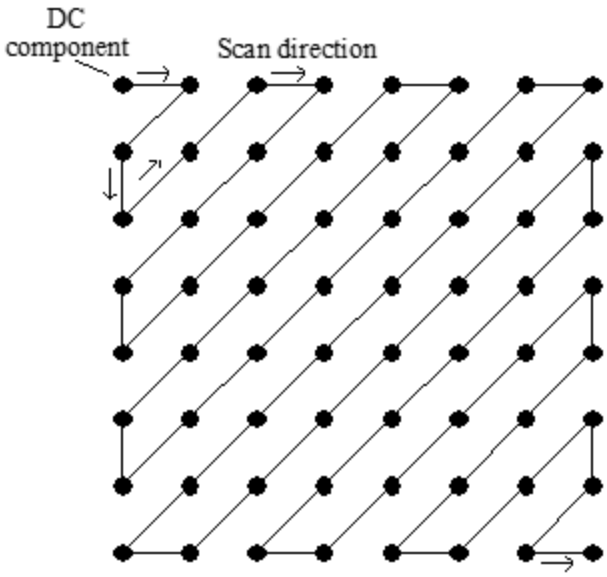
3 0 یعنی ۳تا ماکروبلاک قبلی کد نشده اند. این برای circuit switch است.

Quantisation and Coding



Two-dimensional variable length coding

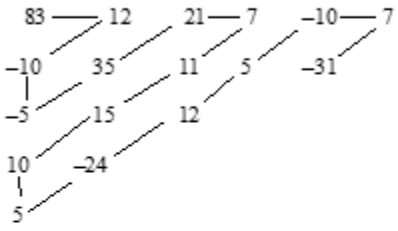
Zigzag scanning of 8×8 transform coefficients



انتظار داریم بیشتر مسیر ضرایب بالا باشند.

Why Zig-zag scanning?

Example
Th=q=16



Raw coefficients	83	12	–10	–5	35	21	7	11	15	10	5	–24	12	5	–10	7	–31
New coefficients	83	0	)	0	35	21	0	0	0	0	0	–24	0	0	0	)	–31
Quantised values	88	0	)	0	40	24	0	0	0	0	0	–24	0	0	0	)	–24
Index	5	0	)	0	2	1	0	0	0	0	0	–1	0	0	0	)	–1

Events to be transmitted: (run, index) (0,5) (3,2) (0,1) (5,–1) (4,–1)

فرض می کنیم th=16 و q=16. اعداد محدوده th, th– باشند صفر می‌شوند. عدد ۸۳ تقسیم بر ۱۶ می شود ۵ که f(Q)=int(83/16)+8=88. به جای ارسال ۸۸ کوانتایز آن یعنی 88/16=5 ارسال می شود. و برای حذف صفرها ایندکس دوبعدی

ارسال می‌شود. (0,5) یعنی ۵ و قبل از آن صفر نداریم. و (3,2) یعنی عدد ۲ و ۳ تا صفر قبل از آن. و (0,1) عدد یک و صفر عدد قبل از آن. (5,-1) یعنی 1- و ۵ عدد صفر قبل از آن. (4,-1) یعنی -۱ و ۴ صفر قبل از آن. (0,5) بصورت سمبل کد می‌شود نه اینکه ۰و ۵ کد شوند. در فرستنده و گیرنده سمبل‌ها شناخته شده هستند. احتمال (0,5) بیشتر از (5,-1) است. Zig-zag طوری انتخاب می‌شود که ۱- زودتر به عدد بزرگتر برسیم (اول اسکن شوند) ۲- صفرها بمانند در آخر. (بسیار بسیار مهم). (run-length). به جای ۱۷ سمبل ۵ سمبل می‌فرستیم. هدف زیگ زگ کم کردن تعداد سیمبل ها هست.

در اینجا می‌توان water mark کرد. بهتر هست به صفرها دست نزنیم تعداد eventها زیاد می‌شوند. هر چه تعداد صفرها بالا برود بیت بیشتری می‌خواهد.

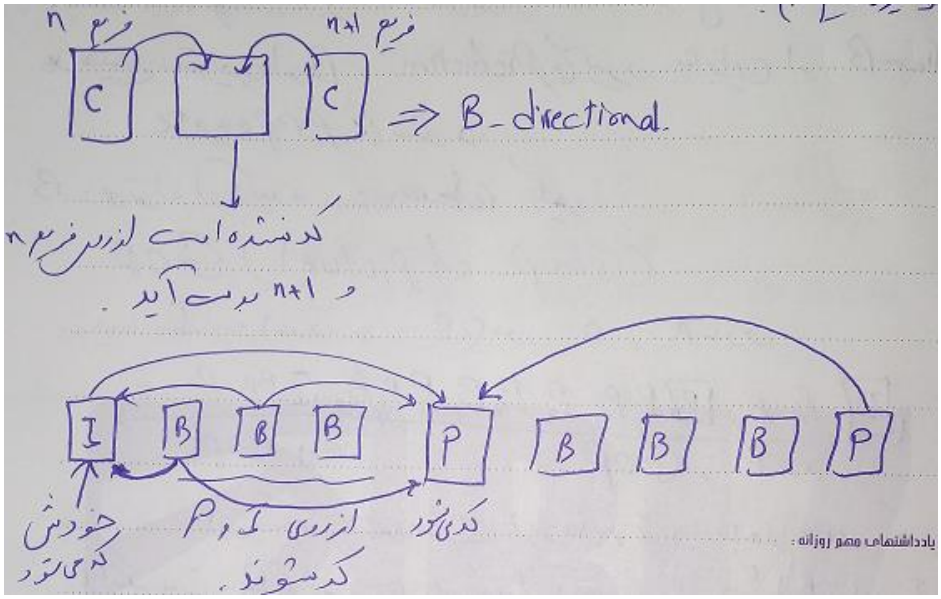
جلسه ۱۳۹۶/۸/۲۲

Why zig zag scanning?

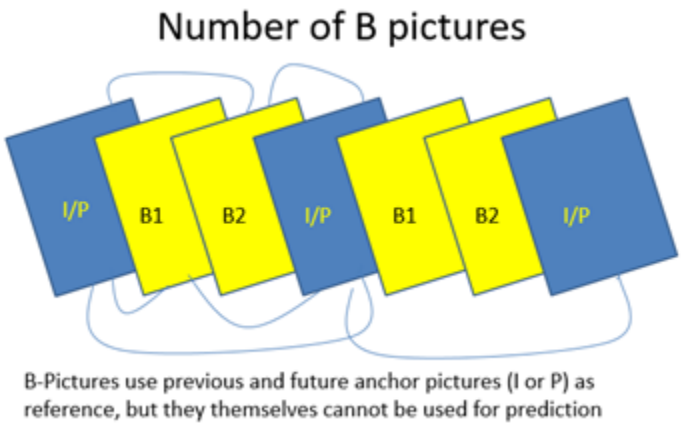
سیمبل یک قرارداد هست بین دیکدر و انکدر و می تواند چند بعدی تعیین شود. بعد از ۳۱- (در شکل فوق) یک کد دوبیتی (۰۱) فرستاده می‌شود به عنوان end of block. که البته با ایندکس دو بعدی می شود تعداد سیمبل ها را کاهش داد. ۹۰٪ حجم دیتا مربوط به ضرایب هست و ۱۰ درصد راجع به آدرس‌ها و ... است. بعد از اینکه سیمبل ها تمام شد یک کد به عنوان end of block ارسال می‌شود یعنی بقیه صفر هست.

MPEG: ایراد روش قبل: یک فیلم اگر ضبط شود نمی توان آن را از وسط فیلم دید. (h.261 از اول تا آخر باید دید). هدف: روش قبل اصلاح شود که مناسب باشد برای ذخیره سازی یعنی بعضی از فریم‌ها را intra کد کنیم تا بتوانیم از هر جایی ویدیو ذخیره شده را ببینیم. هرچه تعداد intra زیاد باشد بیت زیاد می‌شود ولی دسترسی را بهتر می‌کند. اگر همه I باشند می‌توان در هر جا edit کرد (کیفیت بالا)

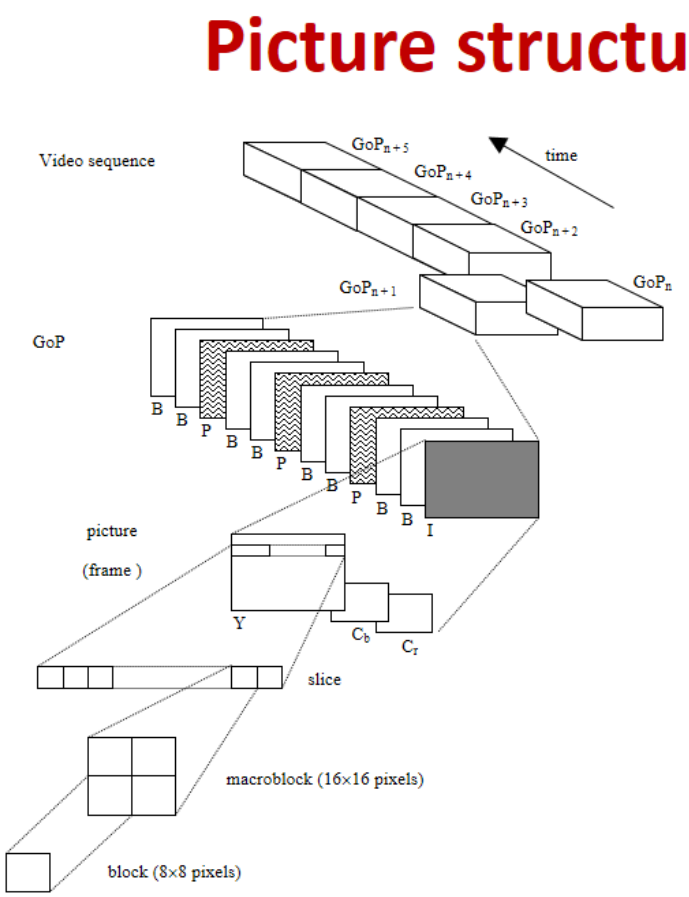
انواع تصویر: ۱- همه intra کد شوند (I) که بیت را بالا می‌برد. ۲- P: نسبت به گذشته کد شوند (predictive) مثل H.261
۳- B: نسبت به گذشته و آینده کد شوند.



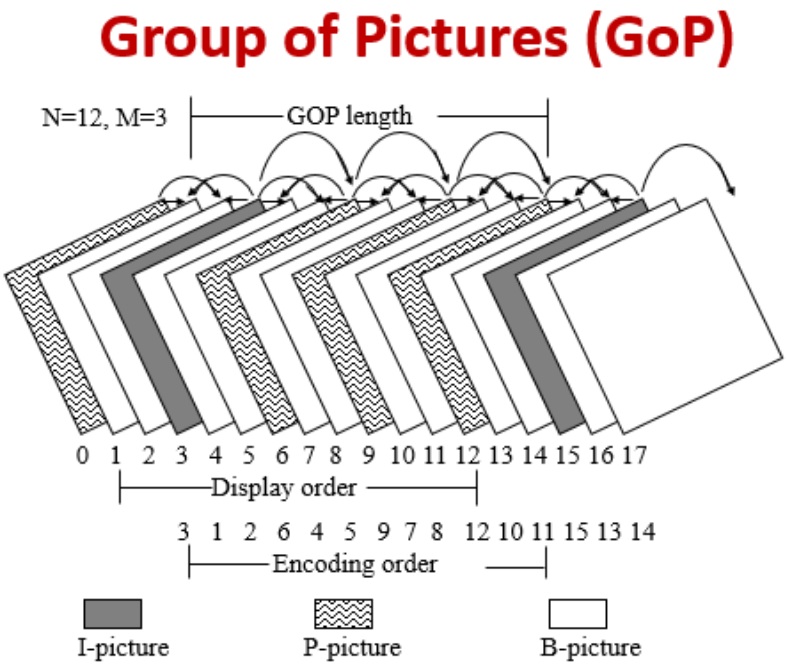
تعداد B ها اگر زیاد باشد فاصله بین I,P زیاد می‌شود و P بیت زیادی خواهد خواست و از یک حدی زیاد شود کمکی نخواهد کرد. معمولاً تعداد B ها ۱، ۲، ۳ می‌باشد. *در سیستم اروپایی هر ۱۲ فریم یک I می‌باشد. (۲۵ فریم، تقریباً نیم ثانیه) در سیستم امریکایی هر ۱۵ فریم یک I است (۳۰ فریم، نیم ثانیه).* هر چه تعداد Iبیشتر شود بیت بیشتر می شود ولی از انتشار خطا جلوگیری می‌کند باید بین network congestion و تعداد Iها یک بهینه سازی داشته باشیم. هیچ تصویری از B، prediction نمی‌گیرد. بنابراین از B خطا منتشر نمی‌شود. B می‌تواند از آینده Reference بگیرد.



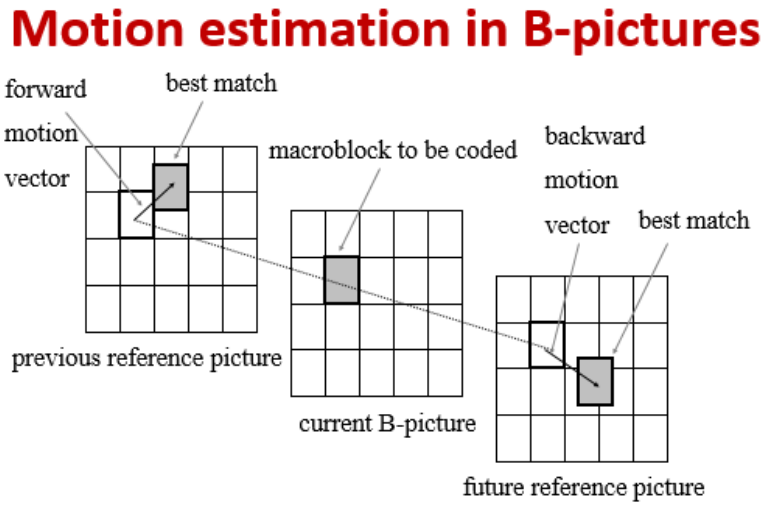
GOP: (group of pictures) ؛ I، یک عدد، P سه عدد، و B ۸ عدد. شامل ۱۲ عدد تصویر است.



یک GOP بصورت IB1B2P1B3B4P2B5B6P3B7B8 می‌باشد. در شبکه در صورت ازدحام می‌شود B ها را حذف کرد در حقیقت B قربانی I,P می‌شود. نکته اینکه در بسته های اینترنت چگونه تشخیص دهیم نوع بسته B است. ترتیب ارسال IP1B1B2P2B3B4P3B5B6. چون B1,B2 از روی I1,P1، predict می‌گیرند. بنابراین یکی از بدی های داشتن B این هست که end-to –end delay زیاد می‌شود. هم در انکدر هم در دیکدر دارای تاخیر هستیم. B7,B8 می‌توانند از P و فریم I در GOP بعدی predict بگیرند.



MPEG1 بیشتر برای ذخیره سازی طراحی شد ولی برای ارسال به مشکل می خورد. برای motion در فریم‌های B میانگین motion بین آینده و گذشته را می گیرند. (past +future)½. که خود این باعث کم شدن اثر خطا در B فریم می شود. یعنی اگر در یکی از فریم های مرجع خطا داشته باشیم نصف خطا در prediction تاثیر می‌گذارد و با تکرار این روند خطا به سمت صفر می‌رود. در شکل فوق N فاصله بین Iها و m فاصله بین anchorها می‌باشد.



نرخ بیت: B-pictureها با بازدهی بیشتری نسبت به P کد می‌شوند براساس: texture, motionو تعداد B-pictureها. معمولا 2 عدد B بین او P وجود دارند. که نرخ بیت آنها 5:2:1 هست یعنی 5unit اگر I باشد 2Unit نوع P و 1Unit نوع B است.

Slice=the same as Group of Blocks inH.261

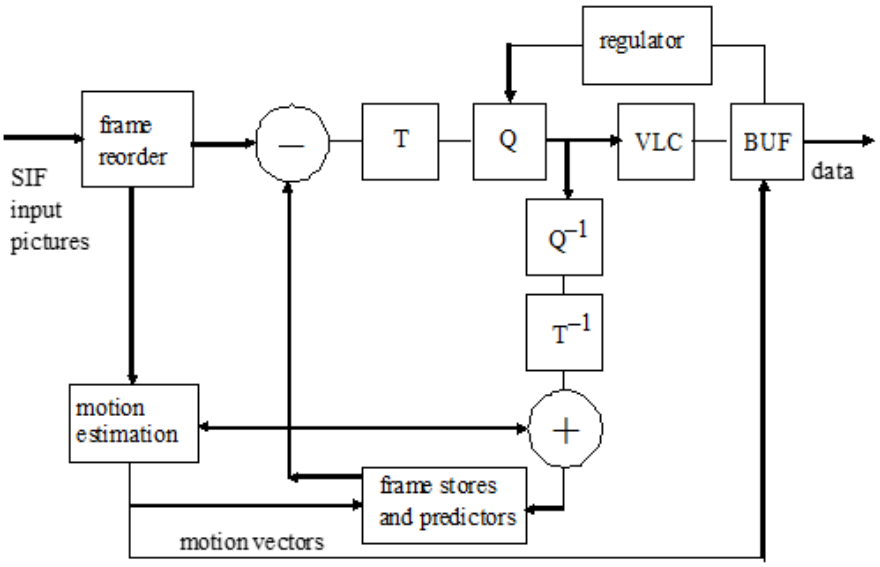
1 begin	end 1
2 begin	end 2
3 begin	end 3
4 begin	end 4
5 begin	end 5
6 begin	end 6
7 begin	end 7
8 begin	end 8
9 begin	end 9
10 begin	end 10
11 begin	end 11
12 begin	end 12
13 begin	end 13
14 begin	end 14
15 begin	end 15
16 begin	end 16
17 begin	end 17
18 begin	end 18

1 begin
end 1| 2 begin
end 2| 3 begin
end 3| 4 begin
end 4 | 5 begin
end 5 | 6 begin
end 6 | 7 begin
end 7 | 8 begin
end 8 | 9 begin
end 9| 10 begin
end 10| 11 begin
end 11| 12 begin
end 12

Slice ها باعث می‌شود جلو انتشار خطا داخل فریم را بگیریم. دلیل خطا یکی VLC ودیگری بلاک نسبت به فریم قبلی کد شده باشد. حتی I داشته باشیم VLC باعث انتشار خطا می‌شود.

اگر از شبکه packet network استفاده شود بهتر هست packet ها independent باشند. بنابراین بهتر هست slice ها در packet مشابهی قرار گیرند. ساختار انکدر MPEG بصورت زیر است. در بخش frame-store فریم جاری و predictorها قرار می‌گیرد. Regulator بین بافر و Q پیچیده تر می‌شود. چون نوع فریم ها سه نوع می شود. تصویر I کلش I باشد، تصویر P: نسبت به قبلی است، تصویر B: نسبت به قبل و بعد کد می‌شوند و خودشان reference نیستند.

MPEG-Type encoder

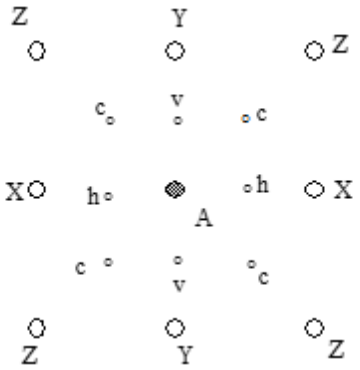


در دیکدر هم یک playback بافر داریم و فریم ها را دریافت می کند و بعد براساس ترتیب به دیکدر ارسال می‌شود.

جلسه ۱۳۹۶/۸/۲۷

تفاوت بین انکدر Mpeg و H.261: در Mpeg باید فریم‌ها reorder شوند. (تاخیر در دیکدر و انکدر) و باعث افزایش end-to-end delay می‌شود. در استریمینگ زیاد مساله ای با delay نداریم. ولی در 2-way communication مهم هست. در h.261 فرض مان این هست که تمام تصاویر inter کد کنیم. سعی می کنیم هر چند وقت یکباری یک intra کد کنیم تا جلو خطا را بگیریم در h.261 تمام یک تصویر اینترا کد نمی شوند. انکدر ها توصیه می کنند در هر فریم چند ماکرو بلاک بصورت اینترا کد شوند مثلا در هر فریم سه تا ماکروبلاک را اینترا کد می کنیم. یعنی از ۳۹۴ ماکروبلاک تقریبا بعد از ۱۳۲ فریم به اندازه کل یک فریم اینترا کد شده است (حدود ۵ ثانیه). در MPEG بعد از هر چند فریم اینتر، یک اینترا کد می کنیم اگر کل فریم اینترا شود باعث congestion در شبکه خواهد شد. به جای intra کردن کل فریم، Sliceها در فریم‌های متوالی intra کد شوند بهتر است تا از congestion در شبکه جلوگیری کند ولی کل فریم کد شود congestion ایجاد می‌شود. در فرکانس های بالا ما نسبت به ضرایب حساسیت کمتری داریم و می توانیم quantiser step size را بزرگ کنیم. از نظر Motion estimation برای P خرج بیشتر است. H.261 را براساس circuit switch طراحی کردند. Mpeg را برای desktop طراحی کردند از لحاظ شبکه در Mpeg1، I خوب نیست ولی B خوب هست بنابراین مزایا و معایبی دارد. Loop filter: موقعیکه ویدیو را کد می کنید بلاکی میشود که فیلتر برای حذف آن هست. در H.261، frame store یک فریم و در Mpeg دو فریم ذخیره می‌شوند (قبل و بعد برای B). Rate regulator: نرخ B, I, P متفاوت هست و تنظیم rate مشکلتر است. نوع تغییرات این سه تصویر هم متفاوت هست.

Motion estimation: در H.261 تخمین حرکت اختیاری هست چون در ان موقع توان پردازشی کم بود و این قضیه خرج زیادی دارد در H.261 تخمین صحیح است در حالیکه در Mpeg1، half pixel داریم.



تنظیم نرخ بیت: در Mpeg همه تصاویر نرخ بیت مشابهی ندارند. نرخ بیت GOP را باید بین I,P,B تقسیم کنیم. این تصمیم براساس پیچیدگی بیت هر تصویر، کوانتایز استپ سایز، و تعداد هر تصویر در GOP گرفته می‌شود. بنابراین پیچیدگی ویدیو برابر است با:

$$SCI = \frac{1}{12}\left[IQ_I + \sum_{j=1}^{3} P_j Q_{Pj} + \sum_{j=1}^{8} B_j Q_{Bj} \right]$$

چون در هر GOP یک I، سه P و هشت B داریم.

مثال: فرض کنید پهنای باند یک مگا بیت است به هر فریم چقدر بدهیم؟

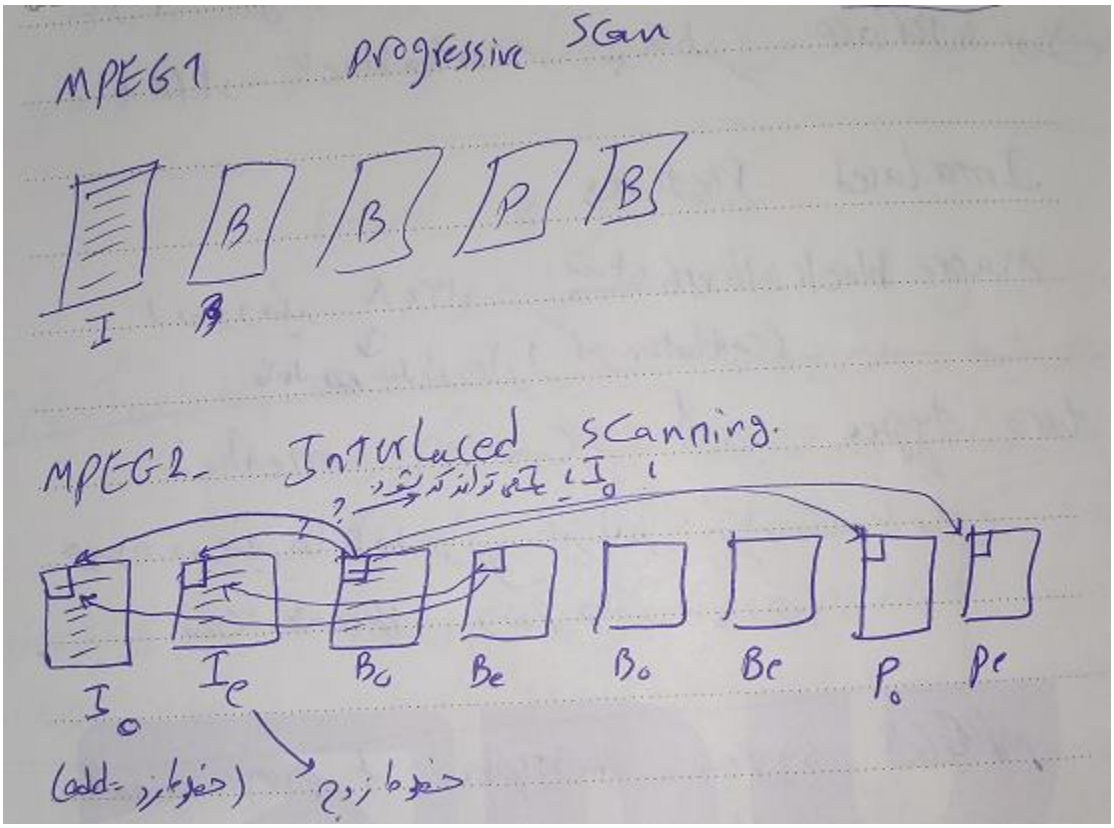
هر فریم بطور متوسط Frame=1000k/25=40kbps می‌خواهد. برای B خوب است برای P بالاخص I بد است.

