باسمه تعالی

سیستم‌های چندرسانه‌ای (۳۴۲-۴۰)
دانشکده مهندسی کامپیوتر
ترم پاییز ۱۳۸۵
دکتر حمیدرضا ربیعی
تکلیف شماره ٤: فشرده سازی تصویر

۱- مقدمه

یک تصویر دیجیتال به دست آمده با نمونه‌برداری و کوانتیزاسیون یک عکس معمولی، فضای ذخیره‌سازی زیادی را نیاز دارد. به عنوان مثال، یک تصویر رنگی ۵۱۲×۵۱۲ که رنگ هر پیکسل را با ٢٤ بیت نشان می‌دهد، ٧٦٨ کیلوبایت از حجم یک دیسک را اشغال می کند و عکسی با اندازه دو برابر عکس قبلی بر روی یک دیسکت نرم جا نمی‌شود. برای انتقال چنین تصویری با یک مودم kbps ۲۸/۸ به حدود ٤ دقیقه زمان نیاز است. هدف از فشرده سازی تصویر کاهش میزان داده‌های مورد نیاز برای نشان دادن تصاویر دیجیتال و بنابراین کاهش هزینه انتقال و ذخیره سازی می‌باشد. فشرده‌سازی تصویر نقش کلیدی در بسیاری از کاربردهای مهم داراست. این کاربردها شامل پایگاه داده تصویر، مخابره تصویر، سنجش از راه دور (استفاده از تصاویر ماهواره ای برای کاربردهای آب و هوا یا منابع زیرزمینی)، تصویربرداری اسناد، تصویر برداری پزشکی، انتقال توسط دورنگار، کنترل وسایل نقلیه نظامی و فضایی از راه دور و کنترل زباله‌های خطرناک کاربرد دارد. به طور خلاصه، کاربردهای روزافزون پردازش تصویر، به دستکاری، ذخیره سازی و انتقال تصاویر دودویی، سیاه سفید یا رنگی به شکلی کارا و مؤثر بستگی دارد.

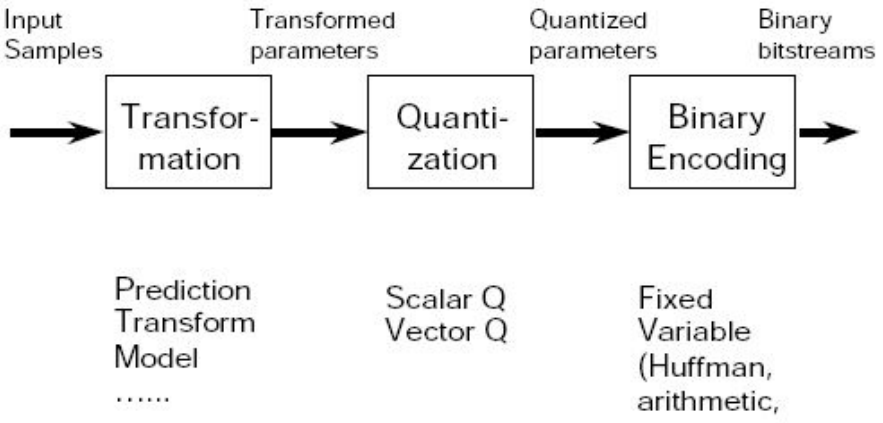
یکی از مهمترین پیشرفت‌ها در فشرده‌سازی تصویر، تأسیس استاندارد JPEG برای فشرده‌سازی تصاویر رنگی بود. با استفاده از روش JPEG، یک تصویر رنگی ٢٤ پیکسل/بیت می‌تواند به ۱ تا ۲ پیکسل/بیت تقلیل پیدا کند. چنین کاهشی، ذخیره سازی و انتقال تصاویر ماهواره‌ای را با هزینه‌ای معقول ممکن می‌سازد، بدون اینکه آرتیفکت‌های آشکار قابل مشاهده ایجاد کند. این کار همچنین امکان ذخیره سازی یک عکس رنگی را از روی اینترنت در زمانی حدود یک لحظه فراهم می کند و بنابراین انتشار و تبلیغات بر روی شبکه جهانی اینترنت به یک واقعیت تبدیل می‌شود. قبل از استاندارد JPEG، استانداردهای G3 و G4 برای فشرده سازی اسناد دورنگار، کاهش زمان انتقال یک صفحه متنی از حدود ٦ دقیقه به یک دقیقه را ممکن ساخته بود.

در این آزمایش، اصول فشرده سازی تصویر برای تصاویر دودویی، سیاه سفید و رنگی معرفی خواهد شد. فشرده سازی ویدیو در آزمایش بعدی بررسی خواهد شد.

۲- تئوری ها و تکنیک های فشرده‌سازی تصویر

به طور کلی، روش های کدینگ می توانند به دو دسته بدون تلفات و با تلفات تقسیم بندی شوند. در کدینگ بدون تلفات، مقادیر نمونه های اصلی دوباره دقیقاً به دست می آیند و فشرده‌سازی بوسیله پیدا کردن زواید آماری سیگنال انجام می شود. در کدینگ با تلفات، سیگنال اصلی تا حدودی تغییر می کند ولی به نرخ فشرده سازی بالاتری دست می یابیم.

پردازش به کار گرفته شده در سیستم کدینگ با تلفات در شکل ۱ نشان داده شده است. می‌بینیم که سیگنال در ابتدا به میدان تبدیل می رود که برای فشرده سازی مناسب تر است. سپس پارامترهای تبدیل کوانتیزه می‌شوند. در انتها، پارامترهای کوانتیزه شده به صورت بدون تلفات به بیت های دودویی کد می‌شوند. در کدینگ بدون تلفات، مرحله کوانتیزاسیون انجام نمی‌شود.



