به نام خدا

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

سیستم‌های چندرسانه‌ای نیمسال دوم ۱۴۰۰-۱۴۰۱

پروژه فشرده‌سازی JPEG

مهلت تحویل ۸ تیر ۱۴۰۱

در این پروژه قصد داریم الگوریتم فشرده‌سازی JPEG که در درس با آن آشنا شدیم را پیاده‌سازی کنیم.

این الگوریتم فشرده‌سازی lossy است که بطور خاص برای چشم انسان ایجاد شده است و از سه ویژگی زیر بهره می‌برد:

۱- اطلاعات مفید در تصاویر نسبتا به آرامی تغییر می‌کنند و بسیاری از اطلاعات تکرار می‌شوند در نتیجه افزونگی فضایی[1] داریم.

۲- چشم انسان به از دست رفتن اجزایی که فرکانس بالایی دارند کمتر از اجزایی که فرکانس پایینی دارند حساس است.

۳- دقت بینایی در تشخیص رنگ سیاه و سفید بسیار بیشتر از سایر رنگ‌ها است.

با استفاده از ویژگی‌های بالا در هر مرحله بخشی از فشرده‌سازی را انجام می‌دهیم که در ادامه به تشریح هر یک می‌پردازیم:

مرحله اول) تبدیل تصویر از فرمت RGB به YCbCr و نمونه‌برداری

با توجه به اینکه فرمت RGB مقدار illuminance و chrominance تصویر را از یکدیگر جدا نمی‌کند باید تصویر را به فرمت YCbCr تبدیل کنیم.

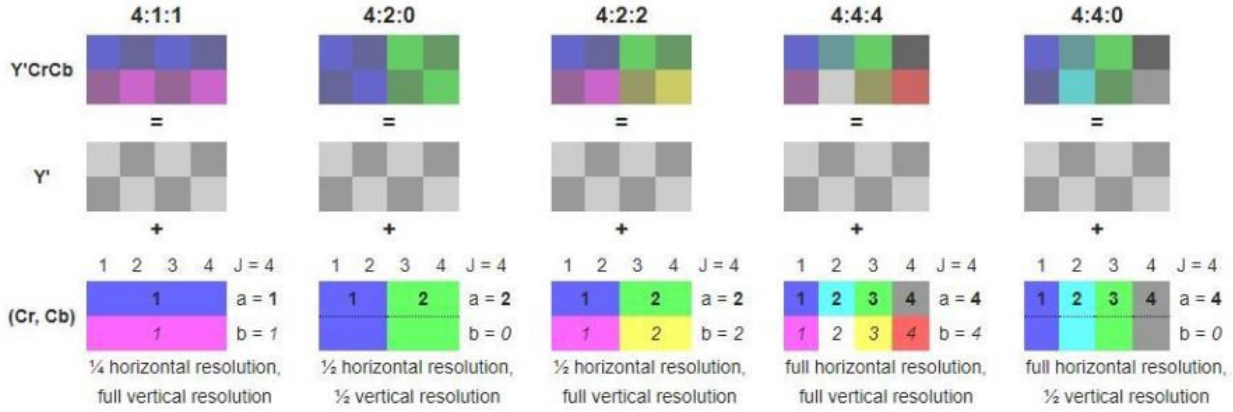
ابتدا تصویر را خوانده و سپس به فرمت مناسب تبدیل کنید.

برای این کار مجاز به استفاده از کتابخانه هستید. کتابخانه پیشنهادی ما، کتابخانه Pillow است.

با توجه به اینکه مقدار chrominance در تصاویر برای ما اهمیت کمتری دارند، با استفاده از chroma subsampling 4:2:0 این نمونه‌برداری را انجام دهید.

[1] Spatial Redundancy

برای درک بهتر به تصویر زیر دقت کنید:



در نمونه‌برداری 4:2:0 در حالت عمودی در هر دو ردیف عنصر پایینی مقدار chrominance عنصر بالایی خود را دارد. در حالت افقی نیز عنصر دوم مقدار chrominance عنصر چپی خود را دارد. برای راحت‌تر شدن کار می‌توانید از کتابخانه numpy استفاده کنید.

مرحله دوم) تبدیل کسینوس گسسته

در این مرحله ابتدا باید داده‌ها به بلوک‌های 8*8 تبدیل شوند. اگر داده‌ها قابل تقسیم به بلوک 8*8 نبودند، بلوک‌های ناقص را با رنگ مشکی پر کنید.

سوال ۱) اشکال استفاده از رنگ مشکی در پر کردن بلوک‌های ناقص چیست؟ یک روش دیگر برای بهبود پیشنهاد کنید.