بسته به پیچیدگی تصاویر بیت را بین آنها تقسیم می‌کنیم. اعداد بدست آمده برای I,B,P در نرخ ارسال متفاوت خواهد بود.

Mpeg1, h.261 تصاویر کوچک و progressive دارند.

MPEG2: در Broadcast از نوع Interlace استفاده می‌شود. Interlace یک سیستم ابتدایی فشرده سازی است. به خاطر منافع کاربر نمی توانند تکنولوژی را به سرعت عوض کنند. دو سیستم Interlace با هم یک فریم می‌سازند. در interlace فریم نداریم فیلد داریم و ۵۰ تصویر کد می شود که هر کدم می شود یک فریم. به Mpeg2 در مخابرات H.262 نیز می گویند.

جلسه ۱۳۹۶/۸/۲۹

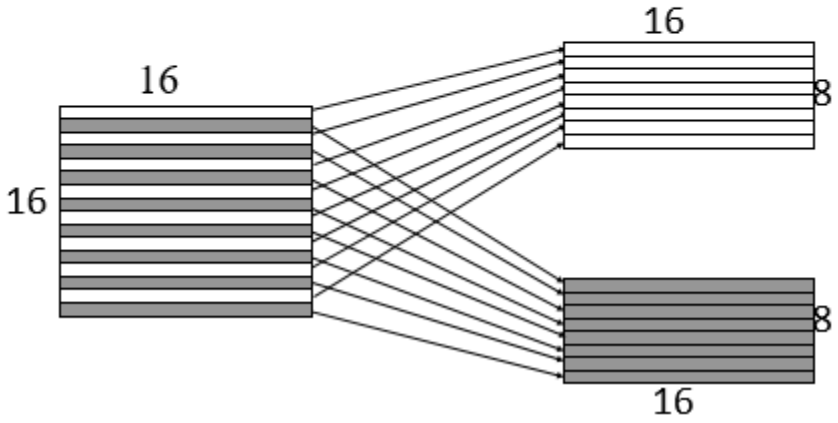


پیچیدگی با Interlace کردن ۴ برابر می‌شود که هر B با کدام P,I کد شود. برای کاهش خرج با پردازش بازی می‌کنیم. با دانستن سرعت می شود تشخیص داد که هر B با کدام I,P ساخته شود.

تفاوت های اصلی بین Mpeg1, Mpeg2: در Mpeg2 اندازه ماکروبلاک ۸*۱۶ به خاطر interlace بودن. یعنی ماکرو بلاک ۱۶*۱۶ تبدیل به دو عدد فیلد ۸*۱۶ می‌شود که چون فاصله بین خطوط دوبرابر هست correlation کمتر است. به صورت شکل زیر.

Interlaced pictures

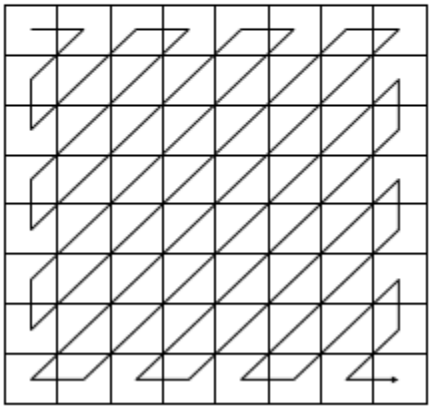
- For interlaced pictures, a target Macroblock can be split into two field macroblocks



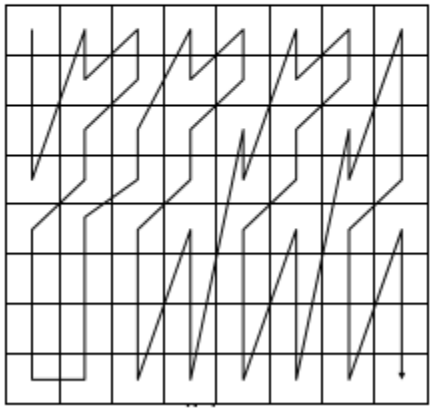
می توان ماکرو بلاک ۱۶*۱۶ کد کرد یا به دو بلاک ۸*۱۶ تبدیل کرده کد کرد. در حالت دوم فاصله بین خطوط زیاد می‌شود.

Two types of scanning method

For filed pictures, vertical distance is larger than horizontal distance and alternate scan cam meet larger values sooner



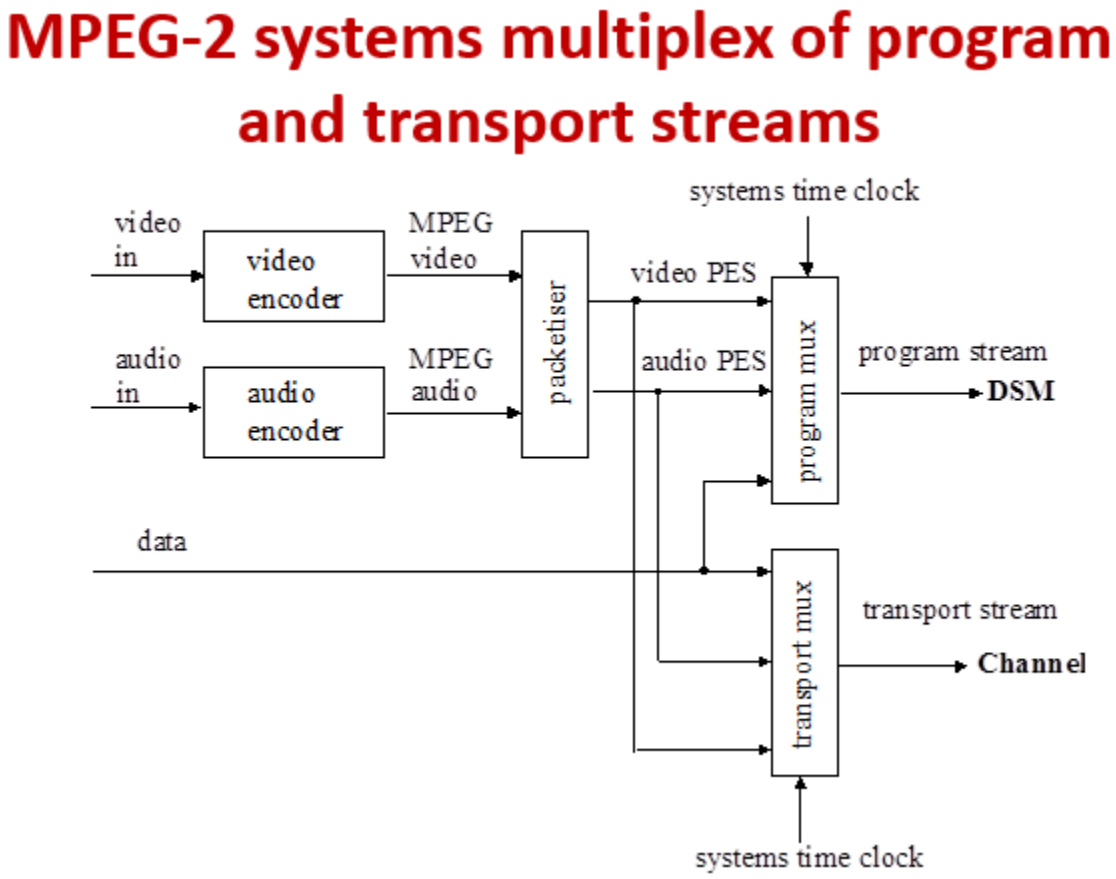
(a) zigzag scan



(b) alternate scan

چون در فیلدها فاصله عمودی از افقی بیشتر است، بنابراین alternate scan زودتر جواب می‌دهد.

حجم و پهنای باند صدا خیلی کمتر از ویدیو است. بنابراین معمولاً صدا نادیده گرفته می‌شود. قاطی کردن صدا و ویدیو به دو صورت است: *۱- فرکانس در آنالوگ (هم صدا هم تصویر) یک محدوده مال تصویر، محدوده دیگر مال صدا. ۲- packetization*. یکسری *packet مال صدا، یکسری مال ویدیو چون rate ویدیو بیشتر است، تعداد بسته های صدا کمتر است.* مثلا ۸ بسته برای ویدیو و یک بسته برای صدا در نظر می گیریم.



در شکل data گاهی استفاده می‌شود مثل teletext. محدودیتی در پهنای باند دیتا، صدا و تصویر نداریم. (هر بسته ۱۸۸ بایت هست). در مخابره end-to-end delay مهم است. فیلدها را دوربین‌های پخش تلویزیونی تولید می‌کنند. بنابراین اگر در پخش تلویزیونی نیستیم می توان از فیلد استفاده نکرد. (broadcast= interlaced)

Level & profile: هر کدک یک level و یک profile دارد.

Level: تعداد بیت ها را level می‌گویند.

Profile: چه کارهایی را انجام می‌دهد، مثلاً فریم I یا B کد شده است. یا قالب تصویر چیست؟ اندازه تصویر، درچه bit rateی کد شود؟ video codec براساس نیاز کاربر تولید می‌شود که چه چیزهایی مورد نیاز است.

Various profiles defined for MPEG-2

Type	Supporting tools	Application
Simple	I & P picture, 4:2:0 format; non-scalable	currently not used
Main	simple profile + B-pictures	broadcast TV
SNR scalable	main profile + SNR scalability	currently not used
Spatial	SNR profile + Spatial scalability	currently not used
High	spatial profile + 4:2:2 format	currently not used
4:2:2	IBIBIB... pictures, extension of main profile to high bit rates	studio postproduction; high quality video for storage (VTR) & Video distribution
Multiview	main profile + temporal scalability	several video streams; stereo presentation

مثلا simple profile: P,I دارد، تصویر 4:2:0 است. یا Main profile: simple profile+ B-picture

Multiview: از زوایای متفاوتی یک تصویر را ببینید. در صحنه های تلویزیونی ورزشی استفاده می‌شود.

The levels defined for each profile

Level	resolutions	Simple I,P 4:2:0	Main I,P,B 4:2:0	SNR I,P,B 4:2:0	Spatial I,P,B 4:2:0	High I,P,B 4:2:0 4:2:2	4:2:2 I,P,B 4:2:0 4:2:2	Multiview I,P,B 4:2:0
Low	pel/line			352	352			352
	line/fr			288	288			288
	fps			30/25	30/15			30/25
	Mbps			4	4			8
Main	pel/line	720	720	720		720	720	720
	line/fr	576	576	576		576	512/608	576
	fps	30/25	30/25	30/25		30/25	30/25	30/25
	Mbps	15	15	15		20	50	25
High 1440	pel/line	1440		1440	1440	1440		
	line/fr	1152		1152	1152	1152		
	fps	60		60	60	60		
	Mbps	60		60	80	100		
High	pel/line	1920			1920	1920		1920
	line/fr	1152			1152	1152		1152
	fps	60			60	60		60
	Mbps	80			100	130		300

جلسه ۱۳۹۶/۹/۴

Video on network: ۷۲ درصد ترافیک شبکه ویدیو است. اشکالات شبکه: packet loss تحت ترافیک که حل آن بافرینگ و افزایش پهنای باند بر اساس نیاز.

بافر بزرگ به درد two-way-communication نمی‌خورد. بنابراین محدودیت روی اندازه بافر است و این باعث سرریز بافر و گم شدن packet می‌شود.

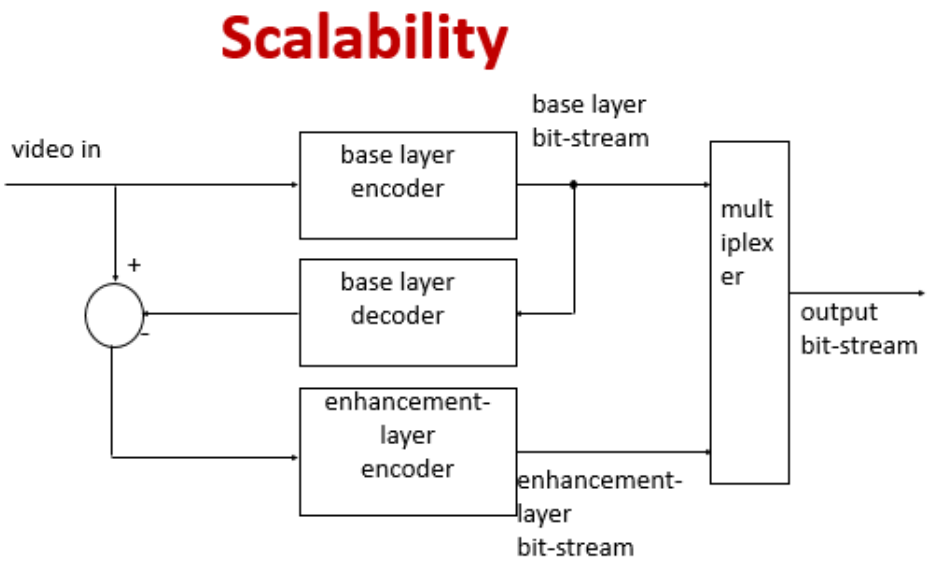
انواع دیتا: ۱- دیتای معمولی، که تاخیر را تحمل می کند ۲- صدا: کدک های خوب صدا را در 7kbps کد می‌کنند و موبایل 13kbps می‌باشد. ۳-ویدیو:میزان دیتا بالاست و احتمال اینکه مواجه با packet loss شود بالاست. سعی کنیم ویدیو را تا حد امکان فشرده کنیم. تغییرات پهنای باند؟؟ کدینگ scalable باشد یعنی در rate پایین یک کیفیت و هرچه پهنای باند بالاتر شود کیفیت بالاتر باشد.

Scalability Vs Layering

Scalability: یعنی ویدیو در یک رشته بیت کد می شود بطوریکه از یک رشته بیت بتوان انواع مختلف فرمت های ویدیو را دیکد کرد.

Layering: یعنی بخشی از رشته بیت طوری کد شود که اطلاعات ضروری بتواند محافظت شوند (مثلا در مقابل packet loss یا خطاهای کانال)

به عبارت دیگر در scalable فرمت های ویدیو گوناگون برای توزیع بیشتر استفاده می‌شود و در layering: انتظار داریم همه لایه ها را دریافت کنیم. ولی لایه ها اهمیت مختلف دارند لایه های پایین اولویت بیشتری دارد. در صورت ترافیک لایه های بالاتر حذف می‌شود. امروزه از scalability کمتر استفاده می‌شود چرا؟ چون user ها زیاد است و یک سرور جوابگو نیست و scalable coding سربار دارد و کدک پیچیده است زمانی مفید هست که سرور آنقدر قوی باشد که بتواند به میلیون ها کاربر جواب دهد. شکل scalability در زیر نشان داده شده است. لایه بالا base layer کمترین کیفیت را دارد (حداقل مورد نیاز)



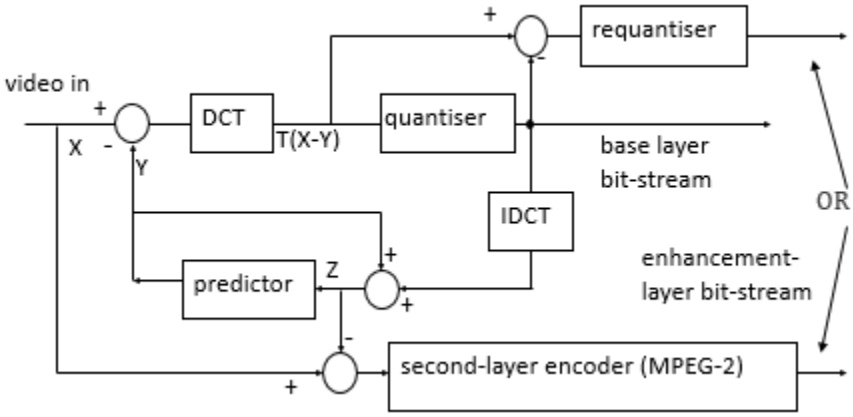
انواع Scalability: سه نوع مقیاس پذیری داریم:

۱- Quality or SNR scalability: لایه های پایه و توسعه یافته فقط در کیفیت فرق دارند. Quantizer step size در لایه توسعه کوچکتر از لایه پایه است و کیفیت enhanced بهتر است.

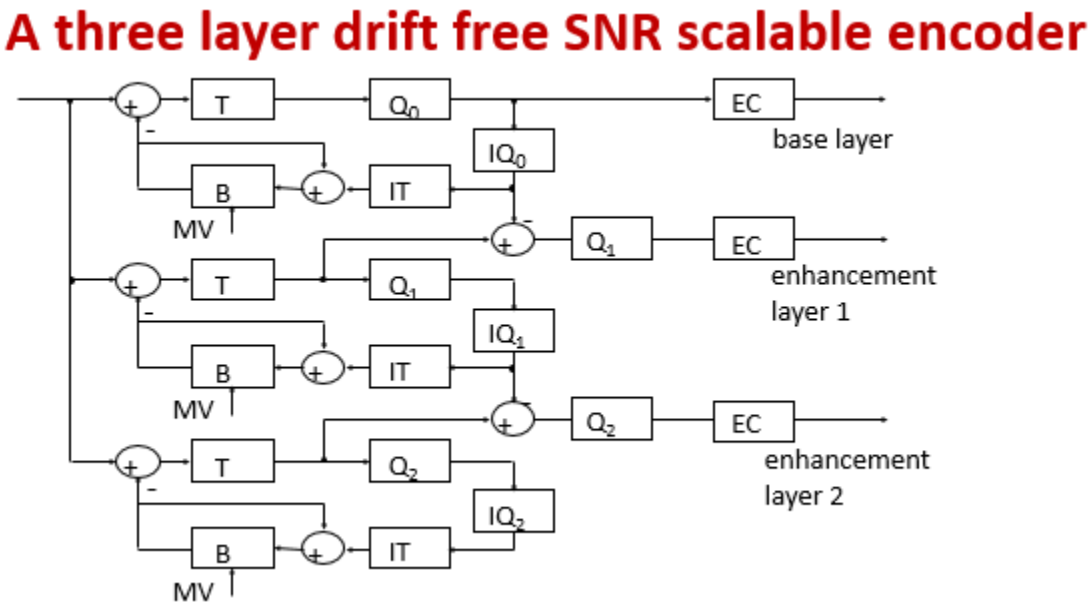
۲-spatial scalability: لایه های پایه و توسعه یافته از لحاظ اندازه تصویر متفاوتند. کیفیت و Quantizer step size مثل هم است ولی اندازه تصاویر در base و enhanced متفاوت است.

۳- temporal scalability: کیفیت و اندازه مشابه هستند اما لایه های پایه و توسعه یافته فقط در نرخ فریم متفاوتند. (base ۵ فریم و enhance ۲۵ فریم در ثانیه). Enhance چندین لایه ایجاد می‌کند. می‌شود هر سه مورد را با هم ترکیب کرد.

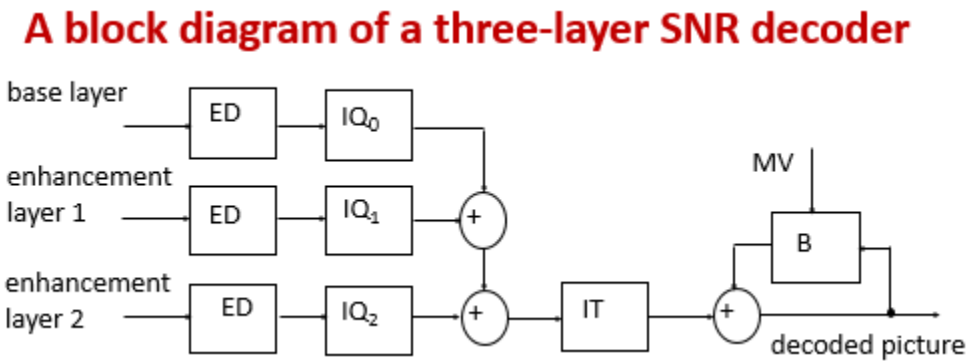
SNR Scalability: هر دو لایه انکدر مشابهی استفاده می کنند، و کدک SNR scalability می تواند بصورت زیر باشد:



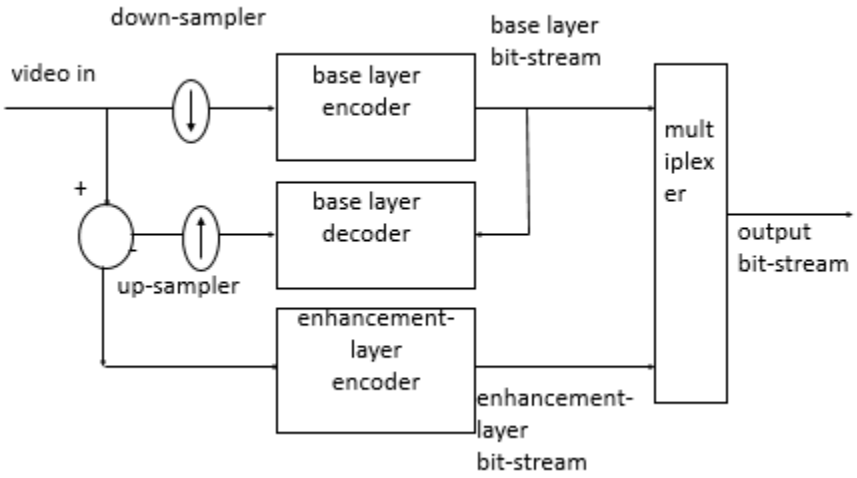
SNR Scalability را می توان به صورت دولایه و سه لایه ساخت. در سه لایه لایه پایه و لایه ۱ و لایه ۲ بصورت موازی کار می کنند.



لایه پایه مهم تر هست. اگر لایه پایه گم شود و لایه های ۱ و ۲ دریافت شوند دریافت بدرد نمی خورد. در دیاگرام دیکدر زیر اهمیت لایه ۲ کمتر از لایه ۱ و لایه ۱ کمتر از base layer است. اگر base layer گم شود، لایه ۲ و لایه ۳ بی اهمیت میشود. در گیرنده اگر نرخ packet loss را در نظر بگیریم مثلا نرخ گم شدن لایه پایه اگر 10^{-4} باشد لایه ۱: 10^{-3} و لایه ۳ 10^{-2} خواهد شد. هر چه میزان داده‌های ارسالی بیشتر باشد packet loss ratio بیشتر است.



Spatial Scalability:



در این حالت ابعاد تصویر در لایه ها فرق می کند. ابعاد base کوچکتر و ابعاد enhancement بزرگتر است. در تصاویر کوچکتر فشرده سازی کمتر است. هر موقع تصویر را کوچک می کنیم یعنی فرکانس های بالا را از دست بدهیم حالا اگر دوباره بزرگ کنیم فرکانس های حذف شده را دیگر نخواهیم داشت به این اثر aliasing artifact می‌گویند. بنابراین کار enhancement layer کد کردن فرکانس های بالا است. که کار سختی است.