شکل۱: بلوک دیاگرم یک سیستم کدینگ معمولی

۲-۱- کدینگ بدون تلفات

۲-۱-۱- کدینگ بدون تلفات

در کدینگ با طول متغیر (VLC) سمبول هایی که احتمال حضورشان بیشتر است با بیت های کمتری نمایش داده می شوند (با استفاده از یک کلمه کد کوتاهتر). قضیه اطلاعات شانون [۱] بیان می کند که طول میانگین برای هر سمبول، l، توسط آنتروپی منبع، H، محدود می شود:

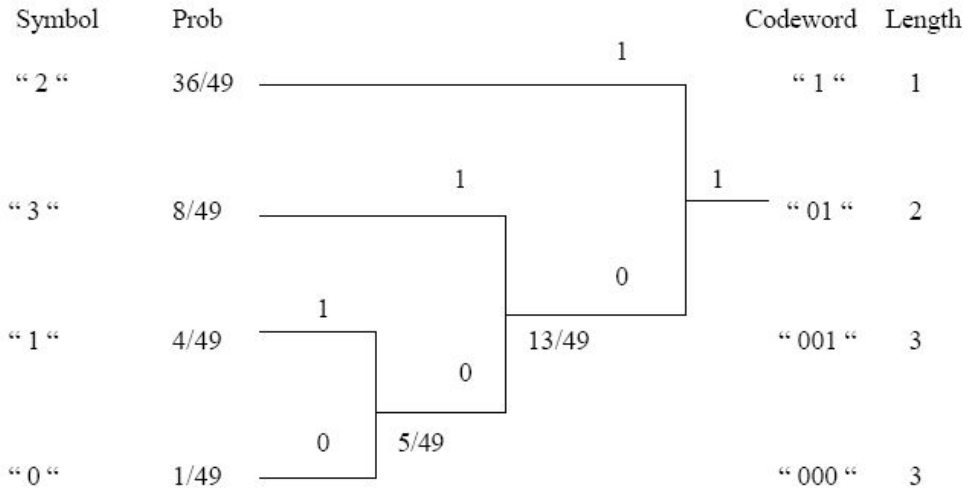
$$H=-\sum p_n \log_2 p_n \le l=\sum p_n l_n \le -\sum p_n (\log_2 p_n+1)=H+1$$

p_n احتمال n-امین سمبول، l_n طول کلمه کد n و l طول میانگین کلمه کد است. H آنتروپی منبع است که اطلاعات میانگین موجود در یک منبع تصادفی را نشان می دهد.

۲-۱-۲- کدینگ هافمن

قضیه شانون تنها محدوده را می دهد ولی راه عملی تشکیل کد برای رسیدن به محدوده را نمی‌دهد. چنین کاری با روشی که به نام کدینگ هافمن شناخته شده است صورت می گیرد.

مثال: تصویری را که به ٤ سطح کوانتیزه شده است در نظر بگیرید: ۰ ، ۱ ، ۲ و ۳ ، فرض کنید احتمال این سطوح به ترتیب ۱/۴۹، ۴/۴۹، ۳۶/۴۹ و ۸/۴۹ باشد. طرح کد هافمن در شکل ۲ نشان داده شده است.



شکل۲: مثالی از کدینگ هافمن

در این مثال، داریم:

طول متوسط

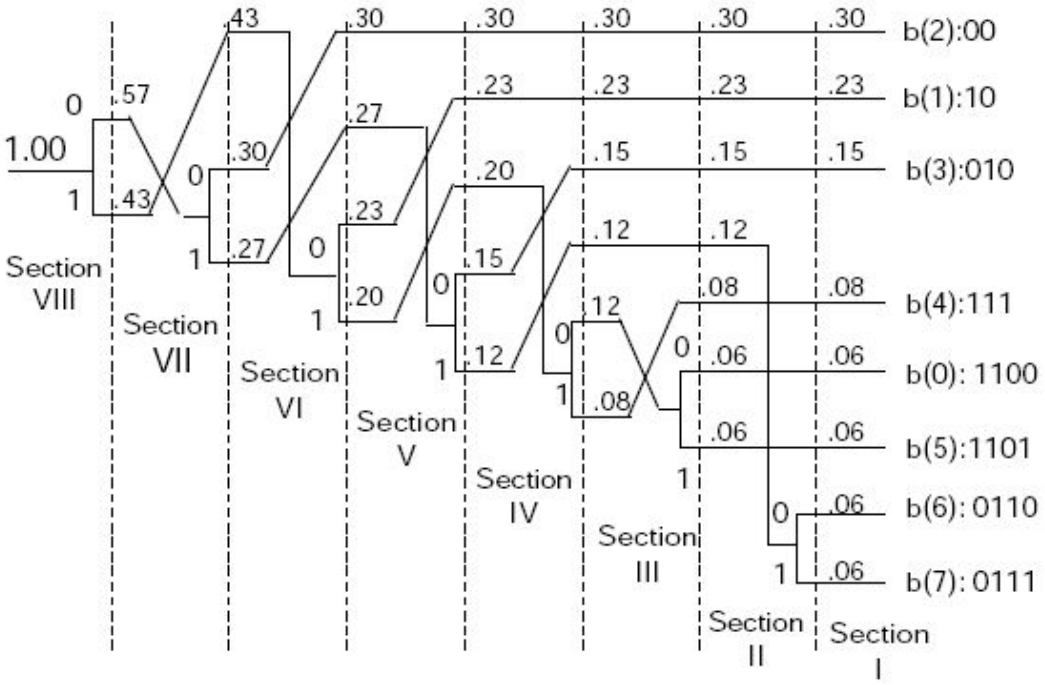
$$l = \frac{36}{49}\cdot 1 + \frac{8}{49}\cdot 2 + (\frac{4}{49}+\frac{1}{49})\cdot 3 = \frac{67}{49} = 1.4$$

