MultiMedia Systems HW2

Armaghan Sarvar 9531807

سوال ۱

از آنجا که می‌توانستیم با زبان‌های برنامه‌نویسی دیگر نیز کار کنیم، این تمرین با پایتون انجام شد.

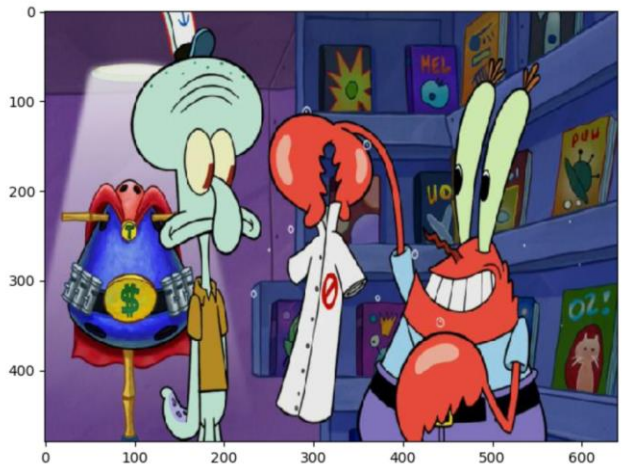
۱٫۱

میدانیم باید:

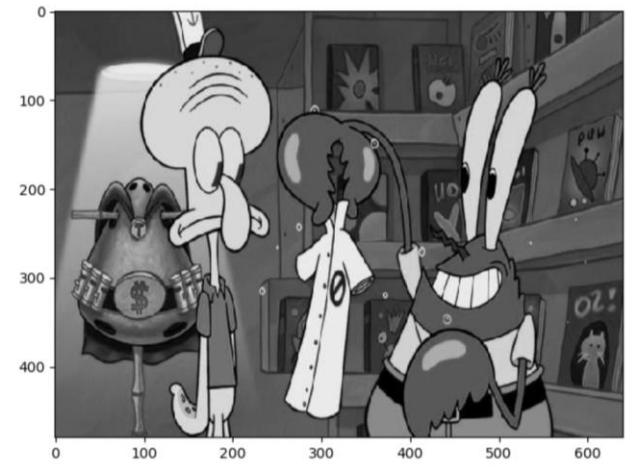
```
L = R * 299/1000 + G * 587/1000 + B * 114/1000
```

پس در تابع مربوط به این بخش، فرمول را پیاده‌سازی می‌کنیم.

تصویر اصلی:



تصویر gray-scale:



۱٫۲

بعد از آن که تصویر به ۸ بیتی سیاه و سفید تبدیل شد، یک threshold برای مقادیر پیکسل‌ها درنظر میگیریم (برای از ۱۲۷ کمتر و یا بیشتر) و تصویر زیر بدست می‌آید.

2bit gray scale

سوال ۲

۲٫۱

دیترینگ به طور کلی یک مبادله رزولوشن شدت است برای رزولوشن مکانی. در اصل این روش برای به دست آوردن الگوهایی است که به کمک آن ها مقادیر بین ۰ و ۲۵۵ قابل نمایش باشند و تصویر خاکستری داشته باشیم. همچنین می‌توان آن‌را یک نوع نویز عمدی برای از بین بردن و کاهش خطای گسسته‌سازی در نظر گرفت. دیترینگ از تکنیک halftone (مثلا ایجاد چند سطح خاکستری با استفاده از ۲ رنگ سیاه و سفید) برای افزایش سطوح بصری رنگ استفاده می‌نماید.

floyd-steinberg

	pixel	$\frac{7}{16}$
$\frac{3}{16}$	$\frac{5}{16}$	$\frac{1}{16}$

original

	.5	.5
.5	.5	.5

after error distribution

	1	0,2813
0,4063	0,3438	0,4688

۲٫۲

داریم:

```
find_closest_palette_color(oldpixel) = round(oldpixel / 255)
```

Year :..... Month :..... Day :.....

ماتریس: $\begin{bmatrix} 10 & 255 & 15 \\ 0 & 215 & 55 \\ 95 & 50 & 0 \end{bmatrix}$

$(0,0)$: $10 \to 0$ $\quad 255 \to 255 + \frac{7 \times 10}{14}$

$0 \to \frac{5 \times 10}{14}$ $\quad 215 \to 215 + \frac{10}{14}$

$\Rightarrow$ $\begin{bmatrix} 0 & 5.1 & 15 \\ 22 & 222 & 55 \\ 95 & 50 & 0 \end{bmatrix}$

$(0,1)$: $255 \to 255$ $\quad 15 \to 15 + \frac{7 \times 94}{14}$

$55 \to 55 + \frac{94}{14}$

$33 \to 33 + \frac{4 \times 94}{14}$ $\quad 222 \to 222 + \frac{5 \times 94}{14}$