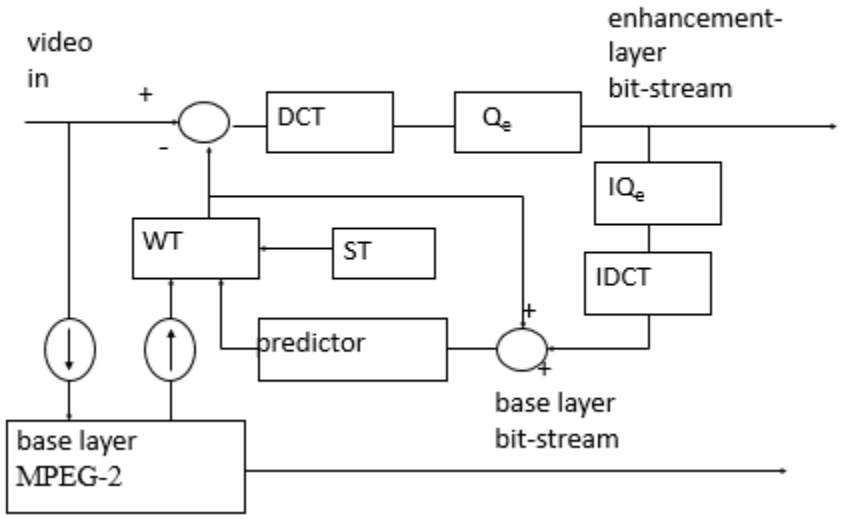
کاربرد: ارسال HD و SD: در اینصورت در لایه پایه SD ارسال می‌شود. در لایه enhance، HD کد می‌کند. چرا به جای یک فرستنده Scalable از دو فرستنده با دو حالت مختلف استفاده می شود؟

یک فرستنده به صرفه تر ➔SD+HD

بطور جدا با دو فرستنده. این پرکاربردتر است. چرا؟ 1-SD , 2-HD

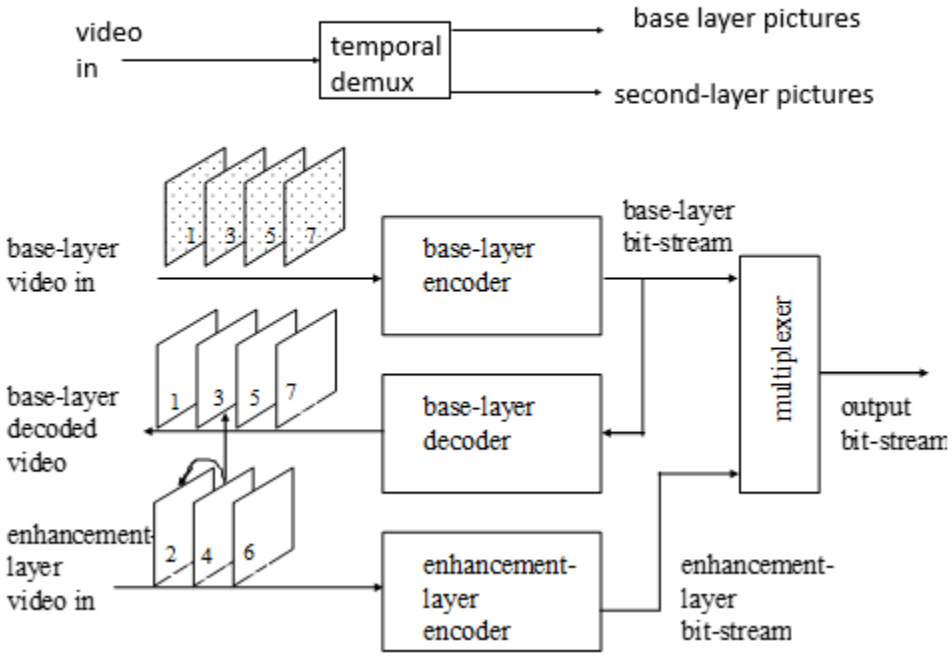
برای تطابق با تکنولوژی. سیستم های قدیمی تر SD را استفاده می کنند و سیستم های جدید HD را می توانند استفاده کنند. دیکدری که دو تا SD+HD را دیکد کند از دیکدری که فقط یکی از آنها را دیکد می کند ارزان تر است.

Details of spatial scalability encoder



Temporal Scalability: فرق این هست که در لایه پایه نرخ کمتر است. ابعاد و رزولوشن مشابه هست. مثلا فریم های فرد در لایه پایه کد شوند و لایه توسعه یافته زوج را کد کند. فریم ۲ مثلا از فریم های ۱و ۳ predict می‌گیرد و مشابه B-picture می‌شود. در Temporal scalability برخلاف قبلی ها که در لایه های توسعه نرخ بیت زیاد می‌شد، در لایه های توسعه نرخ بیت کم می‌شود چون نوع B بهتر کد می‌شوند.

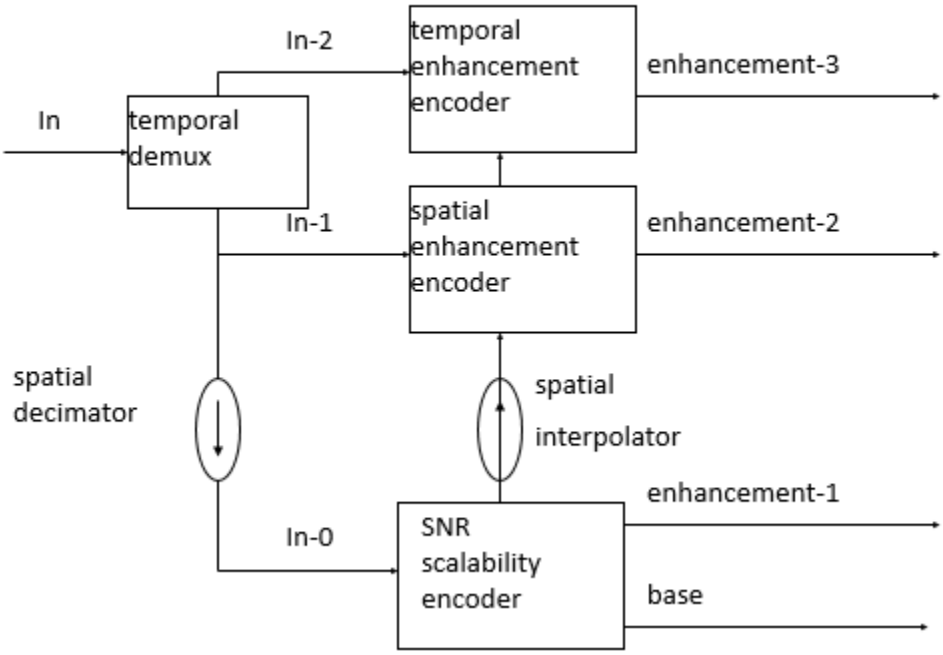
Temporal scalability



در لایه پایه فریم ها I,P کد می‌شوند در enhance تصاویر B است. البته در enhance هر فریم لزوماً B نیست می‌تواند از P کد شود می‌تواند I کد شود و بهتری ارسال شود. ترجیح می‌دهیم B از دو تا بدست آید برای حذف یا کم کردن error. اگر در فریم ۱ خطا داشته باشیم ممکن است در ۳ نباشد. و فریم ۲ از ۱و ۳ استفاده می‌کند میانگین خطا را حذف می‌کنیم.

ترکیب SNR, spatial, temporal:

SNR, spatial and temporal hybrid scalability encoder



دلیل سربار spatial scalability: به خاطر aliasing artifact و کد کردن فرکانس های بالا. و کاربرد آن تغییر فرکانس هست مثل کد کردن SD, HD. در temporal scalability می‌توان روی progressive کار کرد.

Applications of SNR scalability

Base layer	Enhancement layer	Application
ITU-R-601	same resolution and format as lower layer	two quality service for standard TV
high definition	same resolution and format as lower layer	two quality service for HDTV
4:2:0 high definition	4:2:2 chroma simulcast	video production/distribution

Applications of spatial scalability

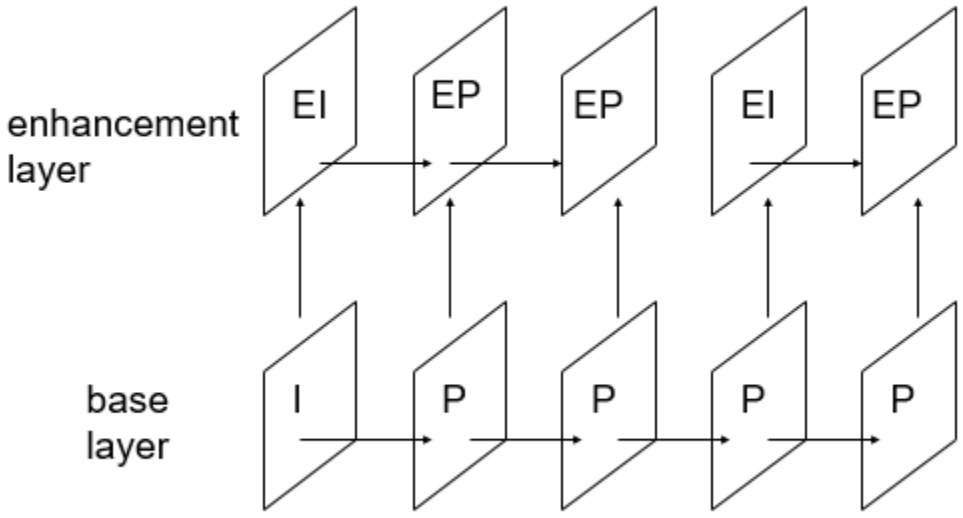
Base	Enhancement	Application
progressive(30Hz)	progressive(30Hz)	CIF/QCIF compatibility or scalability
interlace(30Hz)	interlace(30Hz)	HDTV/SDTV scalability
progressive(30Hz)	interlace(30Hz)	ISO/IECE11172-2/compatibility with this specification
interlace(30Hz)	progressive(60Hz)	Migration to HR progressive HDTV

Applications of temporal scalability

Base	Enhancement	Higher	Application
Progressive (30Hz)	Progressive (30Hz)	progressive (60Hz)	migration to HR progressiveHDTV
Interlace (30Hz)	Interlace (30Hz)	progressive (60Hz)	migration to HR progressiveHDTV

جلسه ۱۳۹۶/۹/۱۱

SNR Scalability

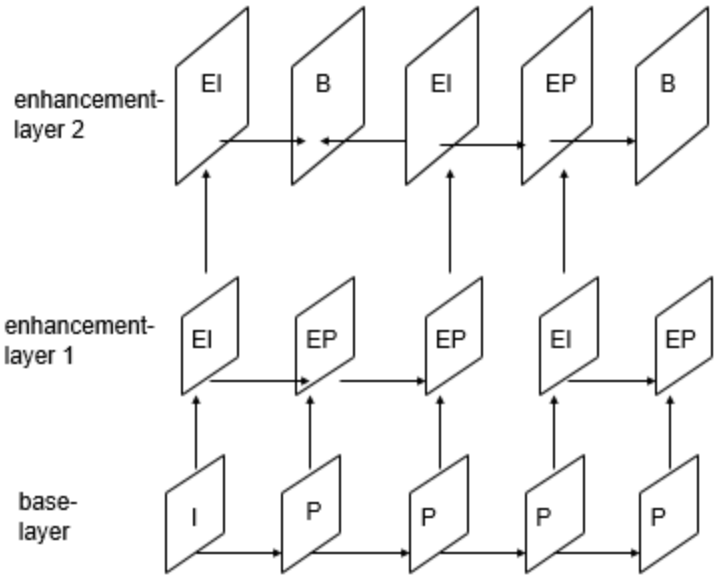


Prediction flow in SNR scalability

Ei فقط prediction از پایین می‌گیرد. Ep از بالا و پایین. EI برای قطع خطا هست. (جلوگیری از انتشار خطا).

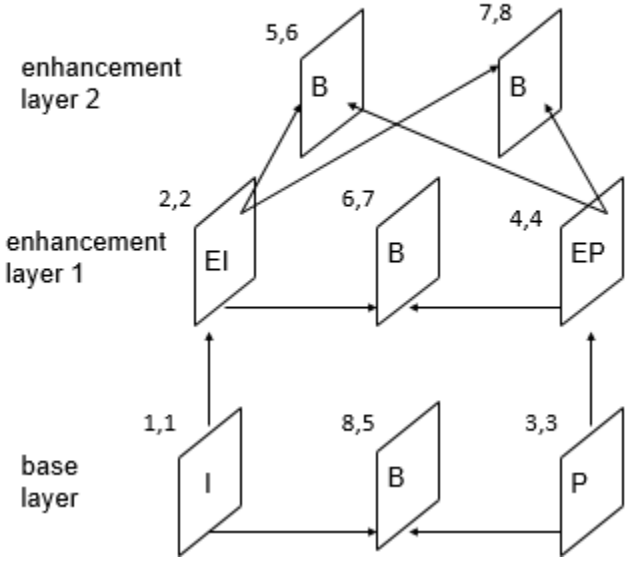
I: بیت بالاتر دارد اما از error propagation جلوگیری می‌کند. اما بیت بیشتر در شبکه congestion ایجاد می‌کند.

Multilayer scalability



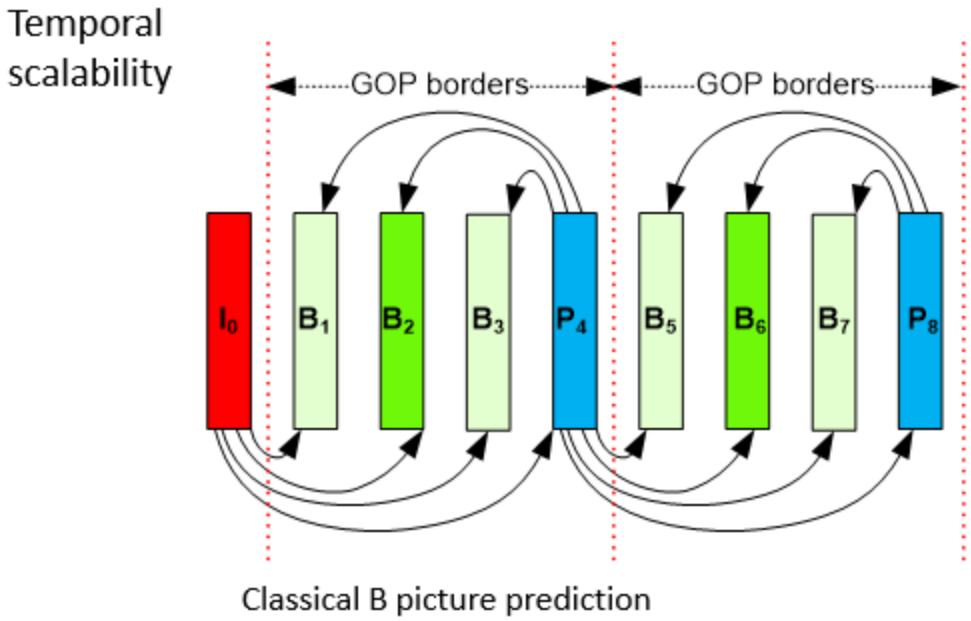
در enhance2 مقیاس پذیری temporal اضافه شد. می توان سه نوع مقیاس پذیری را با هم ترکیب کرد. اما سوال اینجاست که ترتیب کد کردن فریم ها چگونه باشد؟

Transmission order of pictures



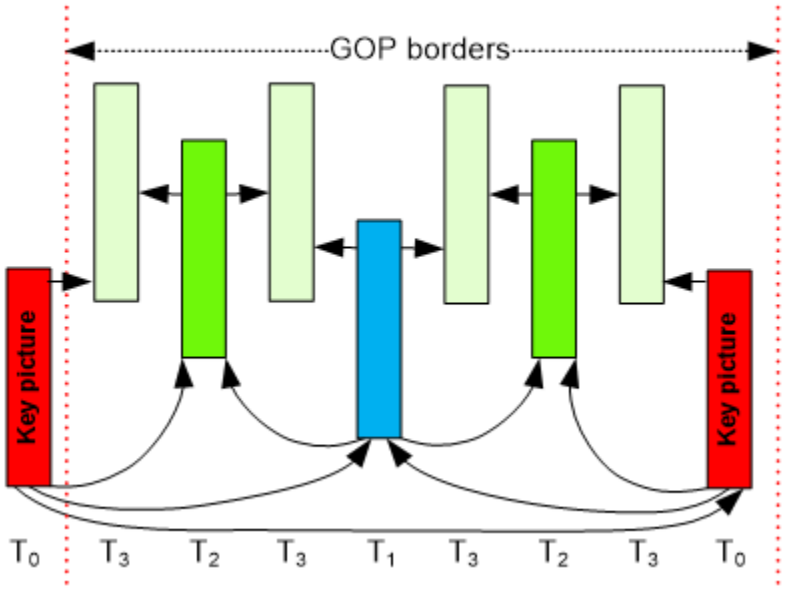
در شکل فوق دو سناریو داریم. روی هر یال دو عدد بصورت عدد۲/عدد۱ وجود دارد که عدد۱ یک سناریو و عدد۲ سناریو دیگری است. اگر فریم B نداشته باشیم می‌توانیم پشت سر هم کد کنیم. یعنی وجود B ایجاد تاخیر می‌کند و end-to-end delay زیاد می‌شود. برای two-way-communication خوب نیست ولی برای streaming اشکالی ندارد. مزیت این هست که در صورت تراکم شبکه، B را می‌توان دور انداخت.

Scalability in H.264/MPEG-4



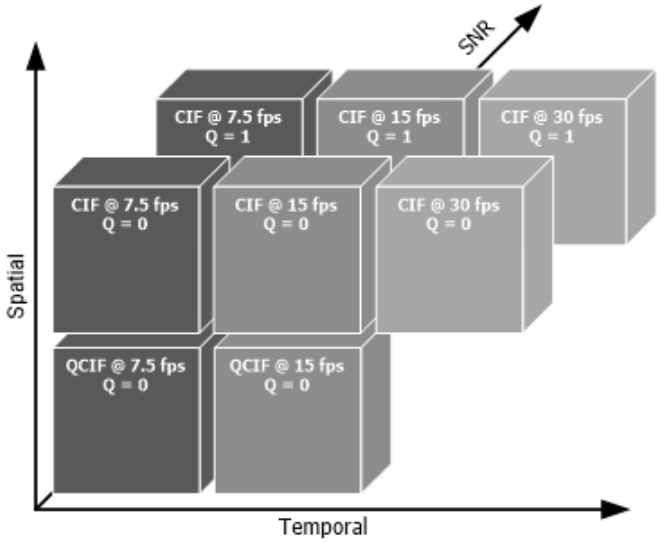
حذف B ها تاثیر آنچنانی روی کیفیت ندارد. B حق انتخاب دارد. بیت پایین است. افزایش تعداد B باعث می‌شود فاصله B با فریم‌های P زیاد شود و یا B شبه P کد شود یا خوب کد نشود. بنابراین تعداد Bها، ۱،۲،۳ باشد خوب هست بیشتر از ۳ خاصیت ندارد (بیت زیاد می‌شود).

Hierarchical B pictures with 4 dyadic levels (GOP of 8).



در شکل فوق خود B هم برای کد شدن پایه شده است. سبزهای کم رنگ از همه کم اهمیت تر هستند و می توانند در حالت congestion حذف شود. بعد سبز پررنگ. البته end-to-end delay بد می‌شود. ۷ فریم باید صبر کنیم تا تصویر را بسازیم.

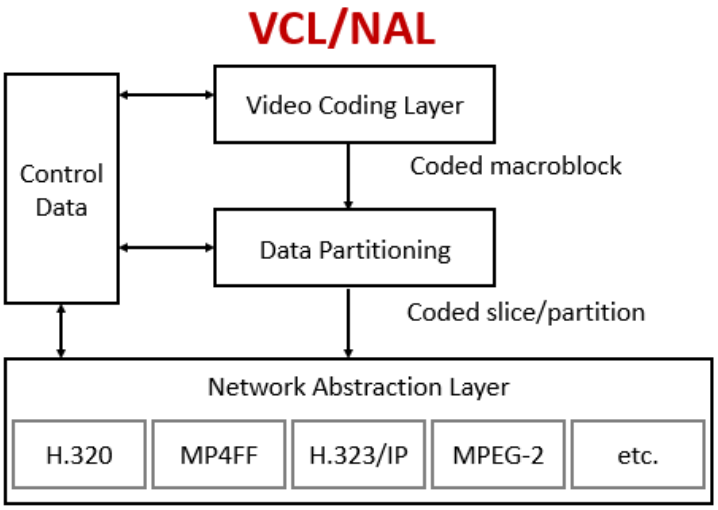
3D representation of spatial-temporal-quality at various points



در شکل فوق ترکیب سه نوع کیفیت مقیاس پذیر نشان داده شده است. در محور x نرخ ارسال فریم تغییر می کند بدون تغییر ابعاد و quantizer step size که از نوع temporal هست و در محور y اندازه تصویر تغییر می کند از نوع spatial و روی محور z کیفیت تغییر می کند با تغییر quantizer step size. هدف این هست که ویدیو ها طوری کد شوند که قسمتی از قسمت دیگر مهمتر باشد.

Advanced video coding(AVC/H.264/Mpeg-4-v-10): اهداف ساخت این کدک عبارت است از: ۵۰٪ از کدک قبلی بهتر باشد، نسبت به delay، adaptive باشد تا ۳۱ عدد B-frame می تواند داشته باشد، error resilience باشد برای موبایل قابل استفاده باشد که خطا در آن بالاست یک روش layering هست و forward error correction است، network friendliness باشد: در هر شبکه‌ای بشود از آن استفاده کرد. که برای اینکار کدک را به دو لایه تقسیم کردند.

لایه های انتزاعی AVC: لایه VCL که کار کدینگ را انجام می دهد. NAL فقط نگاه میکند که ویدیو را به چه شبکه ای وصل خواهید کرد.

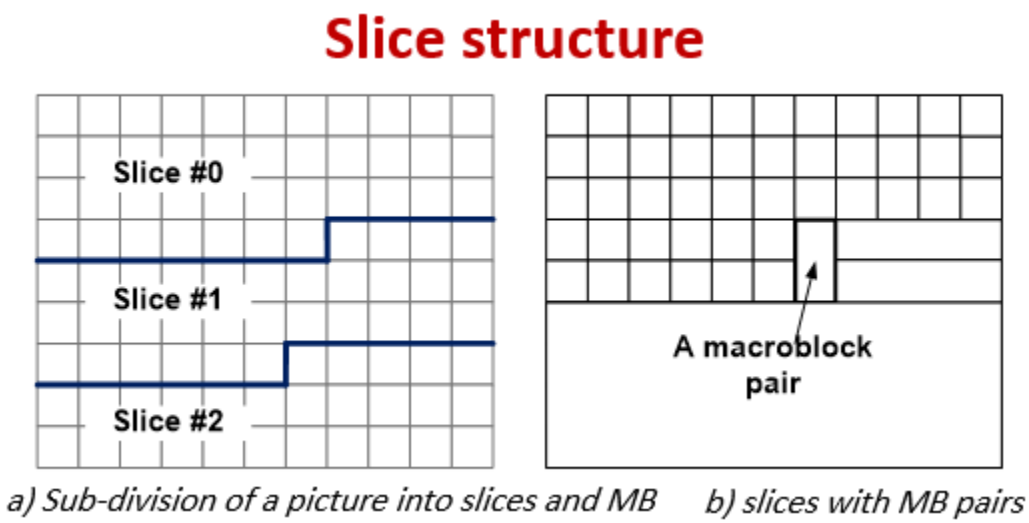


ویژگی‌های H.264/AVC/MPEG-4:

- از دو شماتیک entropy coding استفاده می کند شامل CAVLC (context adaptive variable length coding) and CABAC (context adaptive binary arithmetic coding).
CABAC ۸ درصد بهتر از CAVLC (یا همان هافمن) است.
- Multiple reference: اجازه می دهد B-slice ها به عنوان رفرنس استفاده شوند حتی برای P-sliceها. بنابراین Pها خیلی خوب کد می‌شوند.
- اندازه بلاک متغیر (4*4 to 16*16). Transform هم ۸*۸ هم ۴*۴ باشد. قبلا فقط ۸*۸ داشتیم.
- دقت تا یک چهارم پیکسل (در موبایل مناسب هست چون تصویر کوچک QCIF می‌شود). Motion کوچک، دقت بالا، motion بزرگ: دقت کم.
- ماکروبلاک ۱۶*۱۶ از پیکسل‌های اطرافش هم prediction می‌گیرد.
- به جای DCT از integer transform استفاده می‌شود. بدی DCT این هست که همه المان های آن اعشاری هست و ساخت inverse آن دقیق نیست. و inverse transform mismatch داریم.
- Deblocking filter: برای حذف اثرات ایجاد شده توسط motion compensation, quantisation استفاده می‌شود. این فیلتر اثرات blockiness را حذف می‌کند.
- Parameter set: ابعاد تصویر ، فرمت رنگ و ... برای هر فریم نیازی نیست ارسال شود. کلی انجام می‌شود تا زمانیکه تغییر نکرد ثابت است.
- به خاطر Slice کردن فریم تحمل خطا بیشتر می شود. Error resilience بهتر است.
- Data partitioning: تشخیص داده‌های مهم از غیر مهم. خود یک ویدیو قسمت هایش از هم جدا شود.
- Redundant transmission: یک مقدار افزونه از بسته قبلی در بسته جدید قرار می‌دهند که در صورتی که بسته قبلی بد بود از آن استفاده می‌شود در غیر اینصورت دور ریخته می‌شود.