آنتروپی منبع

$$H = -\sum p_k \log p_k = 1.16.$$

$$H < l < H+1$$

شکل ۳ مثال دیگری ازکدینگ هافمن را نشان می‌دهد:



$$\text{Average Bit Rate} = \sum_n p_n l_n = 2.71 \text{ bits}; \quad \text{Entropy} = \sum_n p_n \log_2 p_n = 2.68 \text{ bits}.$$

شکل۳: مثال دیگری از کدینگ هافمن

۲-۱-۳- روش های دیگر کدینگ با طول متغیر

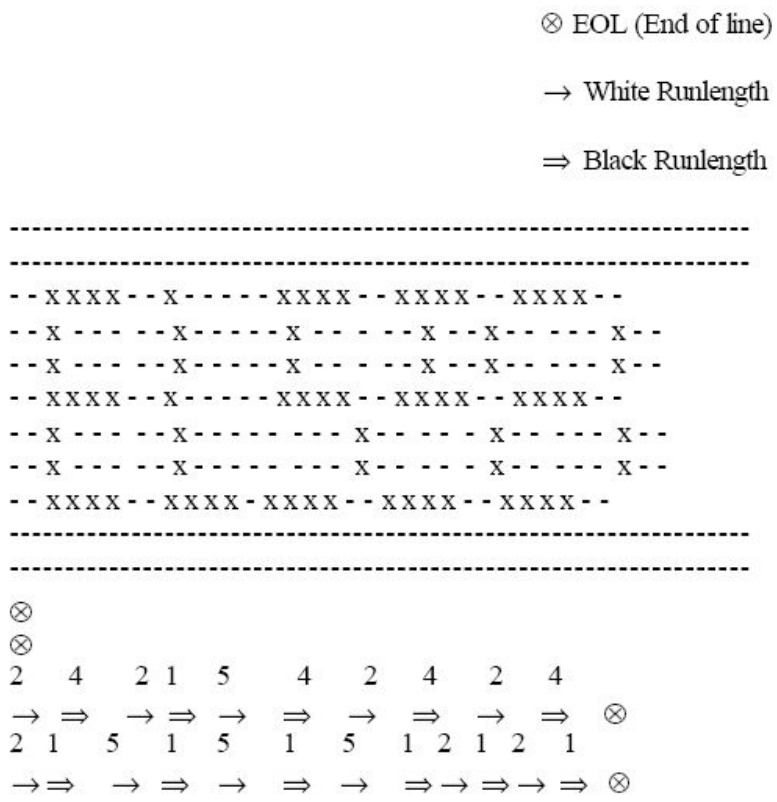
کدینگ LZW (Lempel, Ziv, Welsh)[۲] الگوریتم مورد استفاده در بسیاری از نرم افزارهای عمومی فشرده سازی اطلاعات مانند gzip و pkzip بوده است. یکی از مشهورترین فرمت های فایل های گرافیکی (GIF) هم از طرح کدینگ LZW استفاده می کند.

روش شناخته شده دیگر، روش کدینگ ریاضی [۲] است که از کدینگ LZW و هافمن قدرتمند تر است ولی محاسبات بیشتری نیاز دارد.

۲-۱-۴- کدینگ RLC تصاویر دوسطحی

در کدینگ RLC یک بعدی تصاویر دوسطحی، پیکسل ها از چپ به راست در هر خط اسکن، اسکن می شوند. فرض کنید که هر خط، همیشه با پیکسل های سفید، شروع و خاتمه می یابد و در هر اسکن، به نوبت تعداد بیت های پیکسل های سفید و سیاه شمرده

می شود و آخرین قسمت پیکسل های سفید با یک سمبول مخصوص "EOL" (انتهای خط) جایگزین می شود. طول قطعات سفید و سیاه با کتاب – کدهای مجزا کد می شود. کتاب – کد، مثلاً برای قطعات سفید توسط روش کدینگ هافمن طراحی می شود. بدین شکل که طول هر قطعه (شامل EOL) به عنوان سمبول در نظر گرفته می شود. یک مثال کدینگ RLC در شکل ٤ نشان داده شده است.



شکل۴: کدینگ Runlength برای تصاویر bi-level

۲-۱-۵- کدینگ RLC دو بعدی

کدینگ RLC یک بعدی تنها همبستگی میان پیکسل های یک خط را جستجو می‌کند. در کدینگ RLC دو بعدی یا کدینگ آدرس نسبی، همبستگی میان پیکسل ها در خط حاضر و نیز خط قبلی جستجو می‌شود. با این روش، وقتی تغییری در رنگ روی می‌دهد فاصله این پیکسل تا نزدیکترین پیکسلی که در آن تغییر روی می‌دهد (هم بعد و هم قبل از این پیکسل)، در خط قبلی و همین طور آخرین پیکسل دارای تغییرات در همان خط محاسبه می‌شود و آنکه فاصله کوتاهتری داشت، به همراه یک اندیس که نوع فاصله را نشان می دهد کد می‌شود. شکل ۱۷-٦ را در [۱] ببینید.

۲-۱-۶- CCITT گروه ۳ و گروه ٤ استاندارد کدینگ دورنگار، کدREAD [۱، فصل ٦. ٦)

در روش‌های گروه ۳، اولین خط در هر k خط با کدینگ RLC یک بعدی کد می‌شود و k-1 خط باقیمانده با کدینگ RLC ۲ بعدی که با نام READ[1] شناخته می شود، کد می‌شوند. برای جزییات این روش و جداول کد واقعی، بخش ۱-٦-٦ [۱] را ببنید. دلیل استفاده از RLC یک بعدی برای هر k خط، تضعیف انتشار خطای انتقال است. در غیر این صورت در صورت استفاده از روش READ به طور پیوسته، وقتی یک بیت خطا جایی در خلال انتقال روی بدهد، تمام صفحه تأثیر می‌پذیرد.