همانطور که در ویژگی ۱ گفته شد تغییر شدید در تصاویر در یک بلوک نادر است. تبدیل کسینوس گسسته مقدار تغییر محتوای تصویر را نسبت به تعداد چرخه‌های یک موج کسینوسی در یک بلوک اندازه می‌گیرد. هدف آن جداسازی سیگنال اصلی به مؤلفه‌های AC و DC است.

حال بر روی هر بلوک باید تبدیل دو بعدی کسینوس گسسته[2] را اعمال کنیم تا در نهایت ضرایب DCT برای هر بلوک بدست آید. برای اعمال تبدیل می‌توانید از هر یک از دو تابع کتابخانه‌های openCV یا scipy استفاده کنید.

سوال ۲) در اینجا برای هر بلوک اندازه 8*8 را در نظر گرفتیم. توضیح دهید بزرگ‌تر کردن یا کوچک‌تر کردن اندازه بلوک چه تاثیری می‌تواند داشته باشد.

مرحله سوم) کوانتیزاسیون

در این مرحله با استفاده از ضرایب بدست آمده در مرحله قبل و ماتریس‌های کوانتیزاسیون زیر داده‌های خود را کوانتیزه می‌کنیم.

[2] 2D DCT

Table 9.1 The Luminance Quantization Table

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

Table 9.2 The Chrominance Quantization Table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

برای کوانتیزه کردن از فرمول زیر استفاده می‌کنیم که در آن F ضریب u,vامی است که در مرحله قبل بدست آوردیم. Q نیز عنصر u,vام یکی از جداول بالا است.

$$\hat{F}(u,v) = round\left(\frac{F(u,v)}{Q(u,v)}\right)$$

مرحله چهارم) پیاده‌سازی الگوریتم Huffman Coding

در این مرحله لازم است الگوریتم کدگذاری هافمن را که در درس خوانده‌اید پیاده‌سازی کنید. این پیاده‌سازی را در فایلی جدا با نام huffman.py انجام دهید. استفاده از کتابخانه به عنوان پیاده‌سازی مجاز **نیست**، اما برای محاسبه احتمال وقوع هر نماد می‌توانید از کتابخانه استفاده کنید.

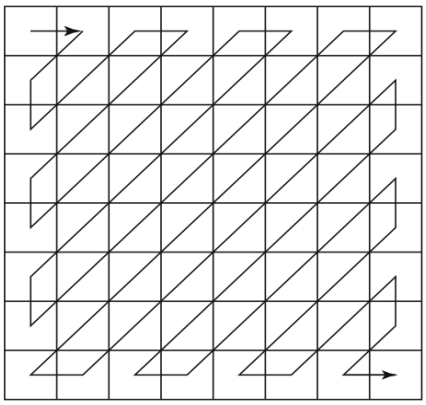
سوال ۳) با استفاده از کد پیاده‌سازی شده خروجی را برای دنباله اعداد زیر بدست آورید. سپس نرخ فشرده‌سازی را محاسبه کنید. (فرض کنید قبل از فشرده‌سازی برای ذخیره‌سازی هر عدد به یک بایت فضا نیاز داریم.)

8, 8, 34, 5, 10, 34, 6, 43, 127, 10, 10, 8, 10, 34, 10

مرحله پنجم) پیمایش زیگ زاگ و Run-Length Coding (امتیازی)

ابتدا با پیمایش بر روی بلوک‌های 8*8 مرحله قبل به صورت زیگ زاگ به شکلی که در تصویر زیر آورده شده است، هر بلوک را به یک بردار ۶۴ تایی تبدیل کنید.

پیمایش زیگ زاگ به این دلیل است که شانس قرار گرفتن صفرهای پشت سر هم افزایش یابد و در نتیجه در مرحله بعد بتوانیم فشرده‌سازی بهتری داشته باشیم.



ابتدا RLC را بر روی ضرایب AC[3] که در مرحله سوم بدست آمده‌اند اجرا می‌کنیم. در RLC با گذر از روی عناصر بردار، هر عنصر غیرصفر به یک دوتایی (runlength, value) تبدیل می‌شود. همچنین در انتهای هر بلوک از دوتایی (0,0) استفاده می‌کنیم. برای تشریح بهتر دنباله ۶۴تایی مثال زیر را در نظر بگیرید:

(32, 6, −1, −1, 0, −1, 0, 0, 0, −1, 0, 0, 1, 0, . . . , 0)

عدد ۳۲ مقدار DC است و آن را در نظر نمی‌گیریم.