جلسه ۱۳۹۶/۹/۱۳

Slicing: یکسری ماکروبلاک پشت سر هم را یک اسلایس می‌گویند.



اگر از packet network استفاده شود. هر اسلایس یک packet شود. وابسته به نوع شبکه و اندازه بسته، sliceها را می‌توان تعیین کرد. مبنا بر slice هست نه تصویر. اگر فریم کوچک بود کل آن را یک slice در نظر می‌گیریم.

Slice خیلی در مقابل خطا robust می باشد.

Bits/per slice: انواع مختلف اسلایس براساس تعداد ماکروبلاک هست که عبارتند از:

- یک اسلایس در هر تصویر: ساده ترین حالت هست و کل تصویر در یک اسلایس قرار می گیرد.

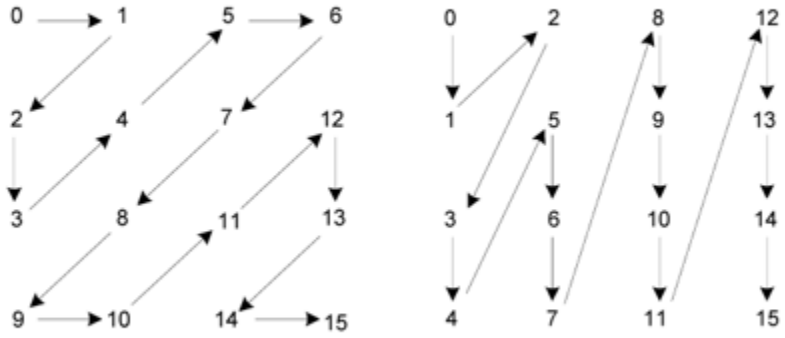
Efficiency= (payload size)/(total packet size)

در این حالت بازدهی بهتر هست.

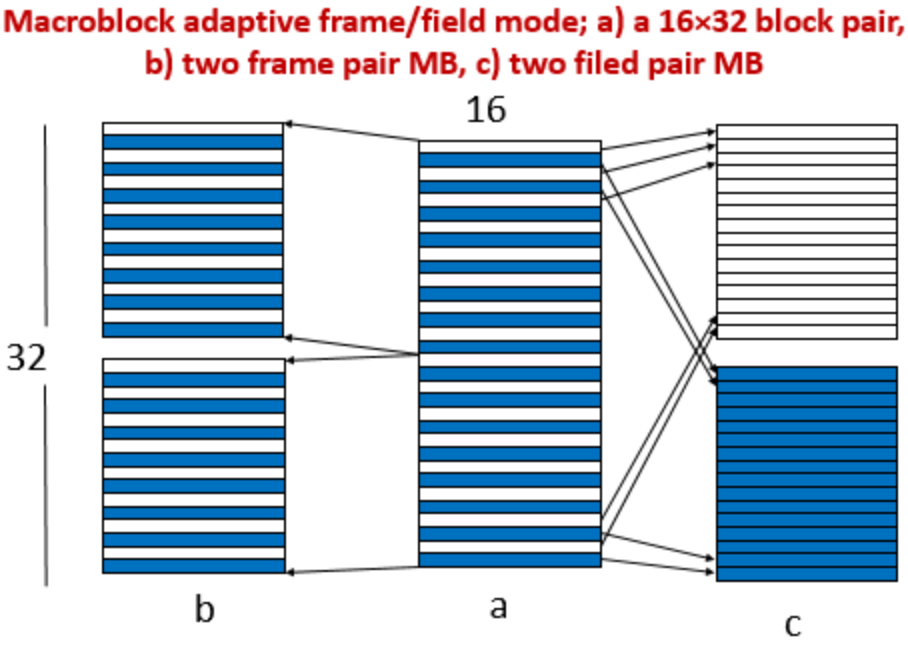
- تعداد ثابت ماکرو بلاک در هر اسلایس
- تعداد ثابت بایت در هر اسلایس
- اسلایس‌ها می‌توانند برای error resilience coding به گروه تقسیم شوند.

انواع slice: ۱-I-Slice، اسلایسی که همه ماکروبلاک های آن اینترا کد شده باشند. ۲- P-slice: ماکروبلاک هایی که بصورت اینتر کد شده باشند. ۳-B-Slice: ماکروبلاک ها از قبل و بعد predict می‌گیرند. ۴- SP-Slice (switching slice/picture) ۵- SI-slice

Progressive and Interlaced coding

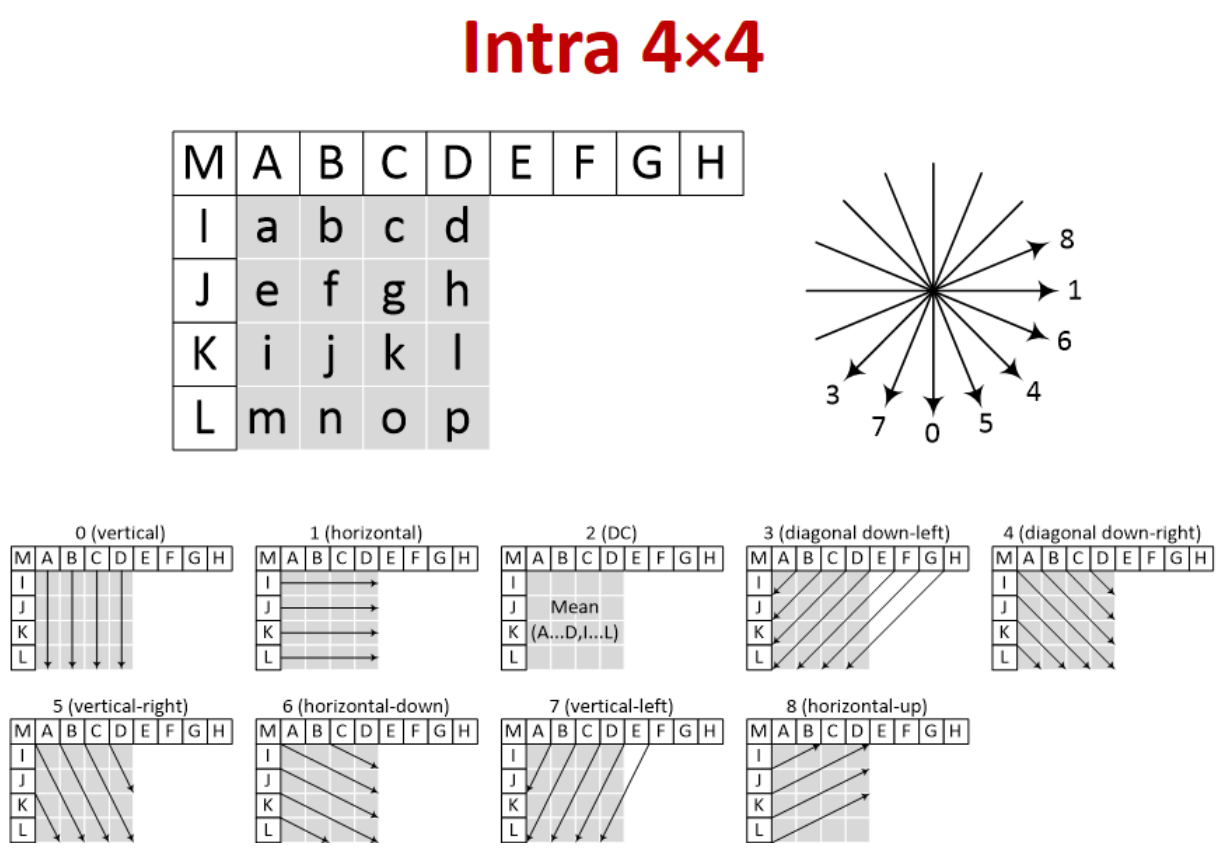


a) frame scan and b) field scan



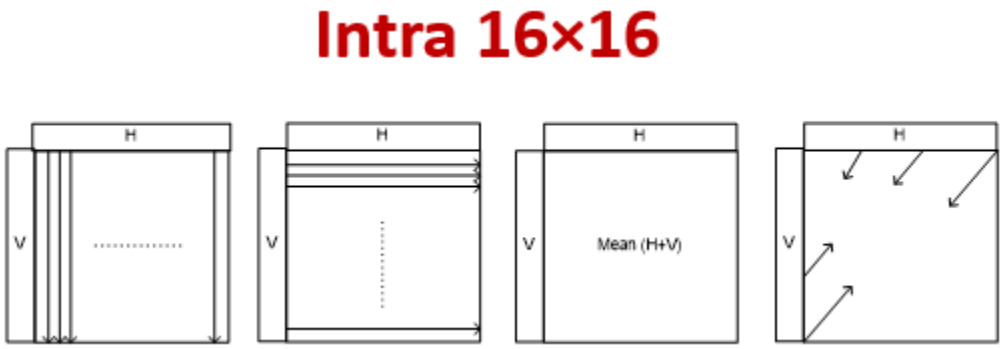
در شکل فوق یک زوج بلاک ۱۶*۳۲ در a داریم که می تواند به دو صورت کد شود. در گزینه b فرقی بین خطوط زوج و فرد نمی‌گذارد. در c خطوط زوج و فرد جدا می‌شوند (سرعت زیاد باشد بهتر است).

در بلاک ۱۶*۱۶ علاوه بر اینکه پیکسل‌های داخل بلاک به هم شبیه هستند به کناری‌های آن خارج ماکروبلاک هم شبیه هستند.



جهت های تشابه ۹ عدد است. مثلا جهت صفر عمودی و جهت ۱ افقی هست. مد ۲ برای تصاویر plain مناسب است. داخل خود تصویر از خود فریم prediction می‌گیریم. ولی چون هنوز داخل frame هستیم intra گفته می‌شود. در Sliceبندی، Sliceها نمی‌توانند از sliceهای دیگر prediction بگیرند. انکدر همه ۹ حالت را در نظر می گیرد. هر کدام بهتر بود استفاده می‌شود. (بار پردازشی خیلی زیاد است). فرق با Mpeg2 وجود همین modeها هست.

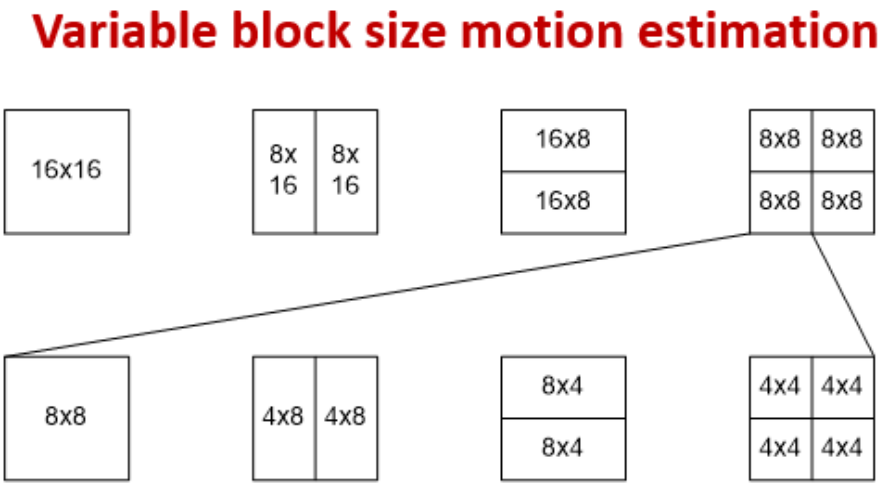
Intra 16*16: نسبت به ۴*۴ سربار کمتر است. فقط در ۴ مد prediction می‌گیرد.



Chroma prediction: آیا برای رنگ‌ها از همان Predictionی استفاده می کنیم که برای luminance استفاده کردیم؟ خیر لزوماً. چون texture رنگ فرق می‌کند. در motion، luminance, chrominance را مشابه استفاده می‌کنیم. ولی در اینجا می توانند متفاوت باشند.

Inter prediction: P-sliceها از فریم های قبل و بعد می توانند prediction بگیرند. یک لیست ۰ مربوط به p ها هست و برای Bها دو لیست صفر و یک دارند.

Motion compensation: block size ها متفاوت است. اگر luminance سه پیکسل حرکت داشت برای chrominance، 1.5 پیکسل هست چون ابعاد آن نصف هست.

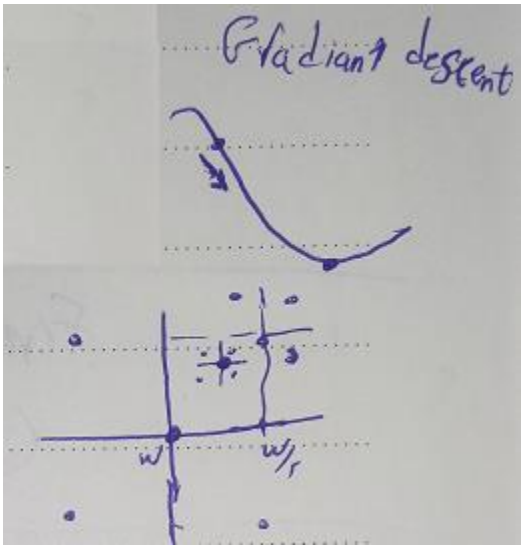


خود اندازه بلاک متغیر باعث پیچیدگی می شود. همه بررسی می شوند بهترین انتخاب می‌شود. هر ماکرو بلاک ۱۶*۱۶ به تعداد زیادی حالت قابل تبدیل هست و سربار زیاد می‌شود. اگر processing power داشته باشیم همه موارد چک می‌شود و بهترین

استفاده می‌شود. در غیر اینصورت یک راه این هست که تخمین زده شود کدام مورد بهتر است. اگر کسی بتواند بدون کدینگ تشخیص بدهد کدام بهتر است کار را ساده می‌کند.

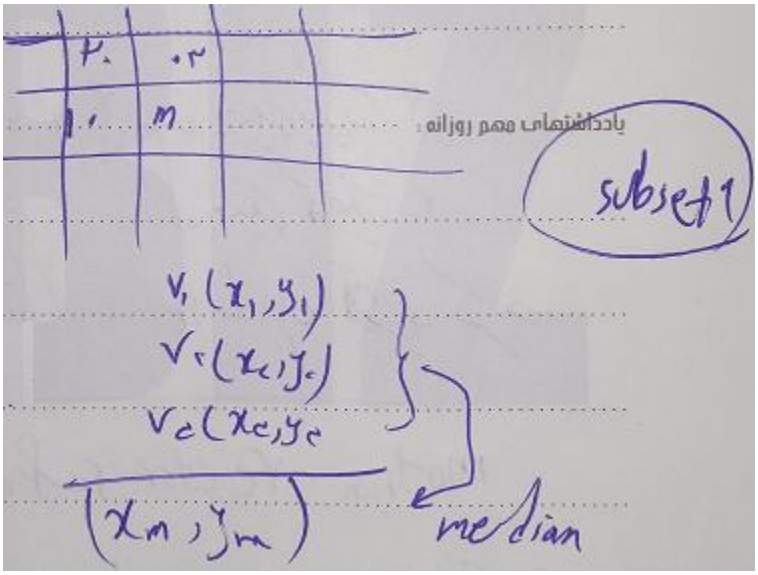
جلسه ۱۳۹۶/۹/۱۸

Fast motion estimation in H.264

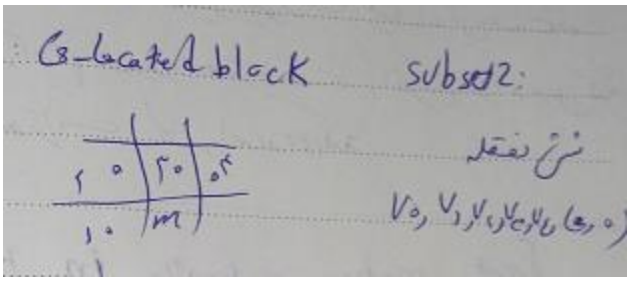


W مرکز است. چهار نقطه اطراف آن چک می‌شود نقطه‌ای که نزدیکتر است مجدد مرکز می‌شود و این عمل روی آن انجام می‌شود و الی آخر.

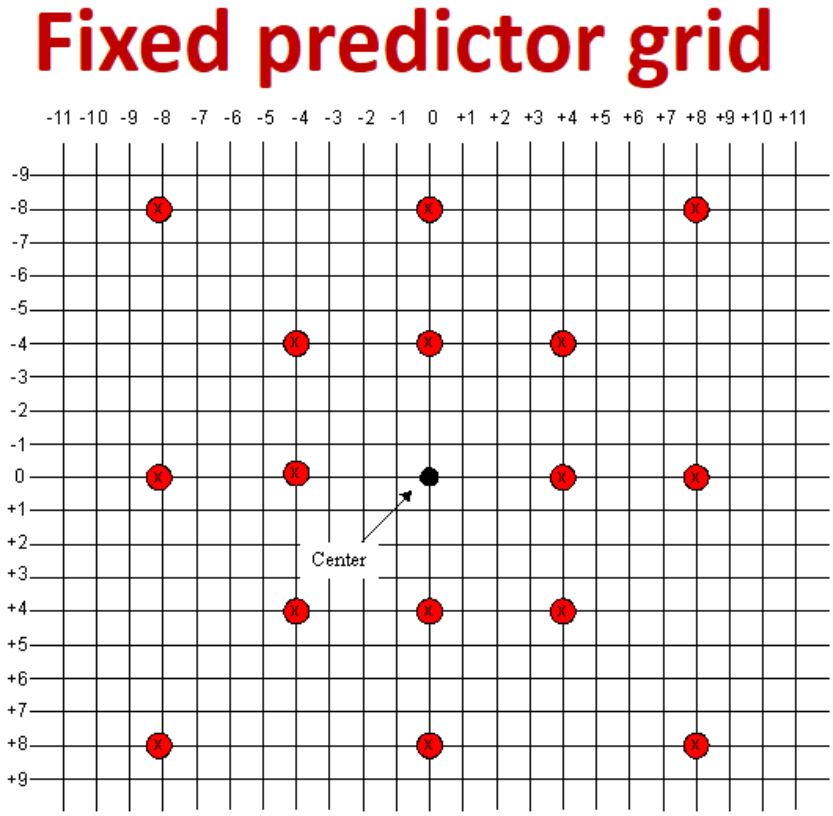
Zonal search algorithms: فرض می‌کنیم می خواهیم برای m ، motion محاسبه کنیم یک روش محاسبه motion سه نقطه چپ، بالا و بالای چپ است.



آیا تخمین درست است؟ یک Threshold در نظر می‌گیریم اگر کمتر از آنها بود خوب است.



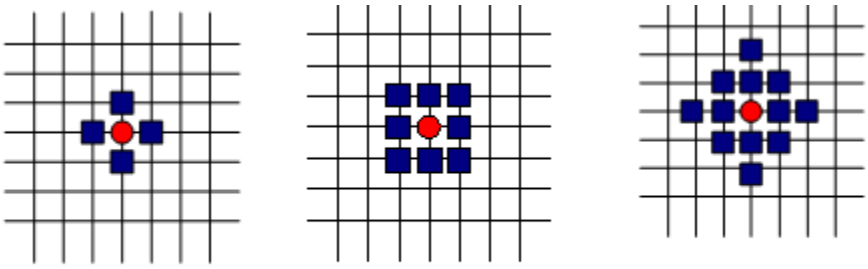
Subset2: collocated block: همان بلوک در فریم قبلی و با شش نقطه حرکت را پیدا می‌کنیم. طبق شکل فوق.



Additional fixed predictor grid for search range 16.

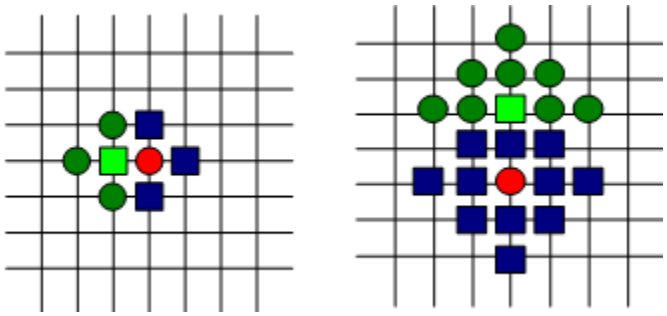
Threshold را چطوری گرفتیم؟ خیلی بزرگ؟ خیلی کوچک؟ می توان خود threshold را بروز کنیم.

طرح های دیگری هم برای مقایسه انجام می‌شود.



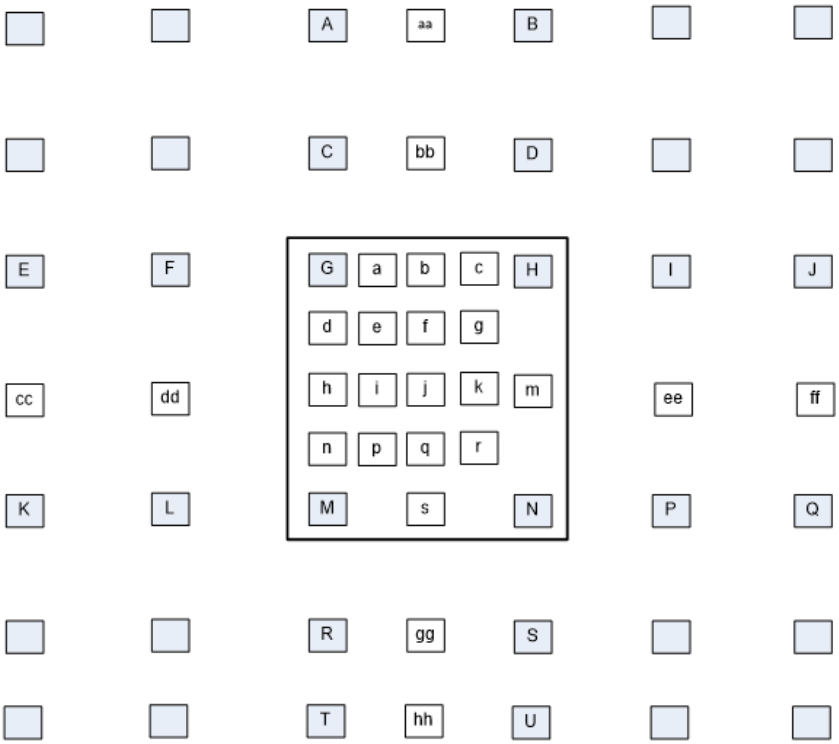
patterns, a) small diamond, b) square and c) large diamond

Iteration of search for a) small diamond and b) large diamond



Overlap ها را مجدد جستجو نمی کنیم.