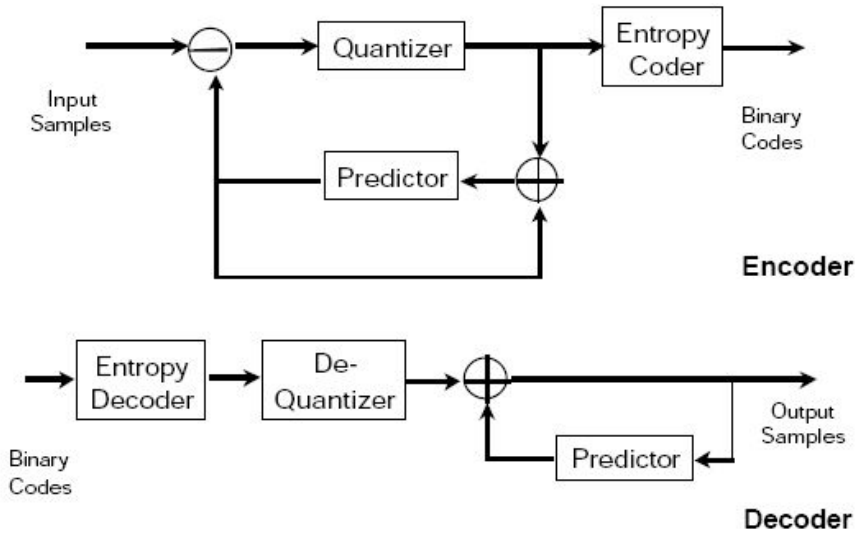
روش گروه ٤ برای رسانهٔ انتقال امن‌تر طراحی شده است، مانند خطوط دادهٔ استیجاری (leased) که نرخ خطای بسیار کوچکی دارند. الگوریتم شکل ساده شده‌ای از روش گروه ۳ است که RLC یک بعدی حذف شده است.

[1] Relative Element Address Designate

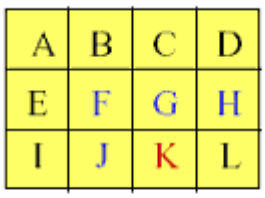
۲-۲- کدینگ پیش بینی کننده

مقدار پیکسل حاضر معمولاً تغییرات سریعی نسبت به پیکسل های مجاور ندارد. بنابراین می‌تواند از روی نمونه های قبلی پیش بینی شود. بنابراین، به جای کد کردن پیکسل های اصلی، می‌توان پیکسل حاضر را از روی قبلی ها پیش بینی کرد و تنها خطای پیش بینی را مشخص کرد. ایدهٔ کدینگ با پیش بینی قبلاً در کدینگ صحبت و صدا استفاده شده است. کدینگ پیش بینی کننده می‌توانند با تلفات یا بدون تلفات باشد، در کدینگ پیش بینی کننده با تلفات، خطای پیش‌بینی ابتدا کوانتیزه شده و سپس کد می‌شود. دلیل این کار این است که پیش‌بینی به شکلی که در ادامه خواهیم گفت به کاهش نرخ بیت کمک می‌کند. خطای پیش بینی توزیع غیر یکنواخت خواهد داشت که عمدتاً در نزدیکی صفر متمرکز است و آنتروپی کمتری نسبت به نمونه‌های اصلی دارد که معمولاً توزیع یکنواخت دارند. با کدینگ آنتروپی (مثل کدینگ هافمن) مقادیر خطا می توانند با بیت های کمتری نسبت به مقادیر نمونه های تصویر اصلی کد شوند.

شکل ۵ بلوک دیاگرام کلی برای کدگذار و کد گشای یک سیستم کدینگ با تلفات پیش بینی کننده را نشان می دهد. در یک کد کننده بدون تلفات، مرحله کوانتیزاسیون انجام نمی‌شود. شکل ٦ چگونگی استفادهٔ کدینگ پیش بینی کننده برای یک تصویر را نشان می‌دهد.



شکل۵: بلوک دیاگرام کلی برای کدگذار و کدگشای یک سیستم کدینگ با تلفات پیش بینی کننده



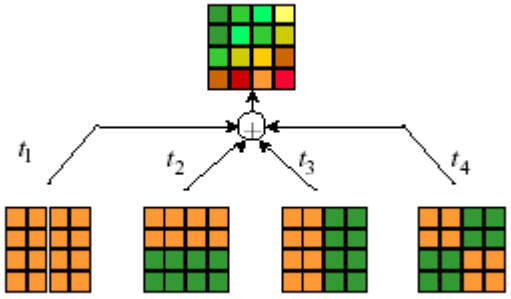
$$\hat{f}_K = a f_F + b f_G + c f_H + d f_J$$

شکل۶: کدینگ پیش بینی شده

۲-۳- کدینگ تبدیلی (کدینگ با تلفات) [۱، فصل ۵-٦]

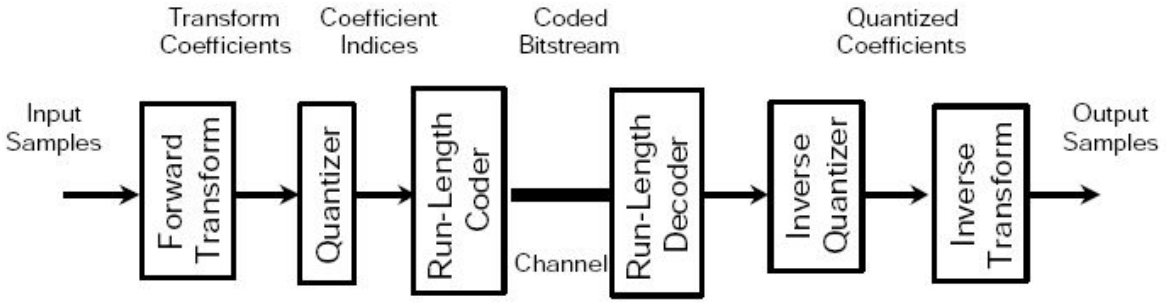
کدینگ بدون تلفات می تواند به نسبت های فشرده سازی ۲ تا ۳ برای بیشتر تصاویر برسد. برای کاهش بیشتر مقدار داده‌ها، روش های با تلفات کدینگ، کوانتیزاسیون را بر روی مقادیر تصاویر اصلی یا پارامترهای میدان تبدیل سیگنال اصلی به کار می‌برند. تبدیل از همبستگی آماری میان نمونه‌های اصلی استفاده می‌کند. روش‌های محبوب، شامل پیش بینی خطی و تبدیل های یک به یک می‌باشد. در این بخش بر روی کدینگ تبدیلی متمرکز می‌شویم که برای تصاویر، کارایی بیشتری دارد.