$\Rightarrow$ $\begin{bmatrix} 0 & 255 & 43 \\ 47 & 234 & 70 \\ 95 & 50 & 0 \end{bmatrix}$

$(0,2)$: $15 \to 0$ $\quad 234 \to 234 + \frac{5 \times 43}{14}$

$70 \to 70 + \frac{5 \times 43}{14}$

$\Rightarrow$ $\begin{bmatrix} 0 & 255 & 0 \\ 47 & 258 & 75 \\ 95 & 50 & 0 \end{bmatrix}$

$(1,0)$: $0 \to 0$ $\quad 255 \to 255 + \frac{7 \times 47}{14}$

$95 \to 95 + \frac{2 \times 47}{14}$ $\quad 50 \to 50 + \frac{47}{14}$

$\Rightarrow$ $\begin{bmatrix} 0 & 255 & 0 \\ 0 & 278 & 75 \\ 110 & 55 & 0 \end{bmatrix}$

Year :..... Month :..... Day :.....

$(1,1)$: $۲۷۴ \to ۲۵۵$ $\quad ۷۵ \to ۷۵ + \frac{۷ \times ۲۱}{۱۴}$ $\quad ۱۱۰ \to ۱۱۰ + \frac{۳ \times ۲۱}{۱۴}$

$۵۳ \to ۵۴ + \frac{۵ \times ۲۱}{۱۴} \Rightarrow \begin{bmatrix} ۰ & ۲۵۵ & ۰ \\ ۰ & ۲۵۵ & ۸۴ \\ ۱۱۴ & ۶۰ & ۱ \end{bmatrix}$

$(1,2)$: $۸۴ \to ۰$ $\quad ۶۰ \to ۶۰ + \frac{۳ \times ۸۴}{۱۴}$ $\quad ۱ \to ۱ + \frac{۵ \times ۸۴}{۱۴}$

$\Rightarrow \begin{bmatrix} ۰ & ۲۵۵ & ۰ \\ ۰ & ۲۵۵ & ۰ \\ ۱۱۴ & ۷۶ & ۲۷ \end{bmatrix}$

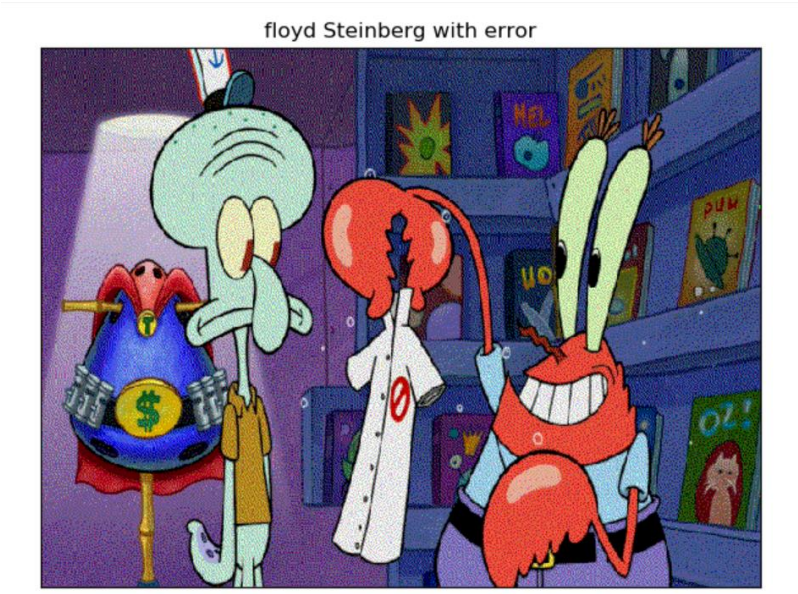
$(2,0)$: $۱۱۴ \to ۰$ $\quad ۷۶ \to ۷۶ + \frac{۷ \times ۱۱۴}{۱۴} \Rightarrow \begin{bmatrix} ۰ & ۲۵۵ & ۰ \\ ۰ & ۲۵۵ & ۰ \\ ۰ & ۱۲۶ & ۲۷ \end{bmatrix}$

$(2,1)$: $۱۲۶ \to ۰$ $\quad ۲۷ \to ۲۷ + \frac{۷ \times ۱۲۶}{۱۴} \Rightarrow \begin{bmatrix} ۰ & ۲۵۵ & ۰ \\ ۰ & ۲۵۵ & ۰ \\ ۰ & ۰ & ۸۲ \end{bmatrix}$

$(2,2)$: $۸۲ \to ۰$ $\Rightarrow$ نتیجه‌ی کل : $\begin{bmatrix} ۰ & ۲۵۵ & ۰ \\ ۰ & ۲۵۵ & ۰ \\ ۰ & ۰ & ۰ \end{bmatrix}$