Fractional precision of motion vectors



Filtering operations

$$b_1 = (E - 5 \times F + 20 \times G + 20 \times H - 5 \times I + j)$$
$$h_1 = (A - 5 \times C + 20 \times G + 20 \times M - 5 \times R + T)$$

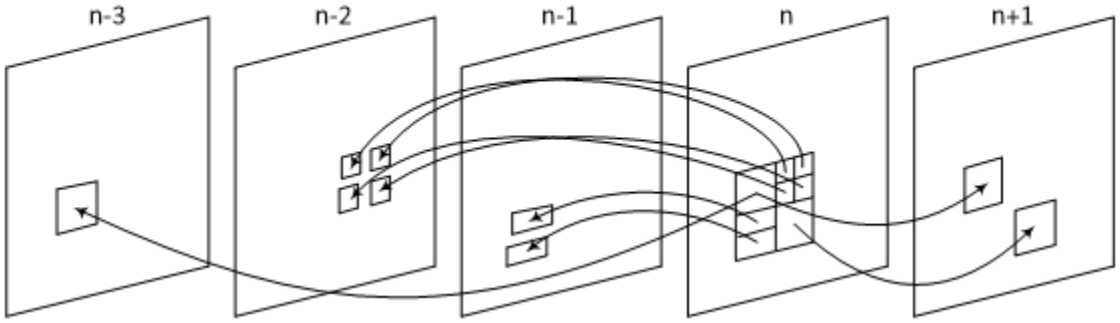
$$b = Clip(b_1 + 16) \gg 5$$
$$h = Clip(h_1 + 16) \gg 5$$

$$j_1 = cc - 5 \times dd + 20 \times h_1 + 20 \times m_1 - 5 \times ee + ff$$
$$j_1 = aa - 5 \times bb + 20 \times b_1 + 20 \times s_1 - 5 \times gg + hh \quad \textbf{or}$$
$$j = Clip(j_1 + 512) \gg 10$$

$$a = Clip(G + b + 1) \gg 1 \qquad e = Clip(b + h + 1) \gg 1$$

موارد در B پیچیده تر است چون ترکیب دو فریم گذشته و آینده است. بلاکی اسکیپ است اگر هیچ حرکتی نداشته باشد. در اینجا چیزی skip است که حرکت آن حرکت بلاک قبلی است.

Multiple reference picture motion compensation



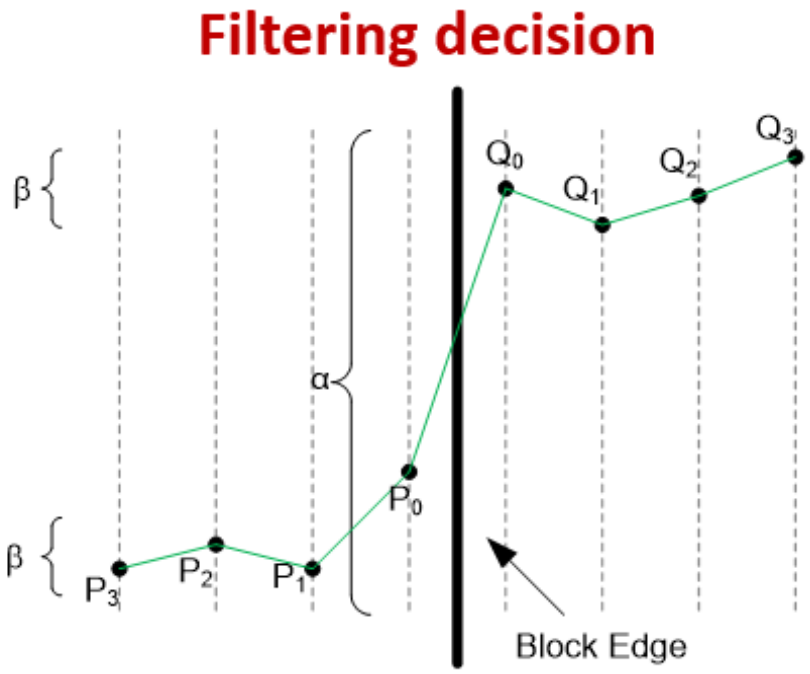
تبدیل: بجای DCT از تبدیل integer استفاده می‌شود. عملیات تبدیل و reverse آن مشابه قبل هست با این تفاوت که ضرایب صحیح می‌باشد.

خلاصه: در انکدر یک بلاکی میگیریم و motion compensate می کنیم سپس آن را به ۴*۴ تقسیم می کنیم حتی اگر اول ۱۶*۱۶ motion compensate شده باشند. ۱۶*۱۶ با ۴*۴ فرقش این هست که در ۱۶*۱۶ یک motion vector داریم و در ۴*۴ هرکدام یک mv دارد. سپس تفاوت را می گیریم transform می کنیم و کوانتایز می کنیم در گیرنده در اینورس کوانتایز

ضرب می کنیم. بعد از ان اینورس ترانسفورم می گیریم اگر پیکسل ها اینترا بود به اطرافیانش که از انها prediction گرفته بود اضافه می کنیم و اگر اینتر بود به MV اضافه می کنیم و پیکسل ها را بدست می‌آوریم.

دامنه کوانتایز کرومینانس کمتر از لومینانس است.

Deblocking filter: تصاویر در بیت ریت های پایین بلاکی می شوند علت آن تبدیلات می باشد. تصویر باید این بلاک ها را حذف کند. کل عملیات فیلتر در سه مرحله انجام می شود: ۱- قدرت فیلتر ۲- تصمیم فیلتر (روش اعمال فیلتر) ۴: پیاده سازی فیلتر.(چطوری فیلتر را بسازیم)



سعی می کنیم p0 و q0 را به هم نزدیک کنیم.

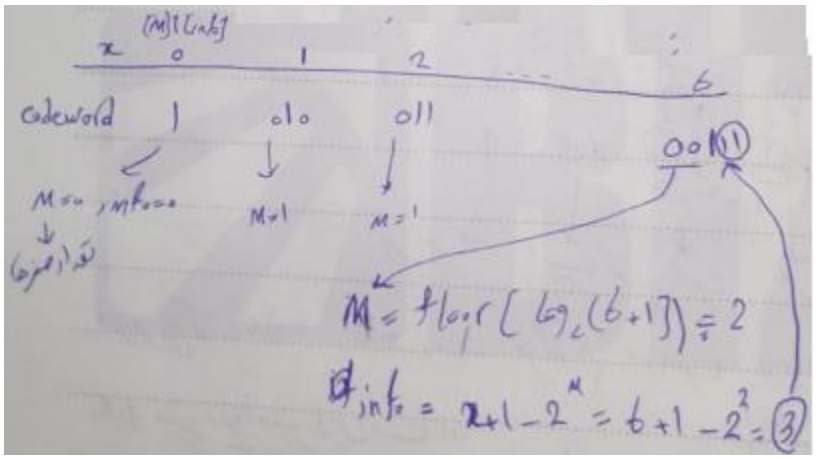
جلسه ۱۳۹۶/۹/۲۰

Entropy coding: سه روش برای Entropy coding دارد که دو روش آن content adaptive هستند که براساس محتوا احتمالات را تغییر می‌دهد. شامل CABAC, Huffman, Exponential Golomb

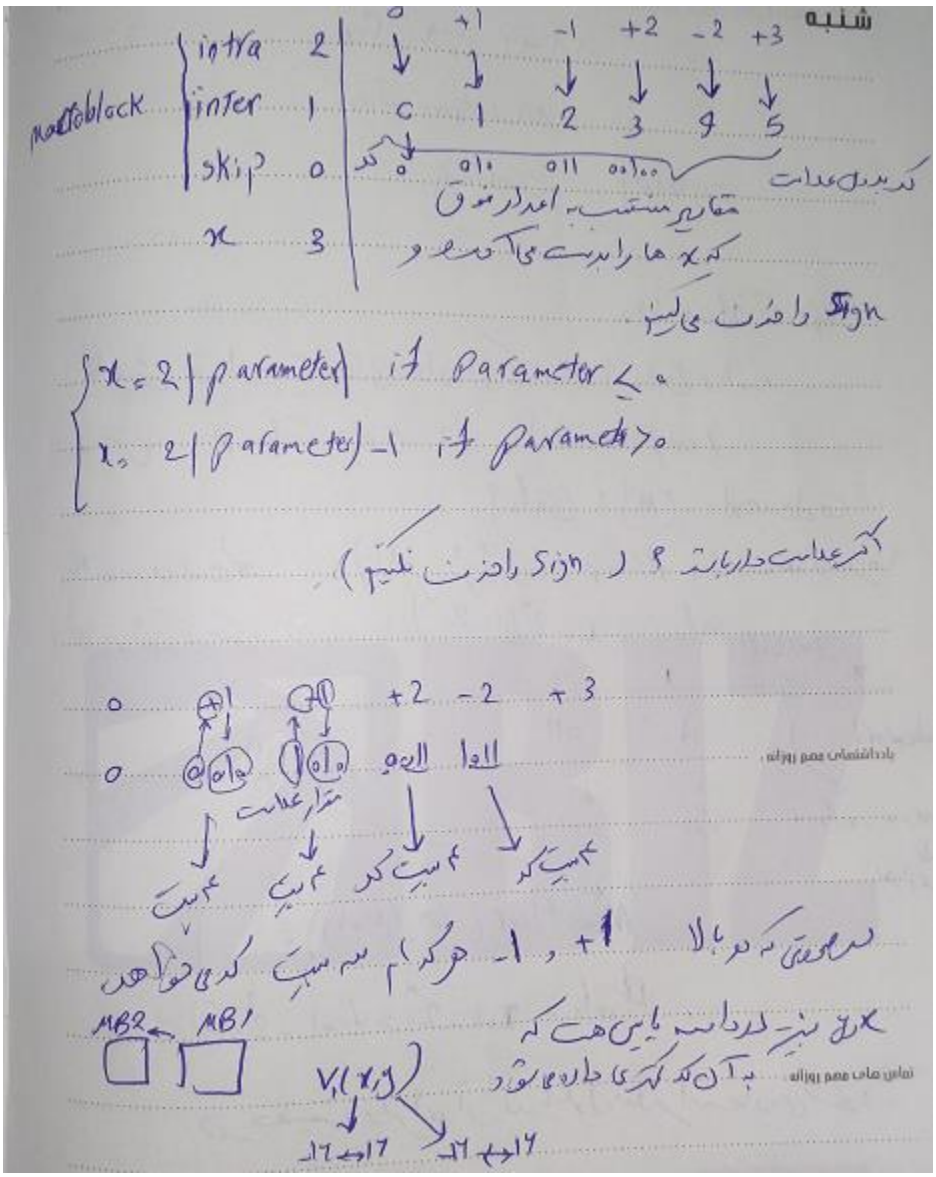
روش Exp Golomb: X عددی است که می خواهد کد شود. M =تعداد صفرها

$$codeword = [M]1[info], \quad M = floor\left(log_2\left[\frac{x}{2^k}+1\right]\right), \quad info = x + 2^k(1-2^M)$$

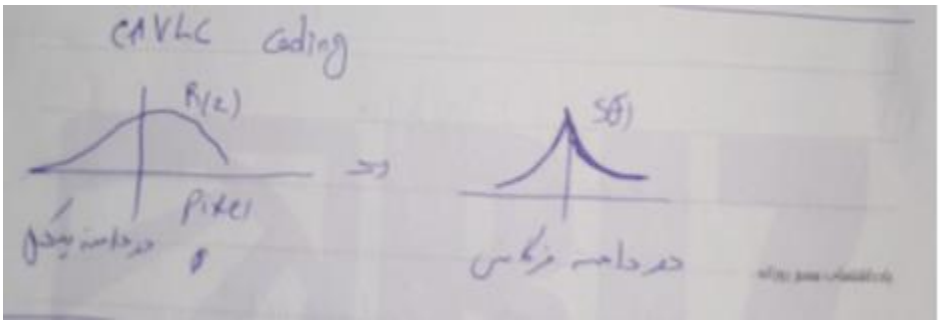
Symbol (x)	0	1	2	3	4	5	6	7	8	.....
Codeword	1	010	011	00100	00101	00110	00111	0001000	0001001	...



در حقیقت برای اعداد کد با طول متغیر استفاده می‌شود.



CAVLC: context Adaptive variable length code: برای اعداد یک lookup table داریم که تعیین می‌کند هر عدد چه کدی دارد. CAVLC از ۱۱ جدول استفاده می کند. انتخاب جدول بر اساس تعداد اعداد غیر صفر در رشته می باشد.



For Example

4x4 block:

0	3	-1	0
0	-1	1	0
1	0	0	0
0	0	0	0

- Scanned: 0,3,0,1,-1,-1,0,1,0……
- TotalCoeffs = 5 تعداد عدد غیر صفر
- TotalZeros = 3 تعداد صفرها تا آخرین عدد غیر صفر
- T1s = 3 تعداد یک ها از راست به چپ

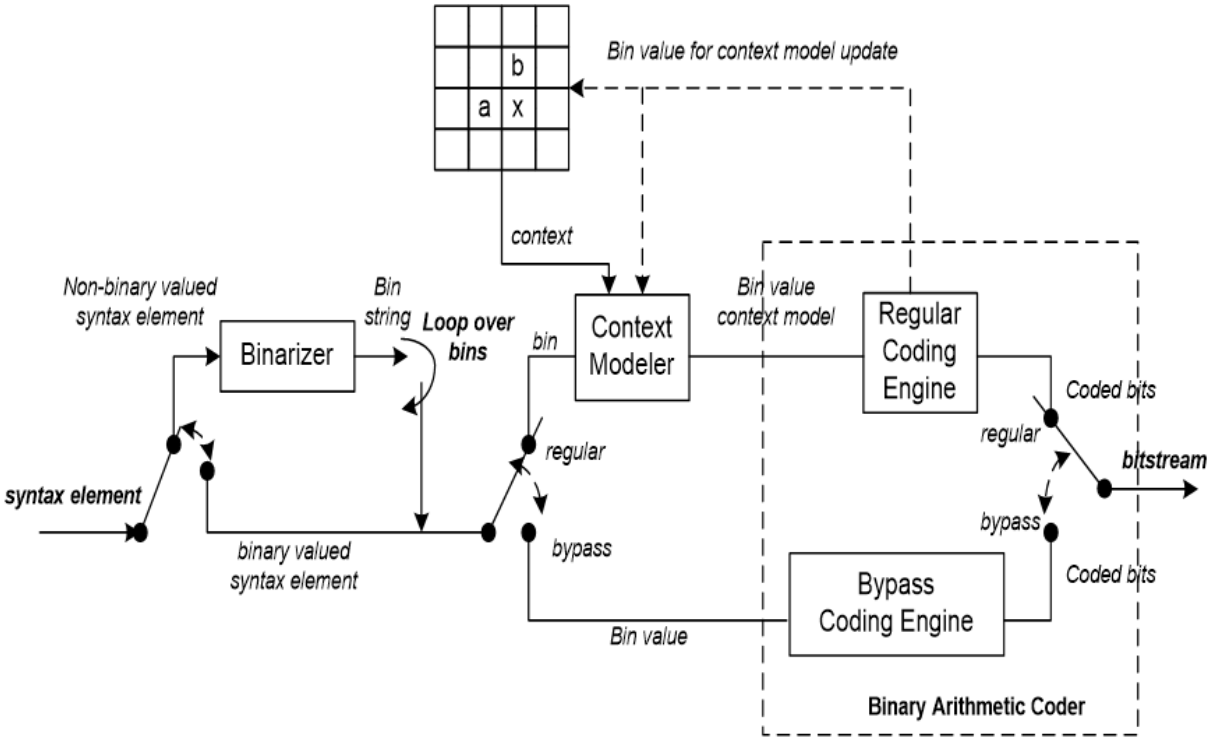
T1S تعداد یک ها از آخر را می گیرد که حداکثر ۳ می باشد. اگر تعداد یک ها بیشتر باشد با بقیه مثل عدد رفتار می‌شود و شمرده نمی شود.

Scanned: 0,3,0,1,-1,-1,0,1,0……

Element	Value	Code
coeff_token	TotalCoeffs=5, T1s=3	0000100
T1 sign (4)	+	0
T1 sign (3)	-	1
T1 sign (2)	-	1
Level (1)	+1 (use Level_VLC0)	1
Level (0)	+3 (use Level_VLC1)	0010
TotalZeros	3	111
run_before(4)	ZerosLeft=3; run_before=1	10
run_before(3)	ZerosLeft=2; run_before=0	1
run_before(2)	ZerosLeft=2; run_before=0	1
run_before(1)	ZerosLeft=2; run_before=1	01
run_before(0)	ZerosLeft=1; run_before=1	No code required; last coefficient.

در watermarking باید سعی کنیم در فرکانس های بالا کار کنیم. فرقش با H.261 این هست که در H.261 اسکن می کنیم و میگوییم هر عدد غیر صفر چند عدد صفر قبلش دارد سپس کد می کنیم. اما در اینجا می‌گوییم چند عدد غیر صفر داریم چند تا صفر داریم و از جدول آنها را پیدا می‌کند. در watermarking نباید بیت بالا برود. اگر بالا برود در حد معقول باشد.

CABAC: context adaptive Binary Arithmetic Coding: بلاک دیاگرام CABAC در زیر نشان داده شده است به سه بلوک اصلی: ۱- binarisation ۲- context modelling (احتمالی که به x می‌دهیم دائما در حال تغییر است). ۳- binary arithmetic coding. این روش حدود ۱۰ درصد بهتر از VLC است.



جلسه ۱۳۹۶/۹/۲۵

Rate distortion optimisation: نرخ بیت با distortion رابطه خطی دارد هرچه نرخ بیت زیاد شود distortion کم می‌شود.

Lagrangian optimisation technique

$$minimize\ D \qquad subject\ to\ R \le R_c$$

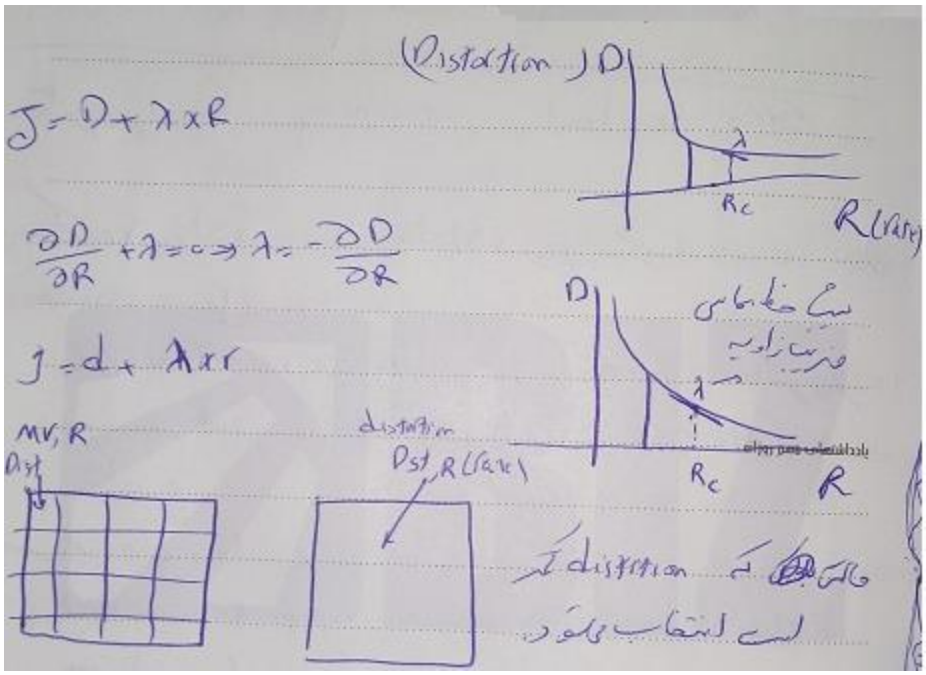
$$J = D + \lambda \times R$$

is simplified to minimising the cost of coding each MB separately:

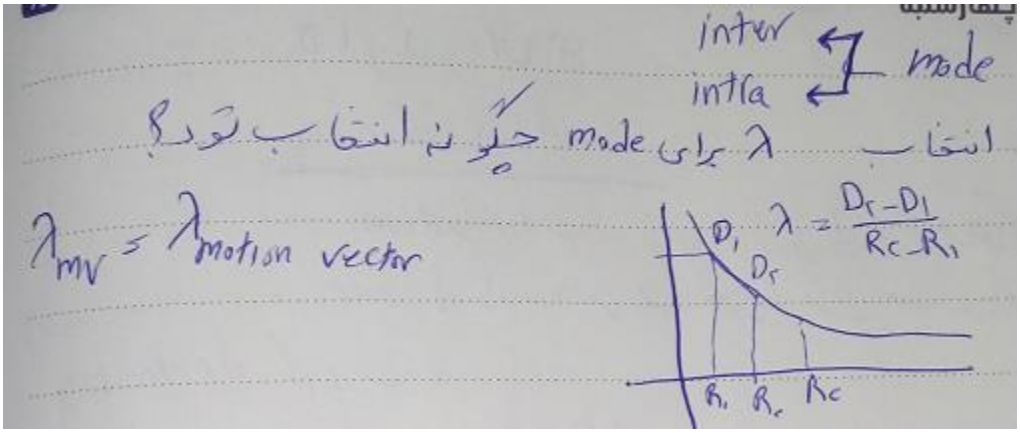
$$j = d + \lambda \times r$$

$$j_{mv} = SAD_{mv} + \lambda_{mv} \times r_{mv}$$

$$j_{mode} = SSD_{mode} + \lambda_{mode} \times r_{mode}$$



ما R یا ضریب زاویه را نداریم. موقعیکه ۴*۴ motion compensate کنیم distortion حداقل است. برای ۱۶*۱۶ بیشتر است. هر کدام از اینها یک مقدار distortion و rate دارد. کدام را انتخاب کنیم. حالتی که distortion کمتر انتخاب است. در H.261 برمبنای انرژی اینکار را انجام می دادیم چون processing power نداشتیم ولی در اینجا همه را محاسبه می کنیم. بهتر را انتخاب می کنیم. شکی نیست که موقعیکه بلاک ۴*۴ MC داریم distortion کمتری دارد ولی بعضی مواقع هست MV دقیق نیست یا جزئیات تصویر آنچنان هست که distortion، ۱۶*۱۶ با ۴*۴ مشابه هستند ولی rate ۱۶*۱۶ بیشتر است در این حالت ۱۶*۱۶ انتخاب می‌شود. حتی mode هم می توانیم انتخاب کنیم. که inter یا intra کنیم.

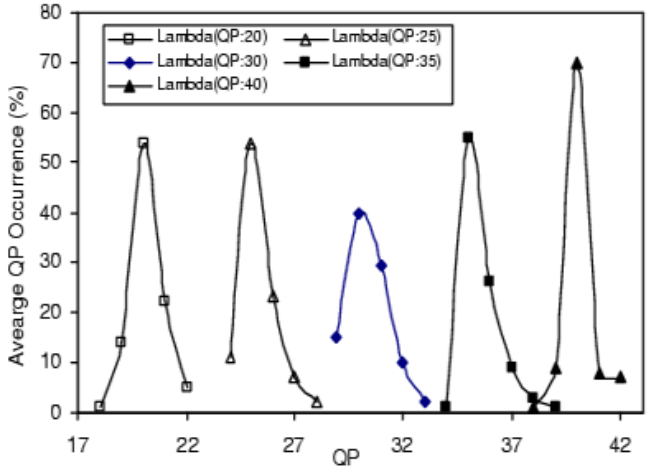


برای اعمال به H.264 باید اول λ را انتخاب کنیم. که شیب خط distortion نسبت به rate هست.

Selection of λ

In the above optimizations, the values of λ are empirically found to be

$$\lambda_{mode} = 0.85 \times 2^{(QP-12)/3} \qquad \lambda_{mv} = \sqrt{\lambda_{mode}}$$

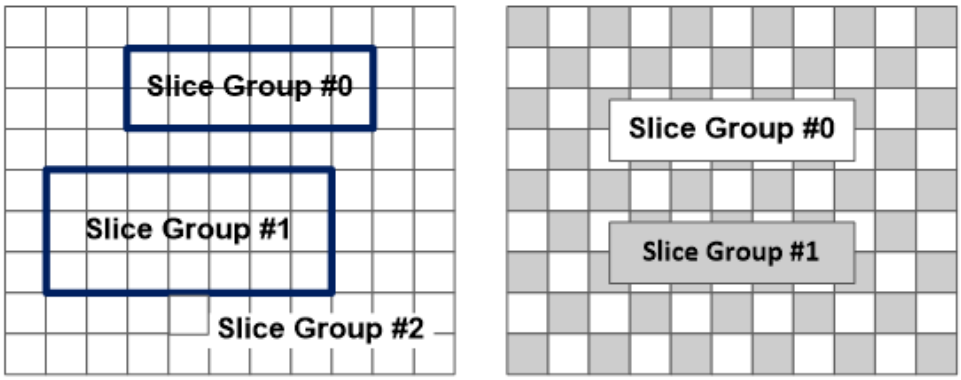


H.265 این بهینه سازی را خیلی بهتر انجام می دهد. و به آن اهمیت بیشتری داده است.