یکی از محبوبترین راهکارهای کدینگ با تلفات برای تصاویر، کدینگ تبدیلی است. در کدینگ تبدیلی مبتنی بر بلوک، تصویر به بلوک‌های بدون همپوشانی تقسیم می شود. هر بلوک با مجموعه ای از الگوهای پایه نشان داده می شود. (که به عنوان توابع پایه تبدیل شناخته می‌شوند) (شکل ۷) سهم هر الگوی پایه به عنوان ضریب تبدیل شناخته می‌شود. از نظر ریاضی این ضرایب با عملیات ماتریسی به دست می‌آیند. مقادیر پیکسل‌های اصلی داخل هر بلوک، یک بردار را تشکیل می دهند. در حالی که تمام توابع پایه تبدیل، یک ماتریس را می‌سازند. بردار شامل تمام ضرایب تبدیل، حاصلضرب ماتریس تبدیل با بردار پیکسل‌ها است. تبدیل طوری طراحی شده است که انرژی سیگنال اصلی تنها در ضرایب کمی جمع شود و همبستگی میان متغیرها برای کد شدن کاهش یابد. هر دوی این موارد به کاهش نرخ بیت کمک می‌کنند. شکل ۷، بوسیله تبدیل بلوکی، هر بلوک تصویر را به صورت ترکیب خطی بعضی الگوهای پایه نشان می‌دهد.



شکل ۷: ترکیب خطی بعضی الگوهای پایه

شکل ۸ پردازش یک بلوک کدینگی تبدیلی را نشان می دهد ابتدا بلوک ورودی از طریقیک عملگر ماتریسی تبدیل می‌شود. ضرایب تبدیل، سپس کوانتیزه شده و کد می‌شوند. در کدگشا، بلوک اصلی از روی ضرایب کوانتیزه شده از طریق عکس تبدیل بازسازی می‌شود.



شکل ۸: پردازش یک بلوک کدینگی تبدیلی

۲-۳-۱- تبدیل کسینوسی گسسته (DCT)

DCT برای سیگنال های تصویر محبوب است زیرا به خوبی با خصوصیات آماری سیگنال‌های تصویر موجود منطبق است. بردارهای پایه یک بعدی N-نقطه ای DCT بدین صورت تعریف می شوند:

$$h_k(n) = \alpha(k)\cos\left(\frac{(2n+1)k\pi}{2N}\right), \qquad with\ \alpha(k) = \begin{cases} \sqrt{\frac{1}{N}} & k = 0 \\ \sqrt{\frac{2}{N}} & k = 1,2,\ldots,N-1. \end{cases}$$

تبدیل و عکس تبدیل بدین صورت تعریف می شوند:

$$t(k)=\sum_{n=0}^{N-1}h_k^*(n)\,f(n)=\alpha(k)\sum_{n=0}^{N-1}f(n)\cos(\frac{(2n+1)k\pi}{2N})$$

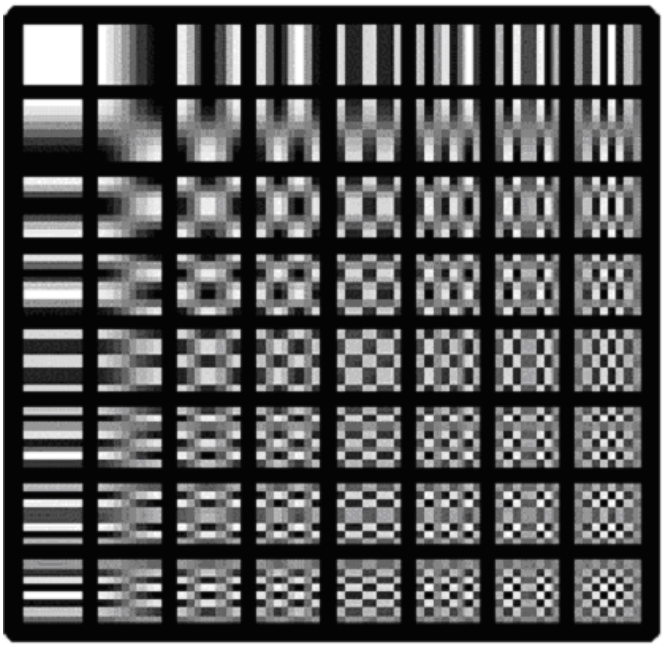
$$f(n)=\sum_{n=0}^{N-1}h_k(n)\,t(k)=\sum_{n=0}^{N-1}\alpha(k)\,t(k)\cos(\frac{(2n+1)k\pi}{2N})\ .$$

توجه کنید که بردارهای پایه در الگوی سینوسی با افزایش فرکانس، تغییر می‌کنند. توجه کنید که N DCT-نقطه‌ای به ۲N DFT-نقطه‌ای مرتبط است ولی جزء حقیقی آن نیست. هر ضریب DCT سهم الگوی سینوسی را در یک فرکانس مشخص در سیگنال اصلی نشان می دهد. کوچکترین ضریب، که به عنوان ضریب DC شناخته می شود، میانگین مقادیر سیگنال را نشان می‌دهد. بقیه ضرایب که به عنوان ضرایب AC شناخته می شوند، به فرکانس‌های بالاتر مرتبط هستند.