بعد از آن به ترتیب با استفاده از RLC به مقادیر زیر خواهیم رسید:

(0, 6)(0, -1)(0, -1)(1, -1)(3, -1)(2, 1)(0, 0)

حال DPCM[4] را بر روی ضرایب DC اجرا می‌کنیم. انتظار ما این است که ضرایب DC بلوک‌های متوالی دارای واریانس کمی باشند. در این حالت ضریب بلوک اول عیناً آورده شده و بقیه آن‌ها به صورت تفاضلی آورده می‌شوند.

---

[3] ضرایب AC تمام ضرایب یک بلوک به جز عنصر اول آن هستند

[4] Differential Pulse Code Modulation

به عنوان مثال اگر اعداد زیر ضرایب DC پنج بلوک اول باشند

150, 155, 149, 152, 144

با استفاده از DPCM به دنباله زیر تبدیل می‌شوند:

150, 5, −6, 3, −8

مرحله ششم) اعمال کدگذاری هافمن بر روی ضرایب

اگر مرحله پنجم را انجام داده‌اید **مورد ۱** (انجام این بخش وابسته به مرحله پنجم است و **امتیازی** است) و در غیر این صورت **مورد ۲** را انجام دهید.

۱- در این مرحله ابتدا می‌خواهیم کد هافمن را بر روی ضرایب DC بدست آمده در مرحله پنجم اعمال کنیم. هر عدد بدست آمده در مرحله پنجم به یک دوتایی به شکل (size, amplitude) تبدیل می‌شود که size تعداد بیت مورد نیاز برای نمایش عدد و amplitude خود بیت‌ها است. همچنین برای نمایش اعداد منفی از مکمل ۱ استفاده می‌کنیم. به عنوان مثال اعداد بدست آمده در مرحله پنجم به شکل زیر تبدیل می‌شوند:

(8, 10010110), (3, 101), (3, 001), (2, 11), (4, 0111)

حال الگوریتم هافمن که در مرحله ۴ پیاده‌سازی کرده‌ایم را تنها بر روی sizeها اعمال می‌کنیم.

حال می‌خواهیم کد هافمن را بر روی ضرایب AC بدست آمده اجرا کنیم. در اینجا نیز تنها مقدار runlength با استفاده از کدگذاری هافمن کد می‌شود. (البته این حالت با پیاده‌سازی اصلی کمی متفاوت است)

۲- کدگذاری هافمن که در مرحله ۴ پیاده‌سازی کرده‌ایم را بر روی ضرایب بدست آمده در مرحله سوم اعمال کنید.

مرحله هفتم) ذخیره سازی

در این مرحله نیاز است تا اطلاعاتی که از مراحل قبل بدست آورده‌ایم را ذخیره کنیم.

مواردی که نیاز است ذخیره شوند عبارتند از:

۱- ضرایب بدست آمده از مرحله ۶ که با استفاده از کدگذاری هافمن encode شده‌اند.

۲- درخت‌های هافمن ایجاد شده

برای ذخیره‌سازی می‌توانید از کتابخانه pickle استفاده کنید.

سوال ۴) عکس photo1.png را با استفاده از کد خود فشرده کنید و حجم فایل ذخیره شده را نشان دهید.

توضیحات تکمیلی

- انجام این پروژه باید به صورت انفرادی باشد. بنابراین در صورت مشاهده هرگونه تقلب، برای همه افراد نمره صفر لحاظ خواهد شد.
- شما می‌توانید با استفاده از هر زبان دلخواهی این پروژه را پیاده‌سازی کنید ولی زبان پیشنهادی ما پایتون است.
- استفاده از کتابخانه‌ها تنها در مواردی که ذکر شده است مجاز است و در موارد دیگر استفاده از کتابخانه‌ها مجاز **نیست**. بدیهی است در صورت عدم رعایت این مسئله نمره‌ای دریافت نخواهید کرد.
- **توضیح مختصری از کد** خود را به همراه **پاسخ به سوالها که به شکل سوال مشخص شده‌اند**، در قالب یک گزارش با فرمت MM_JPEG_studentID.pdf آماده کنید.
- فایل گزارش و کد خود را در قالب یک فایل زیپ با فرمت MM_JPEG_studentID.zip در سامانه درس بارگذاری کنید.
- سوالات و ابهامات خود را می‌توانید از طریق ایمیل esrafiliyanm@gmail.com مطرح کنید. حتما در موضوع ایمیل کلمه **سیستم‌های چندرسانه‌ای** را ذکر کنید.
- بخشی از افراد به شکل رندوم برای تحویل انتخاب می‌شوند و بخشی از نمره شما به تسلط شما هنگام ارائه وابسته است.
- ددلاین این تمرین **۸ تیر ۱۴۰۱ ساعت ۲۳:۵۵** است و امکان ارسال با تاخیر وجود ندارد، بنابراین بهتر است انجام تکلیف را به روزهای پایانی موکول نکنید.