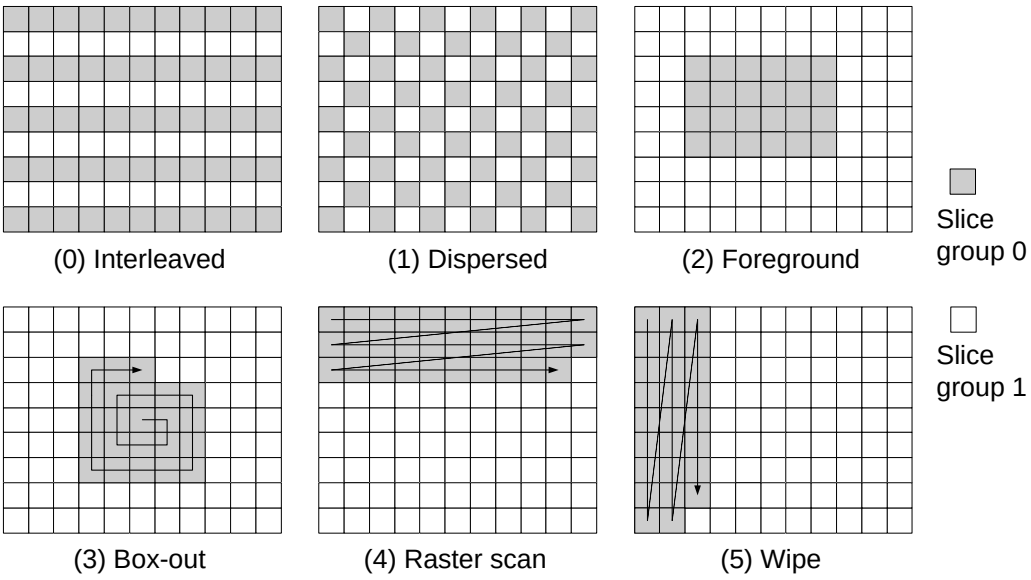
مقاوم در برابر خطا Error resilence encoding: در شکل زیر فرض کنید هر مربع یک ماکروبلاک باشد. در اینصورت می‌توان گروهی از ماکروبلاک ها را در Slice قرار می‌دهیم.

Error resilient encoding

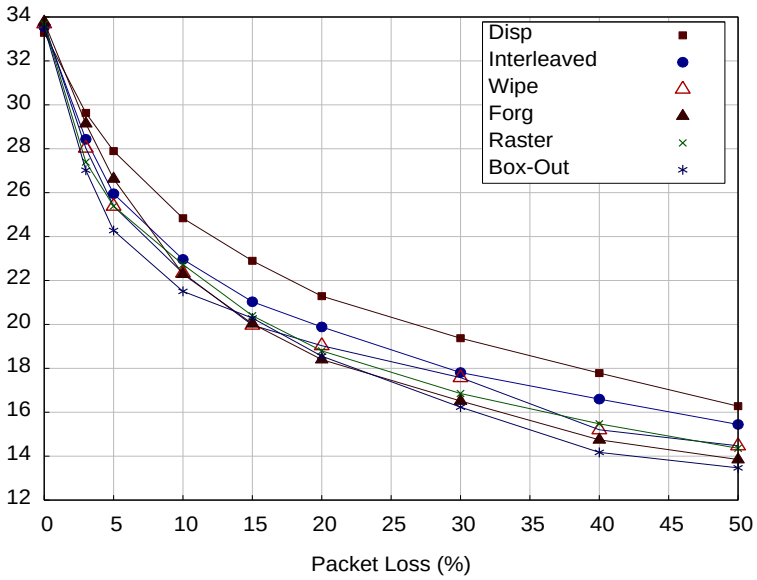
Flexible macroblock ordering (FMO)



گروه ۲ می‌تواند قسمت static تصویر باشد یعنی قسمتی که اطلاعات آن تغییر نکند یا در شکل سمت راست فوق دو گروه داریم، سفید Group0، خاکستری گروه ۱. می تواند در ارسال گروه ۰ در یک مسیر و گروه ۱ در مسیر دیگری ارسال شود. (مسیر فرکانسی، یا spatial یا زمانی). بهتر است در packet size، Sliceها در پکت مشابهی قرار گیرند. مسیر بسته ها می‌توانند متفاوت باشند بنابراین کیفیت گروه ۰ با یک فرق می کند. حالا اگر گروه صفر گم شد. از روی correlation می‌توان گروه دیگر را ساخت. H.264 شش گروه ضمنی از ماکروبلاک ها را تعریف می‌کند. که با شماره های ۰ تا ۵ نام گذاری می‌شوند مثل شکل زیر:



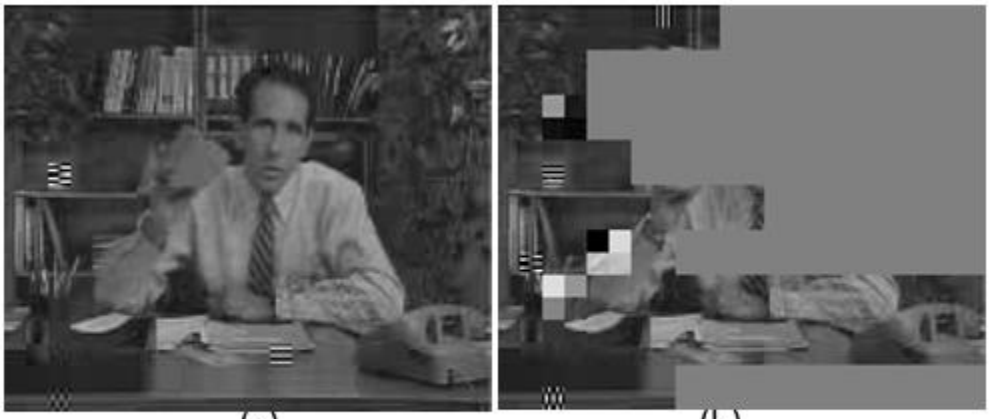
نوع inrerleaved, dispersed بهتر است. با گم شدن یک گروه می توان گروه دیگر را حدس زد. این می شود مقاوم در برابر خطا. Explicit pattern: از قبل مشخص می‌کنیم برای گیرنده یعنی خودمان تعیین می کنیم که بسته ها چه ماکروبلاک هایی دارند و نوع آن را برای گیرنده می‌فرستیم. که این سربار دارد ولی اگر برای مثلا ۱۰۰ فریم pattern مشابه باشد خوب هست. مقایسه شش روش ضمنی با PSNR بصورت زیر است که طبق آن Disp از همه بهتر است سپس interleaved.



Delay در Disp بیشتر است چون باید صبر کند که کل بلوک پردازش شود تا بسته های ۰ و ۱ ساخته شود. (پخش TV)، payload 188byte در MPEG2. هر بسته از یک header و یک payload تشکیل شده است.

Data partitioning: یعنی اطلاعات را گروپ بندی می کنم براساس اهمیت مثل آدرس. در ویدیو آدرس، نوع ماکروبلاک، MV مهم هستند که حدود ۱۰٪ یا ۸٪ نرخ بیت است. می توان اطلاعات مهم را با Forward error correcting محافظت کرد یا

می‌توان بخش مهم اطلاعات را Duplicate کرد. یعنی protect بخش‌های مهم که ۱۰٪ نرخ بیت را بالا می‌برد ولی بر error resilience تاثیر دارد، هزینه آن تاخیر (delay) است.



Effects of errors on (a) with and (b) without data partitioning

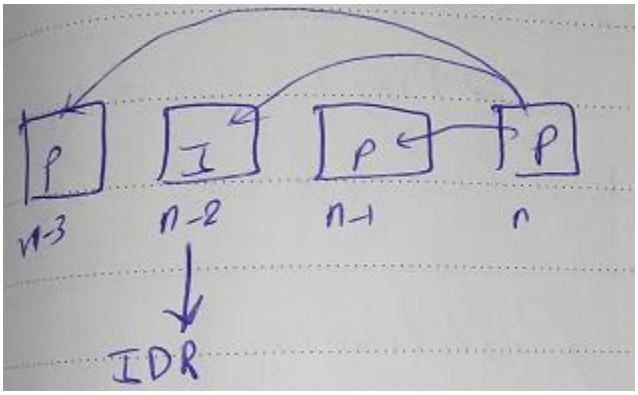
این سیستم از Mpeg2بود ولی در موبایل خیلی با ارزش است. در قبل آنها را به عنوان ضمیمه در نظر می گرفتند ولی الان جزو داده‌ها هست. در H.264 سه تا بخش داده داریم:

Data partition A (DP_A): موارد مهم هست. بهتر protect می‌شود.

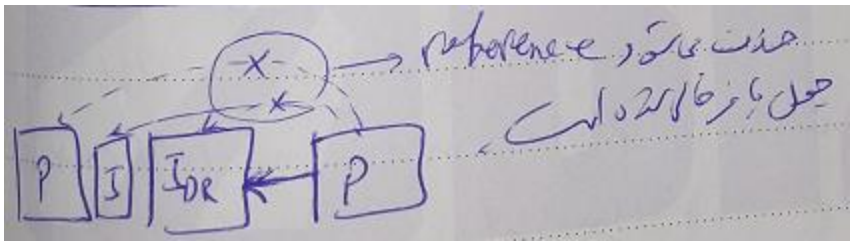
Data partition B (DP_B): بخش میانی داده که بصورت intra کد شده اند.

Data partition C (DP_C): بخش غیر مهم داده‌هاو شامل الگوی بلاک های کد شده به روش inter است.

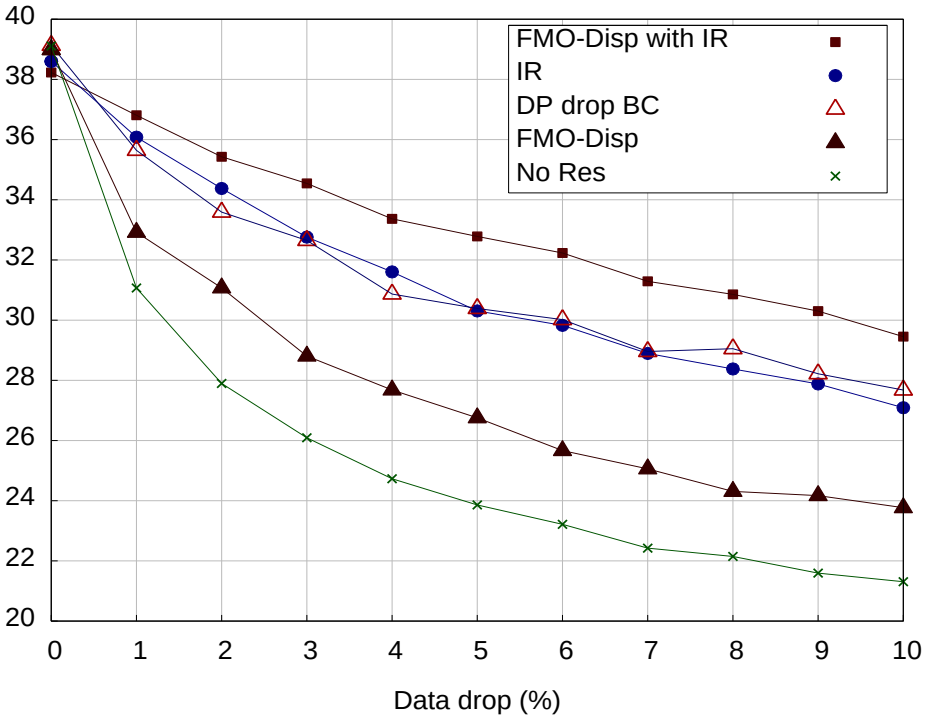
Intra MB/IDR: در H.264 دو نوع I داریم: ۱- I معمولی که intra کد شده است. ۲- IDR (instantaneous decoding refresh)



وقتی IDR می‌فرستیم بافر صفر می‌شود و reference ها صفر می‌شوند (مبارزه با propagation delay).



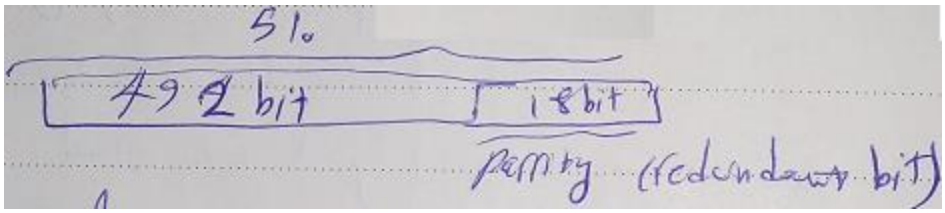
I که کد کنیم از آن به بعد خطا منتشر نمی شود ولی اگر بخواهیم خطای قبلی را هم حذف کند از IDR استفاده می‌کنیم. کارایی این روش IR=Intra refresh در نمودار زیر مشاهده می کنید که در بعضی حالات تا 11db بهبود داریم کیفیت در مخابرات 1db, 2db بهتر شود خیلی آورده خوبی است. IDR حالت خاصی از I است با این تفاوت که بافر خالی می‌شود به صفر.



در data partitioning در بدترین حالات ۲۰٪ نرخ بیت بالا می‌رود. (سه بار داده DP_A را تکرار کرده باشیم) ولی اضافه کردن فریم I نرخ بیت را خیلی بزرگ می‌کند (تا ۲۰۰٪). در ویدیو یک بسته گم شود چون بسته های دیگر به آن وابسته هستند خطا خیلی تاثیر بدی دارد. (FMO= گروه بندی و تعیین ترتیب بلاک ها) FMO همراه با Data partition خرج زیادی ندارد ولی کیفیت را 6db بهتر می‌کند.

Protection against error:

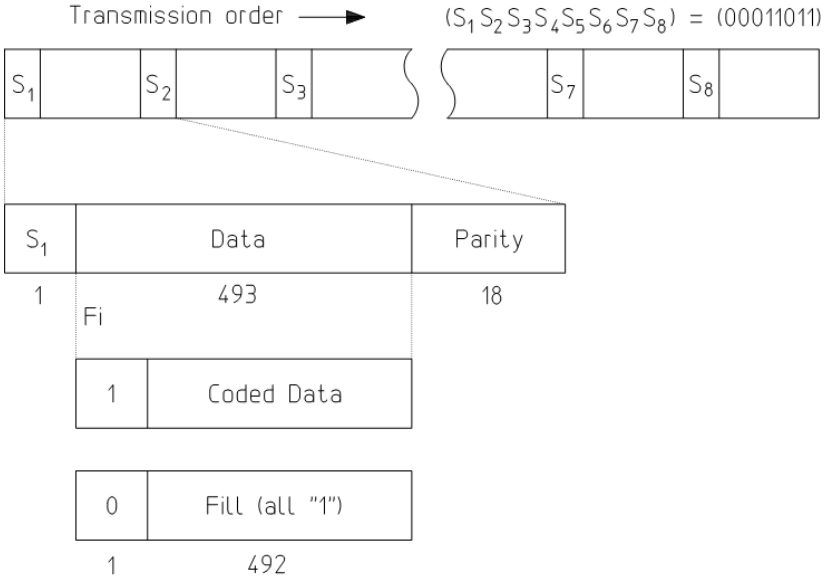
Froward error correction: در مخابرات Error با احتمال اتفاق می‌افتد. اگر احتمال خطای یک بیت p باشد احتمال خطای دو بیت p^2 است. بنابراین ارزش ندارد دو بیت اصلاح شود. Error detection در ویدیو مناسب است. از هر ۵۱۰ بیت ۱۸ بیت توازن است (redundant). در ویدیو detection از correction بهتر است.



در ویدیو بدانیم در خطا هست خیلی بهتر هست تا ندانیم. مثلا اگر MV در خطا باشد و بدانیم به جای آن صفر قرار دهیم اثر کمتری دارد تا اضافه شود. خطا را با CRC هم می توان تشخیص داد. (تقسیم و باقیمانده)

$$g(x) = (x^9 + x^4 + 1)(x^9 + x^6 + x^4 + x^3 + 1)$$

در شبکه packet network اگر هر اسلایس در یک بسته قرار گیرد و بسته گم شود کاری نمی توانیم انجام دهیم. کاری که می توانیم انجام دهیم interleave کردن بسته ها هست. که اگر یک بسته گم شد یک بیت اسلایس گم شود و بتوان آن را correct کرد.

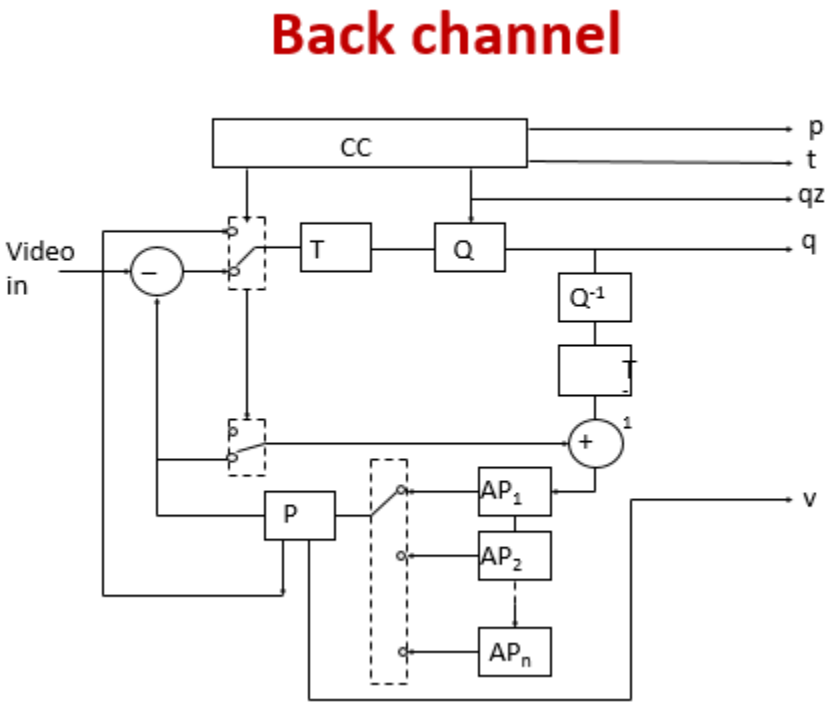


جلسه ۱۳۹۶/۹/۲۷

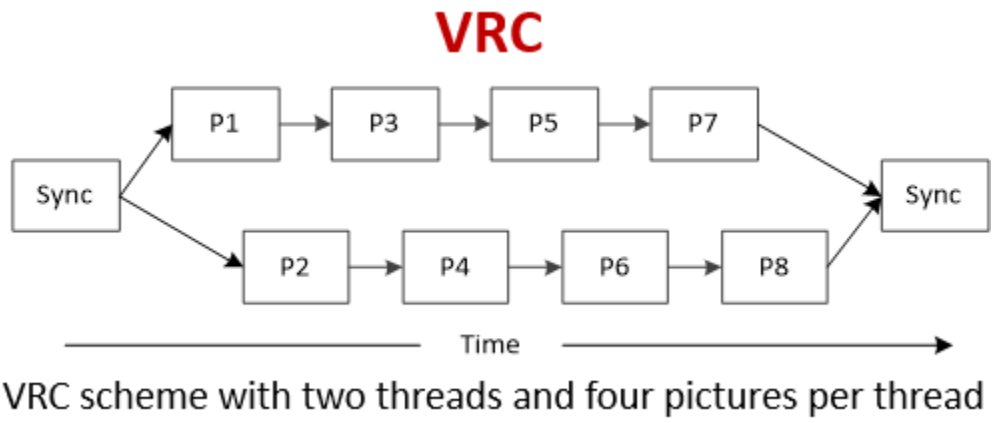
Ack/Nack: A positive acknowledgment (ACK) or a negative acknowledgment (NACK)

در ادامه راه حل های error resiliency در بعضی مواقع گیرنده به فرستنده یا دیکدر به انکدر دسترسی دارد که به آن Back channel می‌گویند. یکی از راه های خوب دریافت کردن این هست که هر بسته ای که فرستنده فرستاد گیرنده بگوید دریافت شد (ack) یا بد بود (nack). اگر بد بود اقدامی انجام شود. اگر اینکار را بکنیم فرستنده می داند داده‌هایش به گیرنده رسیده است یا خیر؟ در multiple reference در انکدر از چند تا رفرنس استفاده می کنیم و دیکدر هم همینطور دارد. و می تواند تا n رفرنس به عقب برود. اگر Ap2 یا فریم ۲ خطا داشت، nack دریافت شد یعنی فریم ۲ خطا داشته باشد در فریم های بعدی فرستنده از فریم ۲ استفاده نمی‌کند تا تفاوت آن منتشر نشود. یا گیرنده detect می کند و مثلا از فریم قبلی کپی می‌کند. این خوبی multiple reference است. اگر یک رفرنس بود مجبوریم intra کد کنیم و این بیت را بالا می برد. ولی در چند مرجع از آن

استفاده نمی شود و از فریم های قبلی استفاده می کند. فرض کنیم کانالی که استفاده می کنیم کانال خوبی باشد یعنی هرچیز بفرستیم با احتمال بالا دریافت می کنیم. اینکه برای همه دریافت شد بفرستیم باعث congestion می‌شود. در اینصورت از -ack nack استفاده می کنیم. مثلا اول با ۵ تا ack شروع می کنیم. تا زمانیکه خراب شد در اینصورت nack می فرستیم. این باعث کاهش بار شبکه در reverse می شود.



Video Redundancy Coding (VRC): در سیستم هایی که back channelندارد از VRC بجای ack/nack استفاده می کنیم. مثلا در streaming نداریم. در این روش ویدئوی ورودی به دو یا چند thread تقسیم می‌شود. فردها با هم کد شوند و زوج ها با هم.



این روش در wireless خیلی خوب هست. این روش robustness را بالا می برد ولی بیت را هم بد می کند چون فاصله بین فریم ها در thread ها زیاد می شود correlation کم می شود و بدتر کد می‌شود.

Redundant Slices: برای بالا بردن robustness در H.264 استفاده می شود. در یک بسته بخشی از اسلایس قبلی هم اضافه می‌شود. اگر برای یک اسلایس بسته خوب دریافت شود افزونه دور ریخته می شود ولی اگر خراب بود از قبلی استفاده می کند. این robustness را زیاد می کند ولی تا ۱۰٪ بیت زیاد می‌شود.