برای به دست آوردن DCT ۲ بعدی از یک بلوک تصویر، فرد می تواند در ابتدا DCT یک بعدی را بر روی هر ردیف بلوک تصویر به کار برد و سپس DCT یک بعدی را به روی هر ستون بلوک حاصل انجام دهد. یک تبدیل ۲ بعدی معادل نمایش سیگنال بوسیله جمع تعداد زیادی الگوهای پایه است. الگوهای بلوک پایه متناظر با DCT ۸×۸ در شکل ۹ نشان داده شده است.

این شکل با دستورات MATLAB زیر به دست آمده است.

```
T=dctmtx2(8);
figure; colormap(gray(256));n=1;
for (i=1:8)
for (j=1:8)
subplot(8,8,n),
imagesc(T(i,:)'*T(j,:));
axis off; axis image;
n=n+1;
end;
end;
```

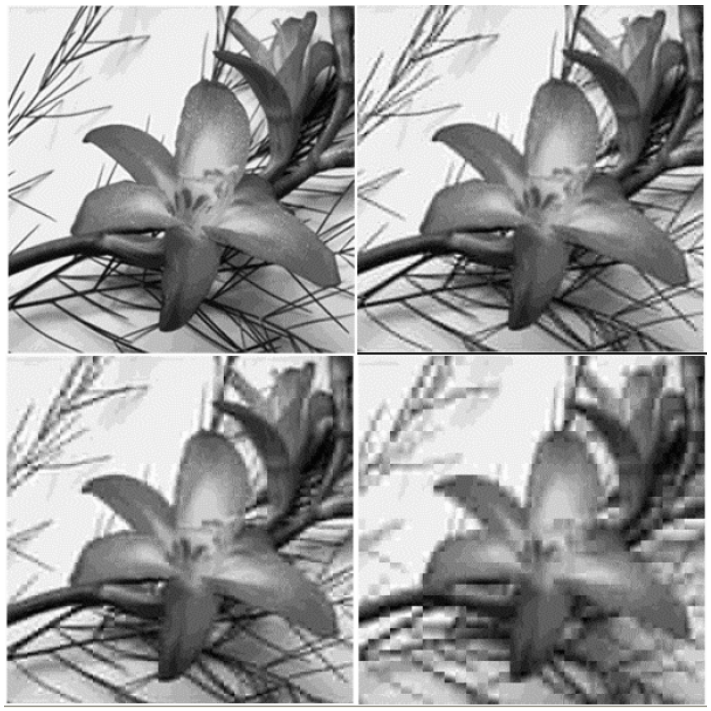


شکل ۹: الگوهای بلوک پایه متناظر با DCT ۸×۸

۲-۳-۲- نمایش بلوک های تصویر با DCT

دلیل مناسب بودن DCT برای فشرده سازی تصویر این است که یک بلوک تصویر معمولاً می‌تواند توسط ضرایب DCT بسیار فرکانس پایین نشان داده شود.

این کار به این خاطر است که مقادیر شدت در هر تصویر معمولاً به صورت هموار تغییر می کنند و مؤلفه‌های فرکانس خیلی بالا تنها در نزدیکی لبه ها وجود دارند. در شکل ۱۰ تقریب یک تصویر با استفاده از تعداد مختلفی از ضرایب DCT نشان داده شده است. می‌توانیم ببینیم که فقط ١٦ تا از ٦٤ ضریب می تواند به خوبی بلوک اصلی را نمایش دهد، شما می‌توانید دقت تقریب ضرایب با تعداد مختلف DCT را با برنامهٔ MATLAB، dctdemo آزمایش کنید.



شکل ۱۰: تقریب یک تصویر با استفاده از تعداد مختلفی از ضرایب DCT. چپ-بالا: شکل اصلی یا تمامی ضرایب DCT، راست-بالا: ۱۶/۶۴ ضریب، چپ-پایین: ۸/۶۴ ضریب، راست-پایین: ۴/۶۴ ضریب.

۲-۳-۳- استاندارد JPEG برای فشرده سازی تصویر ثابت

استاندارد JPEG برای فشرده‌سازی تصویر ثابت توصیه شده توسط گروه متخصصین عکس ISO (سازمان استانداردهای بین‌المللی) از سه بخش تشکیل یافته است:

۱- روش فشرده سازی با تلفات مبتنی بر DCT برای دقت و رزلوشن استاندارد (≤ ۸bits/color/pixel).

۲- روش کدینگ توسعه یافته برای تصاویر با دقت و رزلوشن بالاتر که الگوریتم قبلی را در داخل خود استفاده می‌کند.

۳- روش کدینگ پیش بینی کننده بدون تلفات.

در اینجا ما تنها الگوریتم اول را که بیشتر از بقیه استفاده می‌شود توضیح می‌دهیم. در ابتدا تصویر ورودی را به بلوک های ۸×۸ بدون همپوشانی تقسیم می‌کنیم. سپس یک DCT ۸×۸ بر روی هر بلوک به کار می‌رود. برای ضریب DC هر بلوک، روش کدینگ پیش بینی کننده استفاده می‌شود، یعنی مقدار DC حاضر از روی مقدار DC بلوک قبلی پیش بینی می‌شود و خطای پیش بینی با یک کوانتیزه کنندهٔ یکنواخت کوانتیزه می‌شود. ضرایب AC دیگر مستقیماً با استفاده از کوانتایزرهای مختلف، کوانتیزه می شوند (با اندازهٔ پله های مختلف). اندازه پله برای خطای پیش بینی DC و دیگر ضرایب AC در یک ماتریس نرمالیزاسیون مشخص می شود.