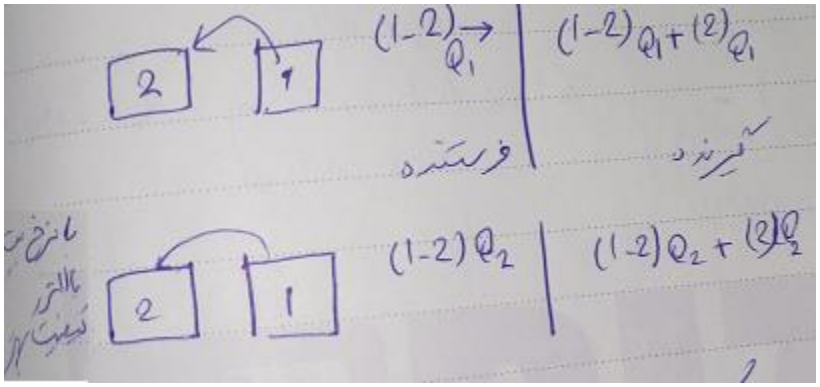
Stream switching:

PSP: **The primary switching predictive (PSP) pictures**

SSP: **secondary switching picture (SSP-picture)**

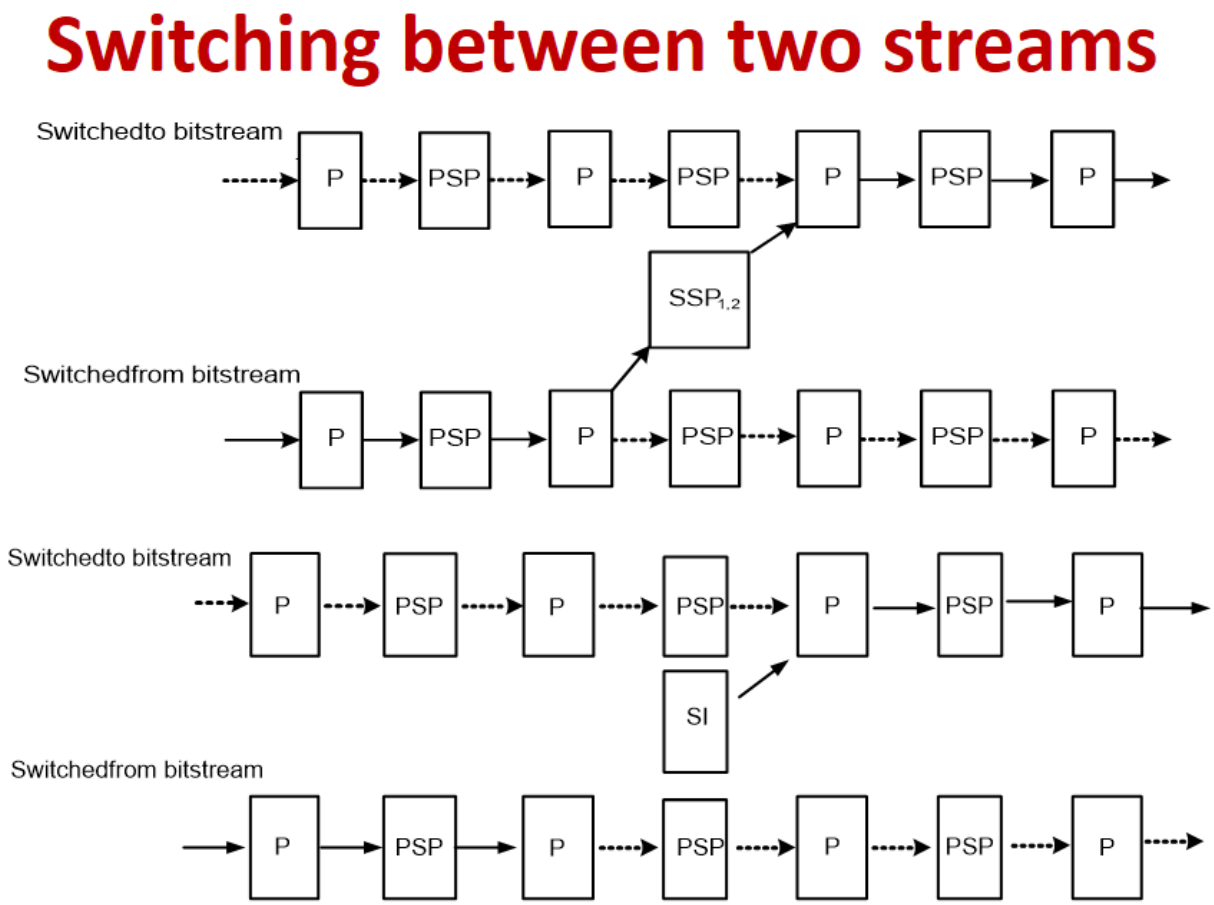
SI-picture: **SI-pictures are the intra coded representation of the SSP-pictures**

فرض کنید به اینترنت وصل هستید و از دو کانال با کیفیت های مختلف به صورت زیر استفاده می کنید

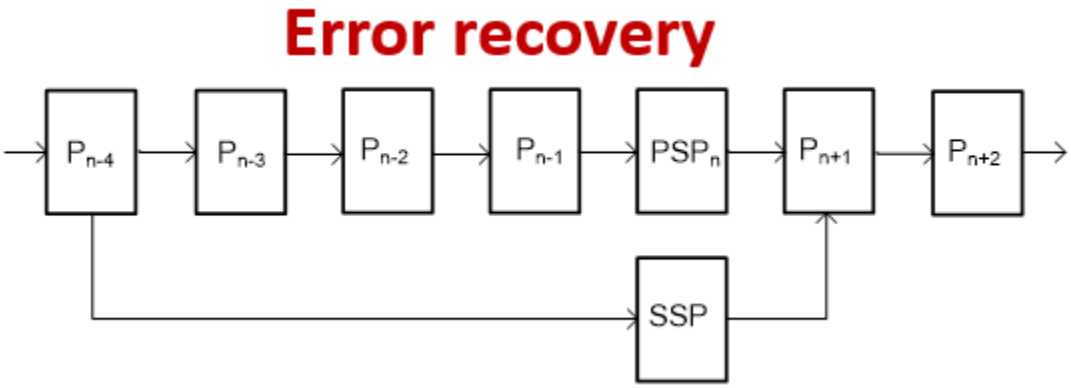


اگر بین دو نرخ سوییچ کنیم و از یک از Q1 به دو از Q2 دریافت شود در گیرنده نمی‌توان تصویر را با (1-2)Q1+(2)Q2 ساخت. راه حل SSP(1,2): 1→2 switchin picture را ارسال می‌کنیم.

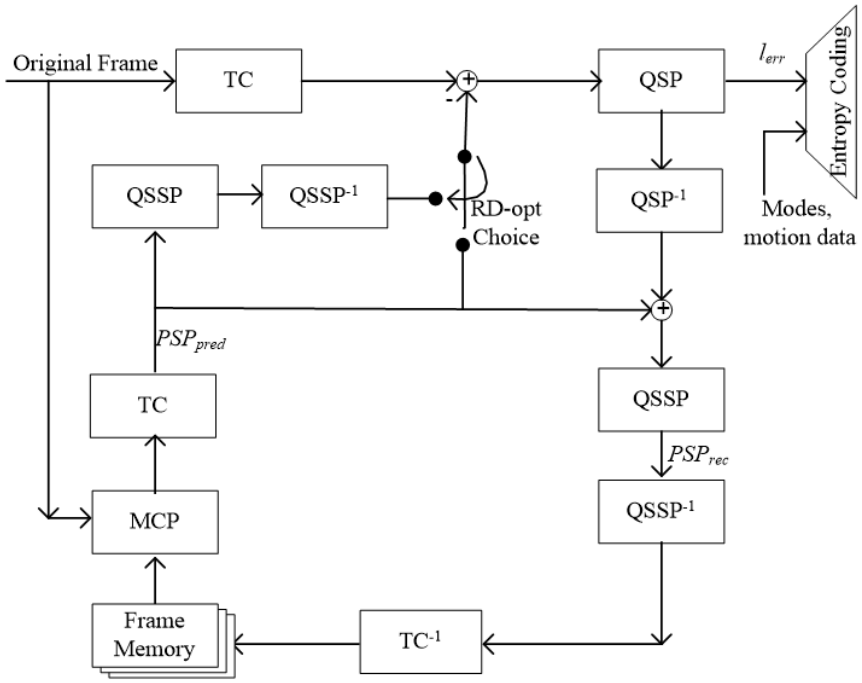
این در شکل زیر نشان داده شده است. در PSP ها سوییچ می‌کنیم. PSP، ۵٪ بیش از P است. این را در مقابل I درست کردند زیرا I از P خیلی بیشتر است. I جلو error propagation را می‌گیرد ولی از لحاظ شبکه خیلی بد است. چون بیت را زیاد می‌کند، خودش باعث خطا می شود. SSP(1,2) ممکن است از I بیشتر حجم داشته باشد ولی تعداد آن زیاد نیست. فقط در زمان سویچینگ داریم. ممکن است در دیدن یک استریم ۱۰ بار سوییچ کنیم یعنی ۱۰ بار SSP ارسال میشود در صورتی که برای I هر ۰.۵ ثانیه یک I ارسال می‌شود. از بالا به پایین هم وارد شویم باید SSP(2,1) را داشته باشیم. Bitrate استریم ها بین انکدر و دیکدر مشخص است. Transmission rate از حافظه گران تر است high bit rate و low bit rate را در حافظه ذخیره می‌کنیم. SI مثل Ipicture است. وارد استریم می‌شویم نه سوییچ بین استریم ها.



Error Recovery: این ایده برای بازیابی خطا خیلی خوب است. در شکل زیر SSP نسبت به P_{n-4} کد شده است. وقتی در P_{n-3}, P_{n-2}, P_{n-1} خطا رخ دهد انکدر خبردار می‌شود به جای PSP_n از SSP استفاده می‌شود (ارسال می‌شود). اما SSP شبیه I است و نرخ بیت بیشتری نسبت به PSP دارد.



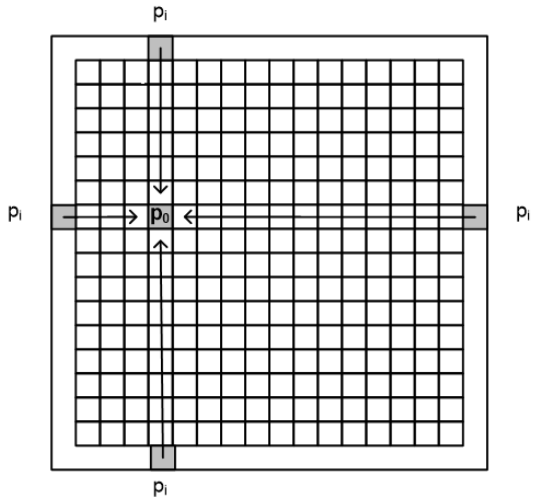
Encoding of switching pictures (PSP)



Error Concealment: با وجود همه روش های فوق برای error resiliency باز هم خطا اتفاق می‌افتد. در این حالت اثر خطا را کم می‌کنیم. یعنی خطا را از دید چشم مخفی می کنیم پیکسلی که گم شده را از پیکسل‌های بغلی اصلاح می‌کنیم. که به دو روش Intra frame concealment و Interframe error concealment می توان انجام داد.

Intraframe concealment:

P_0 می‌خواهد ساخته شود. میزان تاثیر P_0 از P_iها متناسب با عکس فاصله P_0 با آنها هست. P_i پیکسل‌ها در بلاک‌های مجاور است. این روش برای عکس‌ها خیلی خوب است. ولی در کدینگ ممکن است یک packet کم شود و تعدادی ماکروبلاک گم شود که بجای intra frame از interframe استفاده می‌شود.



Interframe error concealment: C0 گم شده است. از کجای فریم قبلی به جای C_0 کپی شود؟

۱-zero MV: کدک‌های ارزان کپی می‌کنند.

۲-Previous MV: می‌توان محاسبه کرد P_0 نسبت به بلاک قبلی اش چقدر تغییر داشته است. به احتمال زیاد C_0 هم نسبت به P_0 همان مقدار تغییر داشته باشد.

۳- mean MV: متوسط (1,2,3,4,5,6) گرفته شود که البته بدترین جواب را می‌دهد چون حرکت‌های ۱-۲-۳-۴-۵-۶ متفاوت هست و شما حرکت جدیدی درست می کنید که لزوماً حرکت درستی نیست.

$$y_0 = \frac{1}{6}\sum_{i=1}^{6} y_i \qquad x_0 = \frac{1}{6}\sum_{i=1}^{6} x_i$$

۴-Majority MV: اکثریت حرکت ها کدام طرفی است؟

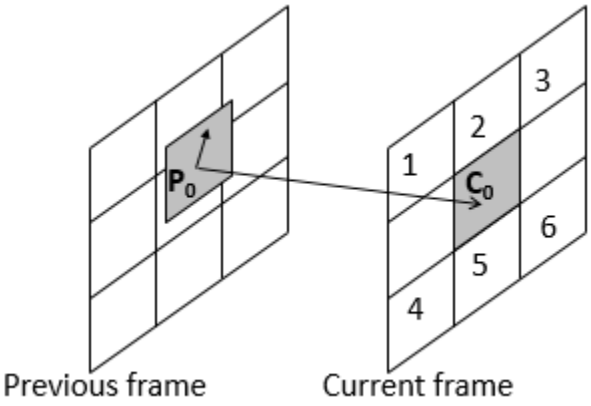
$$y_0 = \frac{1}{N}\sum_{i=1}^{N} y_i \quad x_0 = \frac{1}{N}\sum_{i=1}^{N} x_i$$

۶-top mv: از ۲ می‌گیرد.

۷-bottom MV: از ۵ می‌گیرد.

۸-Vector median mv: شش عدد فاصله بدست می‌آوریم و کوچکترین را پیدا می کنیم. یعنی در i=1، فاصله یک از بقیه محاسبه می شود در i=2 فاصله دو از بقیه و الی آخر

$$dist_j = \frac{1}{5}\sum_{i=1}^{6}\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad i \neq j$$



مقایسه روش ها در جدول زیر نشان داده شده است: No-error اصلی هست بدون خطا. همانطور که در جدول مشاهده می کنید median از همه بهتر است و zero از همه بدتر است.

*PSNR [dB] of various error concealment methods at **5** frames/s.*

Type	64Kbps, 5 fps, QCIF			
Sequence	Seq-1	Seq-2	Seq-3	Seq-4
Zero	**17.04**	**18.15**	**14.54**	**13.08**
Previous	**17.28**	**18.34**	**14.53**	**13.48**
Top	**19.27**	**21.08**	**17.04**	**16.25**
Average	**19.18**	**21.74**	**17.51**	**16.18**
Majority	**19.35**	**21.83**	**17.89**	**16.61**
Median	**19.87**	**22.52**	**18.29**	**16.89**
No Errors	**22.57**	**26.94**	**20.85**	**19.88**

در جدول فوق نرخ ارسال فریم 5 frame/s است یعنی فاصله بین فریم ها زیاد است. وقتی نرخ فریم را زیاد می‌کنیم به 12.5 frame/s چون سرعت زیاد می شود و فریم ها نزدیک می‌شوند نتایج بهتر می‌شوند. به صورت جدول زیر.

*PSNR [dB] of the various error concealment methods at **12.5** frames/s.*

Type	64Kbps, 12.5 fps, QCIF			
Sequence	Seq-1	Seq-2	Seq-3	Seq-4
Zero	**20.62**	**22.11**	**18.64**	**16.62**
Previous	**22.49**	**22.19**	**18.19**	**16.53**
Top	**22.97**	**25.16**	**20.97**	**20.04**
Average	**22.92**	**25.99**	**21.24**	**20.08**
Majority	**23.33**	**26.32**	**21.57**	**20.36**
Median	**24.36**	**26.72**	**22.16**	**20.81**
No Errors	**26.12**	**29.69**	**23.93**	**23.04**

در عکس زیر تفاوت استفاده بدون مخفی کردن خطا و با مخفی کردن خطا نشان داده شده است.

Visual Performance

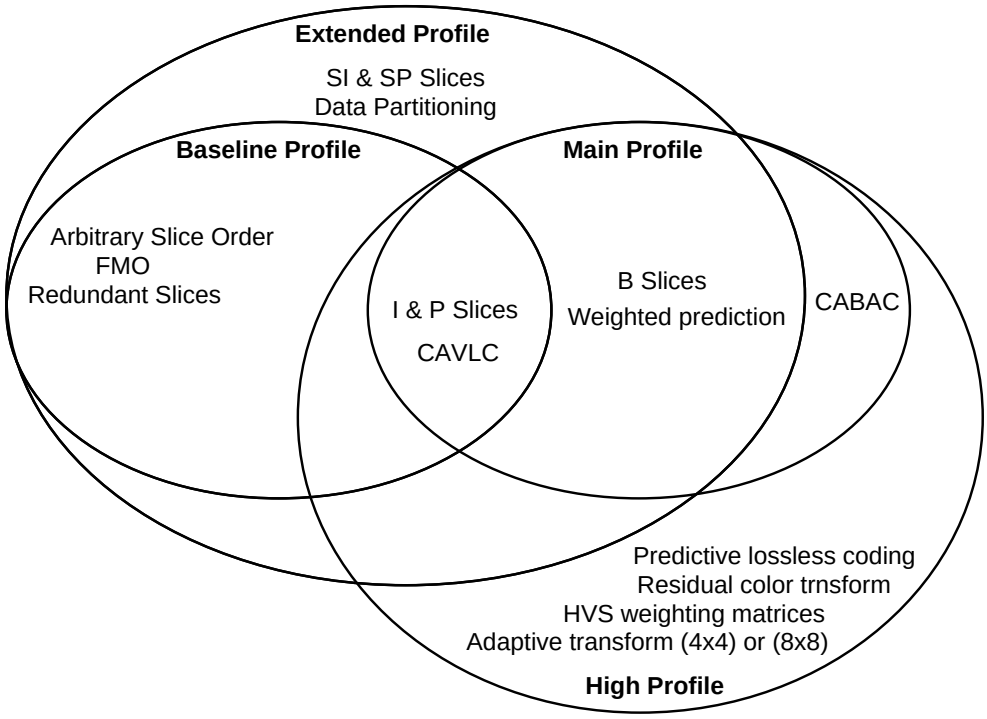
An erroneous picture along with its error concealed one

Profiles and levels: نمی توانیم کدکی بسازیم که بدرد همه چیز بخورد.

B-slice: end-to-end delay را زیاد می‌کند. بعضی کاربردها به B نیازی ندارند.

ساده ترین main profile هست. (broadcast)

Extended profile: based line +main بجز CABAC



Levels: بیشتر با نرخ بیت سروکار دارد:

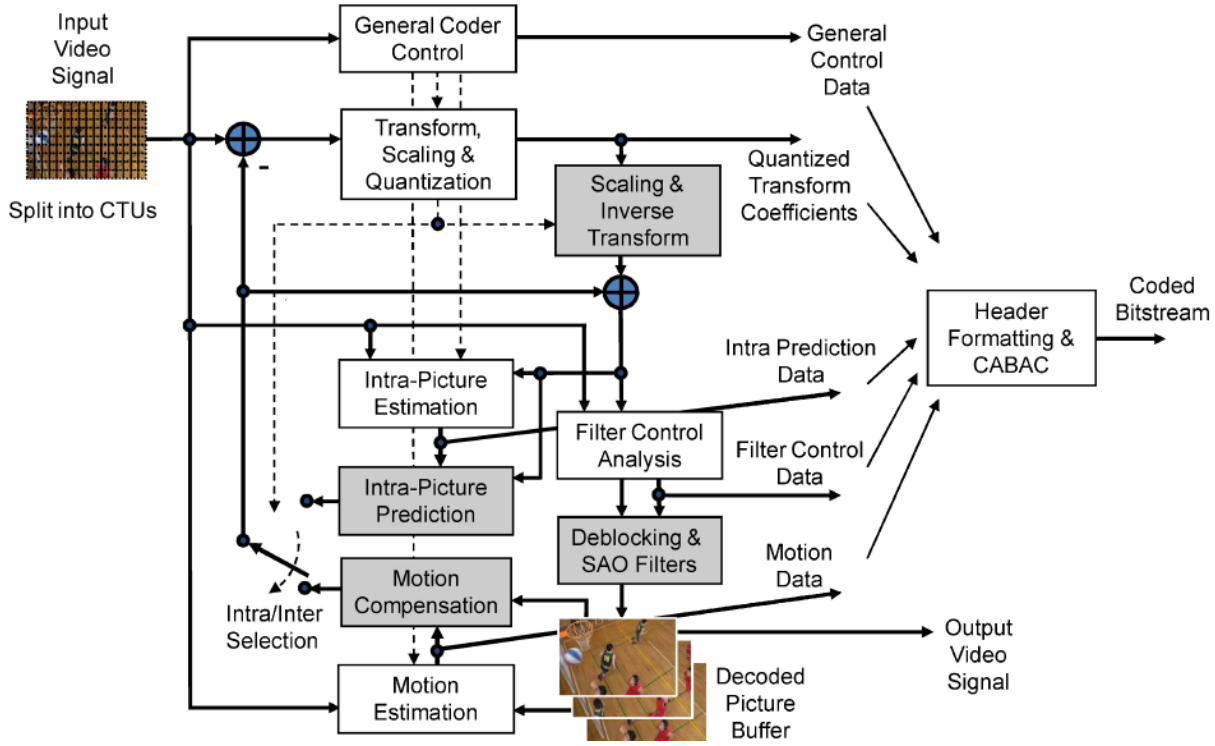
Level 1	**15 Hz QCIF@64kbit/s**	**Level 3**	**25 Hz 625SD@10Mbit/s**
Level 1b	15 Hz QCIF@128kbit/s	Level 3.1	30Hz 720p@20Mbit/s
Level 1.1	30 Hz QCIF@192kbit/s	Level 3.2	60Hz 720p@20Mbit/s
Level 1.2	15 Hz CIF@384kbit/s	**Level 4**	**30Hz 1080@20Mbit/s**
Level 1.3	30 Hz QCIF@768kbit/s	Level 4.1	30Hz 1080@50Mbit/s
Level 2	**30 Hz QCIF@2Mbit/s**	Level 4.2	60Hz 16VGA@135Mbit/s
Level 2.1	25 Hz 625HHR @4Mbit/s	**Level 5**	**30Hz 16VGA@135 Mbit/s**
Level 2.2	12.5 Hz 625SD@4Mbit/s	Level 5.1	30 Hz 4k×2k@240Mbit/s

Complexity of H.264: انکدر ۱۰ تا ۲۰ برابر پیچیده تر از دیکدر است. H.264 حدود ۴۰٪ نسبت به قبلی بهتر است. ۷۰٪ توان پردازشی صرف rate distortion optimization می‌شود. با MV حدود ۸۰ درصد می شود.

جلسه ۱۳۹۶/۱۰/۹

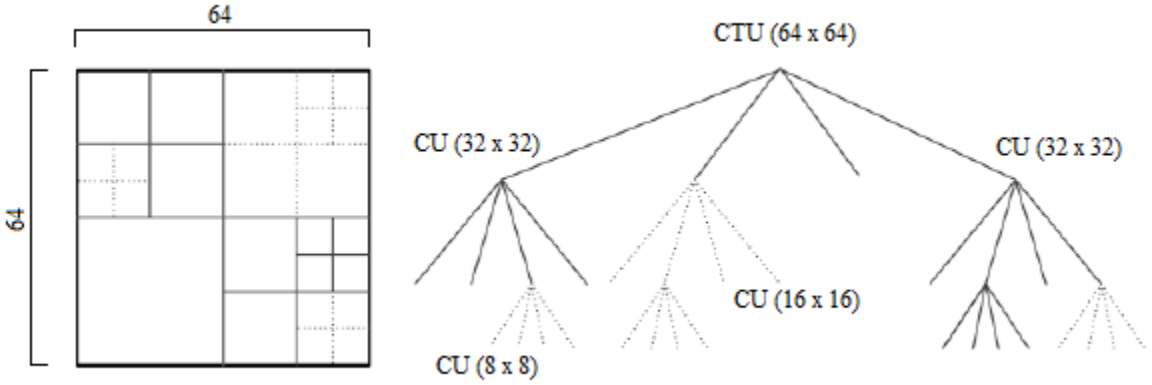
HEVC: High efficiency Video Coding

- از این کدک برای ابعاد ویدیو بزرگ (e.g. 4k x 2k or 8k x 4k) استفاده می‌شود. در ابعاد بزرگ correlation بیشتر است. بنابراین ساده تر می‌شود آن را فشرده کرد. بلاک ۱۶*۱۶ در ابعاد بزرگ مثل یک نقطه می‌شود. بنابراین ماکروبلاک را بزرگتر می‌گیرند به نام CTU: Coding Tree Units. H.265 بهتر از H.264 است و از Parallel processing استفاده می‌کند. امکان نمایش stereo یا multiview دارد. هدف بهبود بازدهی فشرده سازی هست، data loss resiliency و استفاده از پردازش موازی و استفاده از شبکه های packet based و circuit switch می باشد. بلاک دیاگرام انکدر بصورت زیر است:

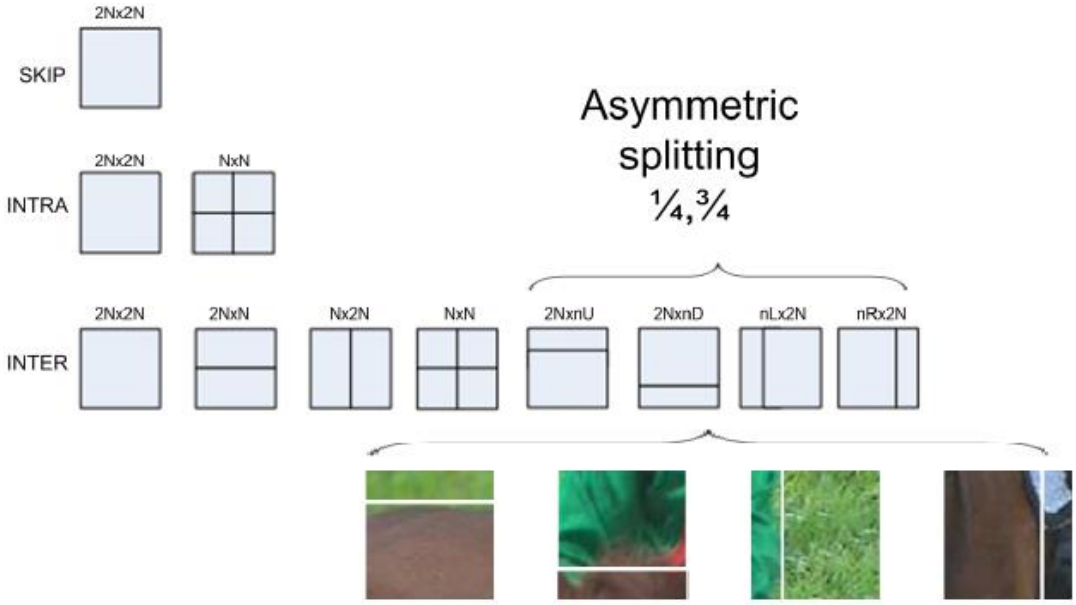


ویژگی‌های جدید: سیستم Interlace نداریم. Interlace روشی برای فشرده سازی آنالوگ بود. بصورت progressive است یعنی ۵۰ تصویر در ثانیه نفرستیم. اگر ۲۵ فریم بفرستیم و گیرنده هر کدام را دوبار تکرار کند اشکالی بوجود نمی‌آید. بجای ماکروبلاک CTU داریم. که ۶۴*۶۴ می‌باشد. از پردازش موازی استفاده می کند. اکثر ویژگی های H.264 را بهبود داده است.

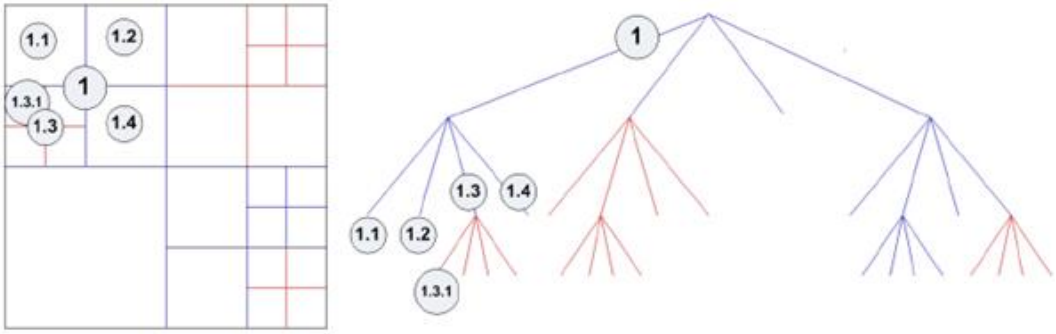
Coding Tree Units: ۶۴*۶۴ هست و بصورت Quad tree تقسیم می‌کنند. هر CU خود از سه بخش Prediction Unit (PU)، Transfomr Unit (TU) و Syntax elements تشکیل شده است. واحد بزرگتر برای بلاک برای اسکیپ سربار کمتری ایجاد می‌کند. در قبل بلاک برای ترانسفورم ۴*۴ و ۸*۸ هست ولی اینجا می تواند اندازه مختلفی داشته باشد



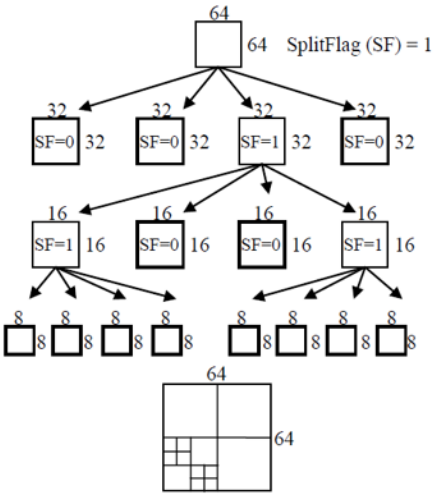
PU: **Prediction Unit**:

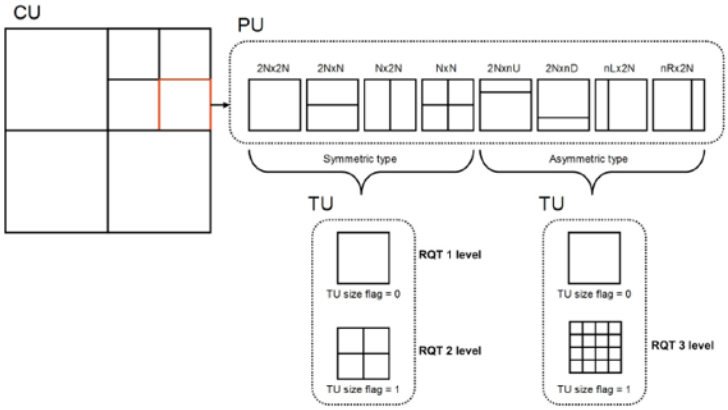


TU: **transformation Unit**: بزرگترین transformation ۳۲*۳۲ است. اگر حرکت uniform باشد بلاک بزرگتر انتخاب می‌شود وگرنه تقسیم می‌شود؟ تحقیق: قبل از تقسیم می‌توان گفت کدام بهتر است؟



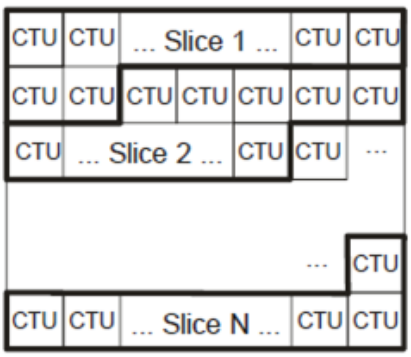
Example of CTU quadtree structure



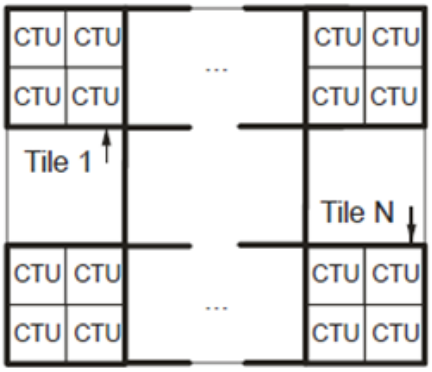


دلیل نیاز به پردازش موازی: ۱- تصویر بزرگ. ۲- هدف بالابردن compretion ratio. هم تعداد بلاک بیشتر است هم پردازش بیشتر.

Tiles: مثل Slice می‌شود. چند CTU را یک Tile می‌گوییم. می توانیم tile ها را مستقل از هم کد کنیم. که برای robustness مناسب هست. در دو شکل زیر دو روش مختلف برای ایجاد Tile داریم.

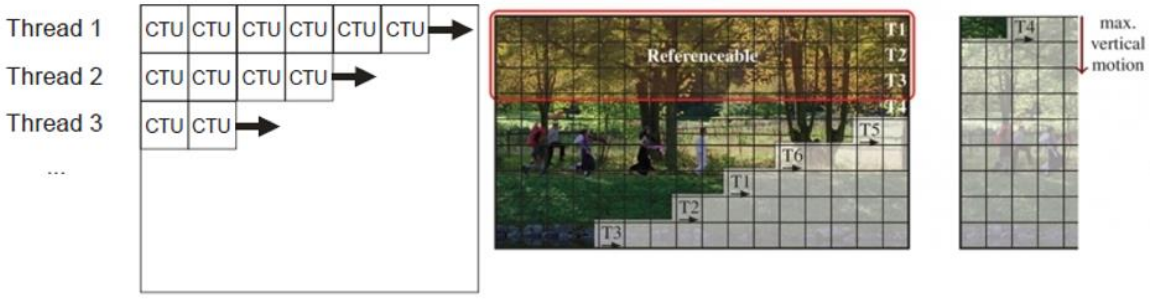


Sub-division of picture into slices



Sub-division of a picture into tiles

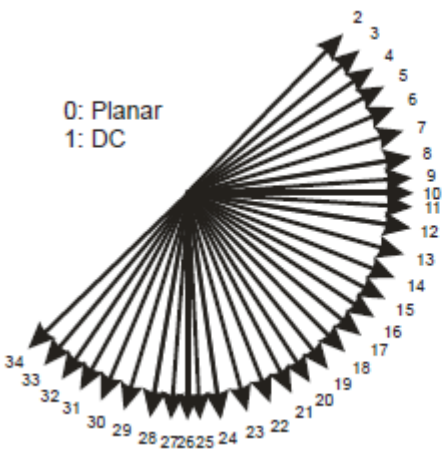
Wavefront parallel processing

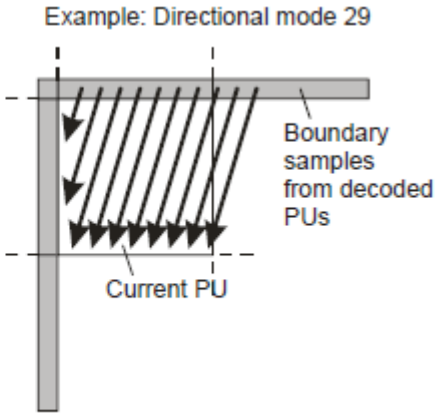


Sub-division of a picture into WPPs

اشکال پردازش فوق load balancing پردازنده ها می‌باشد.

Intra prediction: در H.264، ۹ حالت داشتیم و در اینجا ۳۵ حالت داریم برای prediction بصورت شکل زیر:



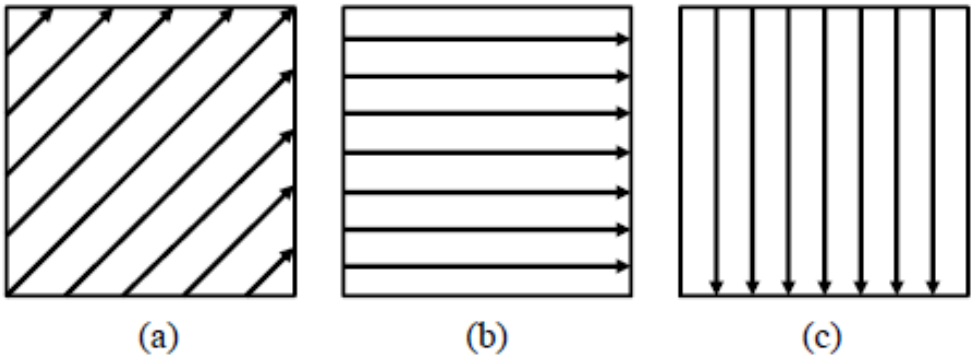


Entropy coding: در H.264 دو روش استفاده می شد اما در H.265 فقط از روش CABAC استفاده می‌شود چون حدود ۸ تا ۱۰ درصد بهتر است.

Fractional Sample Interpolation: در H.264 از فیلتر 6-tap استفاده می‌شود در اینجا از ۷ و 8tap استفاده می‌شود.

Transform: ۳۲*۳۲، ۱۶*۱۶، ۸*۸، ۴*۴ دارد.

Adaptive coefficient Scanning: سه روش scanning داریم چون Interlace نداریم.



Quantization: در کدک های قبلی ۳۲ سطح quantization داشتیم در H.264 ، ۵۲ سطح داریم در H.65 نیز کوانتایز مثل H.264 هست.

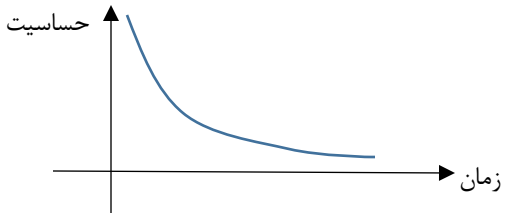
فیلتر برای رفع blockiness مثل H.264 می باشد اما نه به قدرت H.264. چون اندازه تصویر خیلی بزرگ هست به شدت تصاویر کوچک بلاکی نمی شود.

H.265 پیچیده تر هست و با وجود تصاویر ابعاد بزرگتر بازدهی بهتری دارد. حتی بهتر از تصاویر کوچک.

جلسه ۱۳۹۶/۱۰/۱۸

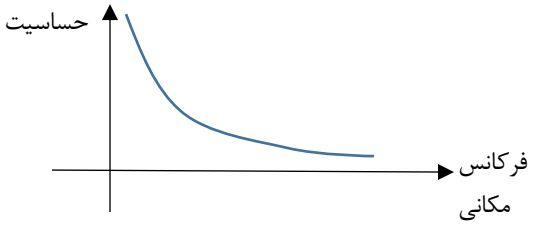
نمونه سوال ۱

۱-الف:با استفاده از رسم تقریبی، پاسخ فرکانسی چشم انسان به تغییرات زمانی اشیاء را توضیح دهید.



در حرکت سریع جزئیات را تشخیص نمی دهیم.

ب- با استفاده از رسم تقریبی، پاسخ فرکانسی چشم انسان به تغییرات مکانی اشیاء را توضیح دهید.



ج- از دو حالت فوق چگونه می توان در فشرده سازی ویدیو بهره برد

می‌توانیم quantiser بزرگتری در نقاطی که حساسیت نداریم استفاده کنیم.

۳-الف: رنگ‌های اصلی (اولیه) در دوربین‌های تلویزیونی و نمایش روی صفحه تلویزیون کدامند؟ چرا انتخاب شده‌اند؟

RGB از ترکیب آنها رنگ های دیگر ایجاد می‌شوند. بعضی را تکنولوژی اجازه نمی دهد.

ب: رنگ های اولیه برای ارسال و ذخیره سازی ویدیو کدامند؟ و چرا انتخاب شده‌اند؟

RGB خیلی correlate هستند آنها را decorrelate می‌کنیم و YCrCb بدست می‌آوریم.

ج:در پردازش ویدیو مثل تعیین بردار حرکت از کدام نوع رنگ اولیه الف یا ب استفاده می شود؟ چرا؟

از YCrCb چون بیشتر texture است که در Y جمع شده است.

۳-الف: در کدک های اولیه مثل h.261 انتخاب اینکه ماکرو بلاکی بصورت intraframe کد شود چگونه است؟

انرژی (تغییرات) بلاک را می‌گیریم نگاه می کنیم کدام کمتر است و آن را انتخاب می کنیم. DC مقدار، AC تغییرات نسبت به DC

ب: در این کدک، پیکسل‌های ماکروبلاک چگونه کد می‌شوند؟

اول به بلک های ۸*۸ تقسیم می‌کنیم سپس DCT می‌گیریم و هرکدام را کد می کنیم.

ج: این عمل در H.264/AVC چگونه انجام می‌شود؟

به بلاک‌های ۴*۴ تقسیم می‌کنیم و از بلاک های کنار هم prediction می‌گیریم که ۹ عدد prediction داریم مثل horizontal, Diagonal, …, Planar

د: این عمل در کدک h.265/HEVC چگونه کد می‌شود و چه فرقی با H.264/AVC دارد؟

بلاک های ۳۲*۳۲ است تعداد جهات ۳۵ حالت می‌باشد.

۴-الف: چگونه از فشرده سازی بخش پذیری scalable coding برای ارسال مقاوم ویدیو در مقیاس خطای کانال استفاده می‌شود؟ از سطوح مختلف استفاده می‌شود سطح مهمتر را با forward error correcting و مقاومتر ارسال می‌کنیم.

ب: در فشرده سازی از نوع multiple description coding ارسال مقاوم ویدیو در مقابل خطای کانال چگونه انجام می‌شود؟ یک کانال فریم های زوج و یک کانال فریم های فرد را می‌فرستیم. اگر یک کانال بد بود و کانال دیگر خوب بود و ۳ خراب شد می‌توان از روی ۲و۴ آن را ساخت.

ج: در چه روشی از کانال انتقال روش الف و ب ترجیح داده می‌شود و در چه شرایطی روش ب ارجحتر است؟

در الف اگر کانال خوبی برای ارسال پایه داشته باشیم در ب وقتی کانال‌ها یکی بر دیگری ارجح نیست ب بهتر است.

۵-الف: کد کردن انتروپی کدینگ در کدک های ویدیو یعنی چه؟ با احتمالات سروکار داریم داده‌ای که احتمال بیشتری دارد بیت کمتری دارد.

ب: این نوع کدک در استاندارد h.264/AVC چگونه تعریف شده است؟ CABAC, HAFMAN کدهای تطبیقی هستند از ۱۱ جدول استفاده می‌شود.

د: مضرات ناشی از entropy coding چیست؟ و چگونه می‌توان از آن مقابله کرد؟ خوبی آن فشرده سازی است. چون طول متغیر است اگر یکی در خطا باشد نمی‌توان دیگری را Detect کرد؟ با slice کردن جلو انتشار خطا را می‌گیریم. در انتهای slice، reset می‌کنیم.

۶-الف: تعریف ماکروبلاک در کدک های استاندارد به چه منظوری است و مشخصات آن چیست؟ کوچکترین واحد کد کردن را تعیین می‌کند. ابعاد ۸*۸، ۴*۴. بشتر فرمت تصویر 4:2:0 است بهتر است توانی از ۲ باشد.

ب: این تعریف در کدک H.265/HEVC چگونه بیان می‌شود؟ ماکروبلاک را بزرگتر می‌گیریم ۶۴*۶۴ چون تصویر خیلی بزرگ است با کیفیت بالا.

ج: قدرت فشرده سازی کدک H.265/HEVC هم در ویدیوی با رزلوشن بالا (HDTV) و هم ویدیو با ابعاد کوچک (حتی QCIF) بیش از قدرت فشرده سازی H.264/AVC است. دلیل آن چه می تواند باشد؟ ۱-CBAC استفاده می‌شود که از hafman بهتر است. ۲- حتی در تصاویر کوچک بلاک های بزرگتر باعث فشرده سازی بیشتر می‌شود.

نمونه سوال ۲:

۱-الف:کیفیت سیگنال‌های کوانتایز شده با حداکثر دامنه A و دامنه کوانتایز Q را بصورت $PSNR=10\ log_{10}(A^2/(s^2))$ تعریف می‌کنند. اگر در کوانتایزر خطی متوسط $s^2=Q^2/12$ و دامنه A به ۸ بیت بصورت خطی کوانتایز شودف کیفیت سیگنال کوانتایز شده چقدر است؟

$PSNR=10\ log(A^2/(Q^2/12))=10\ log(12A^2/(A/2^8)^2)=10\ log(12A^2/(A^2/2^{16})=10\ log\ (3*2^{18})=180*0.3+10*0.5=59$

$Log2=0.3$, $log\ 3=0.5$

ب: اگر دامنه A به ۱۰ بیت کوانتایز می‌شد، این کیفیت چقدر بود؟

$10log(3*2^{22})=5+66=71db$

ج: هر بیت اضافی چقدر کیفیت را بهبود می‌بخشد؟

(71-59)/2=6db

۲-الف: منظور از اندازه گیری کیفیت (objective quality) که با کیفیت عینی (subjective Quality) سازگار باشد را بیان کند.

Objective: اندازه گیری می‌کنیم. Subjective: آنچه چشم می‌بیند. اگر بخواهد همخوانی داشته باشد باید طوری objectiveرا اندازه بگیریم که در جایی که چشم نمی بیند تاثیر نداشته باشد.

ب- سه روش برای اندازه گیزی کیفیت که تشکیلات VQEQ آنها را پیشنهاد داده است که می‌توانند با کیفیت عینی سازگار باشند را به اختصار شرح دهید.

Full Reference: تصویر اصلی را داریم با تصویر پروسس شده مقایسه می کنیم

Reduced Reference: یکسری پارامتر از تصویر اصلی و پروسس شده بدست می آوریم با هم مقایسه کنیم.

No Reference: تصویر مرجعی برای مقایسه نداریم و باید distortion هایی مثل blockiness, blurriness را بدست آوریم و مقایسه کنیم

ج-مزایا و معایب هر یک از روش‌های فوق در چیست؟

Full reference: تصویر اصلی را نداریم – مزایا دقت بشتر

Reduced دقت به full referenceنزدیک هست ولی باید قبل از ارسال پارامترهای را بدست آوریم و از طریق side channel بفرستیم.

No reference: دقیق نیست ولی سربار ندارد به راحتی در داخل شبکه می‌توان از ان استفدده کرد.

۳-الف: در کدک ها ضرایب تبدیل دو بعدی بین قابی را بعد از کوانتایز کردن برای فشرده سازی بهتر ابتدا جاروب zig-zag کرده و بعد آنها را کد می‌کنند چرا؟ برای کاهش سیمبل ها استفاده می‌شود (تعریف دوبعدی) زودتر رسیدن به فرکانس های بالا و اینکه صفرها به انتها منتقل شوند.

ب: در عمل فوق نوع جاروب ضرایب برای تصاویر ویدئویی بهم بافته (interlaced) و پیشرفته (progressive) می‌تواند متفاوت باشد چرا؟ در progressive خطوط به هم نزدیک هستند zig-zag استفاده می‌شود و در interlaced دور هستند نوع جاروب فرق می کند. در جهت عمودی حرکت کنیم زودتر به اعداد بزرگتر می رسیم.

ج: در کدک H.264/AVC ضرایب تبدیل کوانتایز شده با فرکانس های بالا بعد از جاروب بنحو دیگری از دیگر کدک ها کد می‌شوند. دلیل این کار چیست؟ zig-zag scanning آن بازدهی بیشتری از H.261 دارد.

۴-الف: در کد پیشرفته مثل H.264/AVC تعیین اینکه یک ماکرو بلاک بصورت درون قابی (intraframe) و یا بصورت بین قابی با جبران حرکت (Interframe) کد شود چگونه انجام می‌شود شرح دهید؟ هم interframe کد میکنیم هم inter بهتر را استفاده می‌کنیم. در قبلی ها processing power نداشتیم

ب: در کدک های قبل از این کدم این تصمیم گیری چگونه انجام می‌شد آنرا شرح دهید؟ فقط انرژی را نگاه می کردیم و کمتر را ارسال می کنیم.

ج: این دو روش تصمیم گیری چه مزایا و چه معایبی نسبت به یکدیگر دارند شرح دهید. مزایا h.264 دقیق تر هست ولی processing power بیشتری می‌خواهد.

۵:الف: درس داده نشد.

۶-الف: در کدک H.264/AVC ماکرو بلوک های بین قابی (interframe) با چند مرجع (multiple reference) کد می‌شوند، این عمل چه مزایایی دارد؟ اگر یکی خراب شد دیگری را میتوان ساخت. قدرت فشرده سازی هم بالا می‌رود.

ب: برای ماکروبلاک های نوع P، ماکرو بلاک‌های مرجع چگونه انتخاب می‌شوند؟ از فریم های قبلی (یک لیست داریم به نام لیست صفر)

ج: برای ماکر بلاک های نوع B، ماکرو بلوک های مرجع چگونه انتخاب می‌شوند. در نوع B دو تا لیست داریم به نام صفر و یک

د:می‌توان ادعا کرد که در کدک H.264/AVC ماکروبلاک های نوع P در مقایسه با ماکرو بلاک های نوع B بهتر از سایر کدک ها (مثل MPEG2) کد می‌شوند چرا؟ در multiple reference چند انتخاب داریم و انتخاب بهتر انجام می‌شود.

۷:الف در کدک h.264/AVC ماکرو بلاک های درون قابی ۱۶*۱۶ (16-intra) از تبدیل hadamard استفاده می کنیم چرا؟

تمام ۱۶*۱۶ در یک جهت prediction گرفته می شود که ۴ حالت بیشتر ندارند. و هدمارد گرفتن انرژی را پایین می‌آورد.

ب: این تبدیل چگونه اعمال می‌شود، شرح دهید. ۴*۴ DCT می گیریم و بین Dcها هدمارد می گیریم.

ج: در فرمت 4:2:0 این تبدیل برای بلوک های روشنایی و رنگ چگونه تعریف می‌شوند (ابعاد تبدیل مد نظر هست)

۱۶ تا DCT داریم

د: آیا از این تبدیل برای بلوک های درون قابی ۴*۴ (4-intra) هم می‌شود استفاده کرد؟ چرا؟ ۴*۴ در ۹ جهت prediction می گیریم. شباهت ندارند بنابراین هدمارد تاثیری ندارد.