ماتریس به خصوص مورد استفاده می تواند در ابتدای جریان بیت فشرده شده به عنوان اطلاعات جانبی بیاید. به عنوان جایگزین، هر کس می‌تواند ماتریس توصیه شده توسط JPEG را که در شکل ۱۱ نشان داده شده است به طور پیش فرض به کار ببرد. معمولاً می توان بین کیفیت و نرخ بیت مصالحه بر قرار کرد (با استفاده از فاکتور مقیاس). به عنوان مثال، یک فاکتور مقیاس برابر ۲ به این معناست که طول پله برای تمام ضرایب ۲ برابر بشود. از سوی دیگر، فاکتور مقیاس ۰/۵کل اندازه پله ها را نصف می‌کند. فاکتور مقیاس کوچکتر، نمایش دقیقتری از تصویر اصلی را نتیجه می دهد ولی بهرهٔ فشرده‌سازی را کم می‌کند.

برای کد گشایی دودویی خطای پیش بینی DC کوانتیزه شده و ضرایب DC، ترکیبی از کدینگ با طول متغیر و طول ثابت به کار می‌رود. تمام مقادیر خطای پیش بینی ممکن و تمام مقادیر ضرایب AC بعد از کوانتیزاسیون به دسته‌های مختلف (بر اساس اندازه‌شان) تقسیم می شوند. جدول ١٤-٦ در [١] را برای قانون دسته بندی ببینید. خطای پیش بینی DC با یک کد دو قسمتی کد گذاری می شود. قسمت اول مشخص می کند که در کدام دسته است و قسمت دوم اندازه نسبی واقعی در آن دسته را نشان می‌دهد. قسمت اول بر اساس فرکانس های دسته های مختلف کد هافمن می شود و در حالی که قسمت دوم با یک کلمه کد با طول ثابت کد می‌شود. برای ضرایب AC، یک روش کدینگ RLC استفاده می شود. در این روش، ابتدا ضرایب DCT به یک آرایه یک بعدی بر اساس ترتیب زیگزاگی همان طور که در شکل ۱۲ نشان داده شده است تبدیل می شوند.

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

شکل ۱۱: یک ماتریس نرمال سازی معمول

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

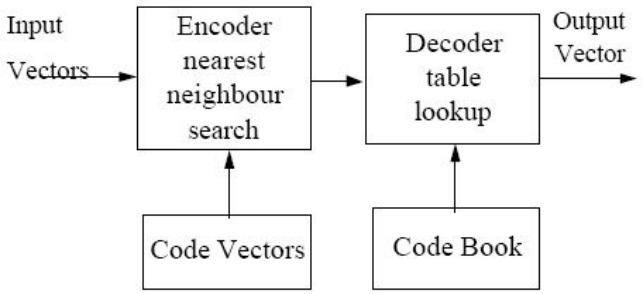
شکل۱۲: یک ماسک زیگزاگ معمول

ضرایب AC به سمبول ها تبدیل می شود و هر سمبول یک جفت تشکیل یافته از اندازهٔ صفرها از آخرین مقدار غیر صفر تا مقدار غیر صفر بعدی است. مقدار غیر صفر توسط دو قسمت مشخص می شود، شبیه کاری که برای خطای پیش بینی DC انجام شد. قسمت اول مشخص می‌کند که به کدام دسته تعلق دارد و قسمت دوم اندازه نسبی را مشخص می‌کند. سمبول تشکیل شده از RLC صفر و دسته مقدار غیر صفر، کد هافمن می شود، در حالی که اندازه نسبی مقدار غیر صفر با کد با طول ثابت کد می شود. استاندارد، بعضی جداول هافمن پیش فرض را برای خطای پیش‌بینی DC و سمبول‌های AC توصیه می کند. ولی کاربر می‌تواند جداول مختلفی که برای خصوصیات آماری تصاویر در یک کاربرد خاص بهینه شده اند را استفاده کند. برای یک تصویر رنگی، هر مولفهٔ رنگ به طور جداگانه با همین روش فشرده می‌شود. شکل ۱۳ تصاویر فشرده شده JPEG به دست آمده با انتخاب فاکتورهای مقیاس مختلف را نشان می دهد.

شکل ۱۳ : تصاویر فشرده شده JPEG به دست آمده با انتخاب فاکتورهای مقیاس مختلف

۲-۴- کوانتیزاسیون برداری

کوانتیزاسیون برداری (VQ) راهبرد دیگری برای فشرده سازی تصویر است. ایده پشت VQ تعیین بهترین دسته از الگوهای بلوک پایه (که هر کدام یک کلمه کد خوانده می شوند) است که به بهترین وجه تمام بلوک های تصاویر را در یک تصویر نمایش دهد. مجموعه تمام الگوهای پایه، کتاب – کد خوانده می شود. معمولاً بهترین کتاب – کد برای یک کلاس از تصاویر، از قبل طراحی می شود. در فرایند کدگزاری برای هر بلوک تصویر داده شده بهترین کلمه – کد منطبق در کتاب- کد پیدا می شود. اگر چه این روش جستجو محاسبات بیشتری را نسبت به JPEG طلب می‌کند، در کدگشایی بسیار سریعتر و ساده‌تر است. کدگشایی اطلاعات کدگزاری شده با کوانتیزاسیون برداری شامل پیدا کردن کلمه – کد مناسب در کتاب – کد – بر اساس اندیس ایجاد شده در خلال پردازش کدگزاری است. پیچیدگی VQ، به طور نمایی با اندازه بلوک (تعداد پیکسل‌های هر بلوک) افزایش می یابد. در عمل اندازه بلوک مساوی یا کوچکتر از ٤×٤ است که اغلب استفاده می‌شود. شکل ١٤، معماری کوانتیزاسیون برداری را نشان می‌دهد.



شکل ١٤: کوانتیزاسیون برداری

ساده‌ترین راهبرد برای پردازش بخش‌های تصویر، تقسیم تصویر ورودی در کدگزار به بلوک‌های یا بردارهای مستطیلی، غیر همپوشانی، چسبیده و کوچک از پیکسل‌ها است که هر کدام جداگانه کوانتیزه شود. ابعاد بردار مساوی تعداد پیکسل های هر بلوک

است. بردار نمونه ها الگویی است که باید با مجموعه الگوهای تست تقریب زده شود. الگوها در یک دیکشنری ذخیره می‌شوند. الگوهای موجود در کتاب – کد، کلمه – کد خوانده می شوند.

در خلال فشرده‌سازی، کدگذار، به هر دو بردار ورودی یک آدرس یا اندیس می دهد که کلمه – کدی را که در کتاب – کد به بهترین وجه بردار ورودی را تقریب می‌زند مشخص می‌کند. در کدگزار هر اندیس به یک بردار خروجی گرفته شده از کتاب – کد متناظر می‌شود. کدگشا تصویر را با همان ترتیب بردارهای ورودی بازسازی می‌کند. ویرایش سوم Indeo از شرکت Intel، بر اساس روش کوانتیزاسیون برداری کار می کند.

۳- آزمایش

۱- از یک تصویر bmp به عنوان تصویر ورودی استفاده کنید و کدینگ RLC دو سطحی را به طور دستی انجام دهید و احتمال طول های مختلف را حساب کنید. سپس کدینگ هافمن را استفاده کنید (به شکل ۳ مراجعه کنید) تا کلمه کد هر سمبول را به دست آورید. سپس طول میانگین کد را محاسبه کنید. می‌توانید از دستور ()bmpread برای خواندن و نمایش فایل ورودی استفاده کنید.

۲- با dctdemo.m بازی کنید. به تأثیر انتخاب تعداد مختلف ضرایب DCT توجه کنید. برای یک تصویر انتخاب شده، حداقل تعداد ضرایبی که باید انتخاب کنید تا نتیجه رضایت بخش بگیرید چیست؟

۳- پیوست الف برنامهٔ dctquant را می دهد که DCT را بر روی بلوک ۸×۸ حساب می کند و سپس ضرایب DCT را کوانتیزه می کند. سپس عکس DCT را انجام می دهد تا تصویر بازسازی شده را به دست آورد. با برنامه بازی کنید تا تأثیر کوانتیزاسیون را با استفاده از فاکتورهای مقیاس مختلف به دست آورید. برای فهمیدن این برنامه شما باید توابع "blkproc" و "mask2.m" را یاد بگیرید.

٤- dctquant.m را اصلاح کنید و به جای کوانتیزه کردن ضرایب DCT با استفاده از ماتریس نرمالیزاسیون فراهم شده تنها L ضریب اول را در یک ترتیب زیگ زاگ استفاده کنید. مقادیر مختلف L بین ۱ تا ١٦ را امتحان کنید.

توجه : شما باید ماسک را در برنامه طوری اصلاح کنید که بتواند تشخیص دهد که کدام ضرایب باید حفظ شوند و کدام ضرایب باید صفر شوند. یا اینکه می توانید یک برنامه "mask" جدید برای این کار بنویسید.

۵- برنامه dctquant.m را اصلاح کنید طوری که به جای کوانتیزه کردن ضرایب DCT بر اساس ماتریس نرمالیزاسیون، تنها ضرایبی را نگه دارد که اندازه ای بزرگتر از آستانه T داشته باشند. مقادیر مختلف T بین ۱ تا ٢٥٦ را امتحان کنید.

٤- گزارش

۱- دیاگرام یک الگوریتم نمونه کدینگ تصویر را بکشید (به شکل ۱ مراجعه کنید) و توضیح دهید کدام قسمت ها با تلفات و کدام بدون تلفات هستند.

۲- برنامه هایی که نوشته اید تحویل بدهید. هم کد برنامه هم خروجی آن.

۳- برای آزمایش های ۲،۳،٤ اثر پارامترهای مختلف کوانتیزاسیون را توضیح دهید، تعداد ضرایب باقیمانده و آستانه را بنویسید. حداکثر فاکتور مقیاس، حداقل تعداد ضرایب، حداکثر آستانه را به طوری که کیفیت تصویر را در حد رضایت بخش حفظ کنیم بنویسید.

٤- برای یک دنباله ویدیو، اندازه تصویر در هر قاب ٢٤٠×٣٦٠ و نرخ قابل ۳۰ قاب بر ثانیه با رنگ کامل (۳ بایت بر پیکسل) دارد. نسبت فشرده سازی برای به دست آوردن دنباله ۱/۵ mbits/second چیست؟

۵- مراجع

[1]. R. C. Gonzalez & R. E. Woods, "Digital Image Processing", Addison Wesley, 1992.
[2]. A. N. Netravali and B. G. Haskell, "Digital Pictures, Representation, Compression, and Standards", 2nd ed., Plenum Press, 1995.

پیوست الف

```
*****************************************************************
* MATLAB Script file for demonstration of DCT representation of images *
* You should find out how to use “blkproc” by on-line help in matlab.
*****************************************************************
function dctquant(FileName,dx,dy);
% usage : dctquant('h:\el593\exp10\lena.img',256,256); (WYT: please verify)
% Note, dctquant calls subfunctions mask2()
Img=fread(fopen(FileName),[dx,dy]);
colormap(gray(256));
image(Img');
set(gca,'XTick',[],'YTick',[]);
title('Original Image');
truesize;
drawnow
y=blkproc(Img,[8 8],'dct2');
yy=blkproc(y,[8 8],'mask2');
yq=blkproc(yy,[8,8],'idct2');
figure;
colormap(gray(256));
image(yq');
set(gca,'XTick',[],'YTick',[]);
title('Quantized Image');
truesize;
drawnow;
```

پیوست ب

```
****************************************************************
* MATLAB Script file for demonstration of DCT (subroutine 1) *
****************************************************************
function [y]=mask2(x);
mask=[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 56 112 100 103 99];

% Normally c=1
c=16;

mask=c*mask;
z=round(x./mask);
y=mask.*z;
```