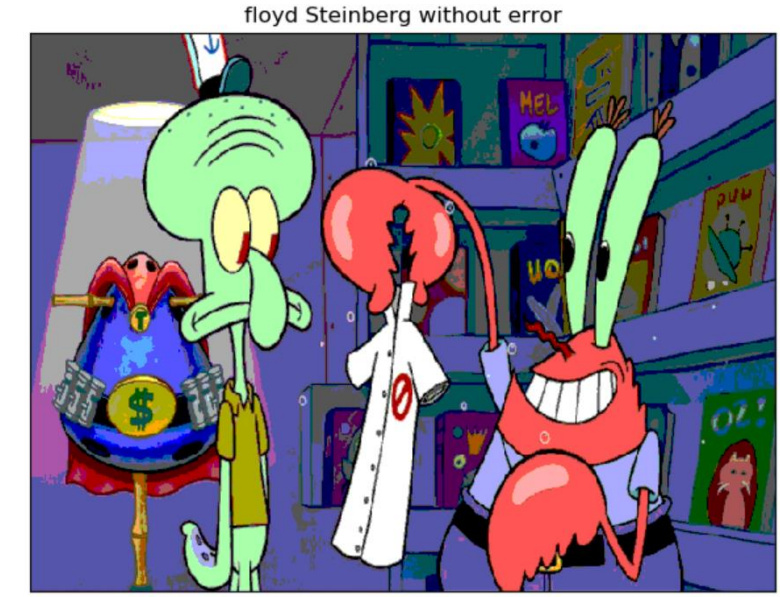
۲٫۳

با انتشار خطا:



بدون انتشار خطا:

در این قسمت فقط quantize می‌کنیم.



۲٫۴

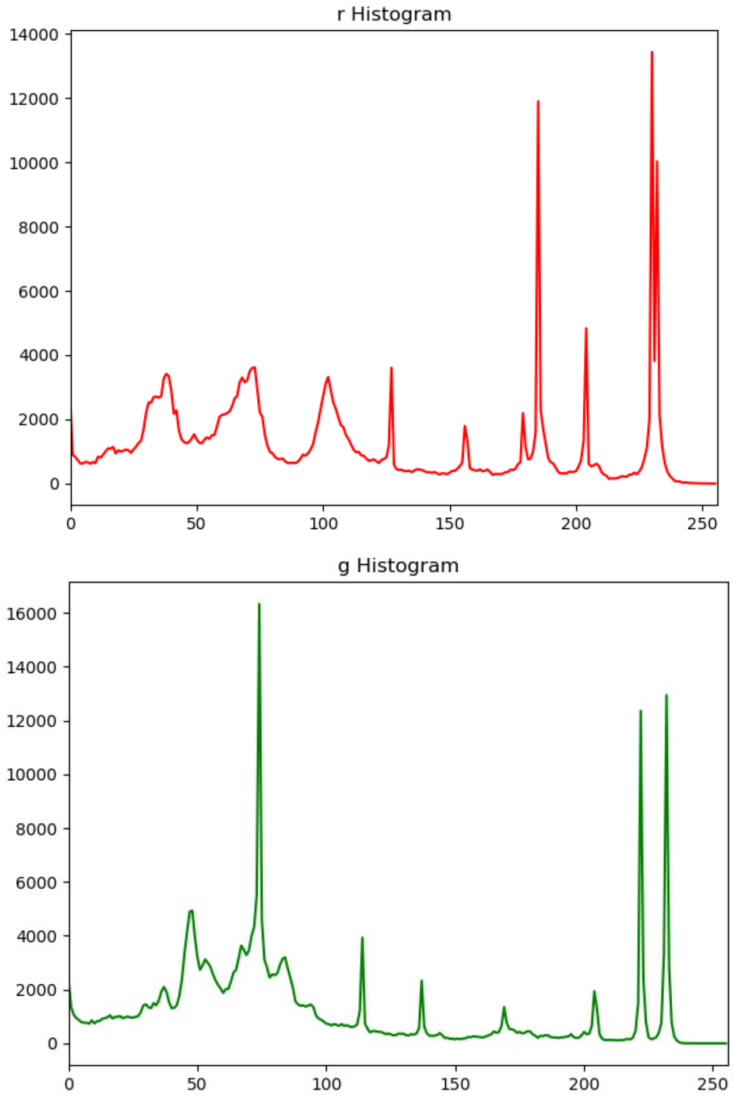
هنگامی که مقادیر همسایه‌ها برای هر پیکسل در نظر گرفته می‌شوند و خطا انتشار می‌یابد، پیکسل‌ها استقلال خود را از دست میدهند و ظاهری وابسته به همسایه‌ها میگیرند. بنابراین ممکن است دقت رنگ‌های قرمز و آبی و سبز برای یک پیکسل تغییر کند و برای همین در عکس نهایی، نقاط ریزی مانند نویز میبینیم. اما تمام رنگ‌های اصلی حفظ شده‌اند.(خطا نرم‌تر می‌شود)

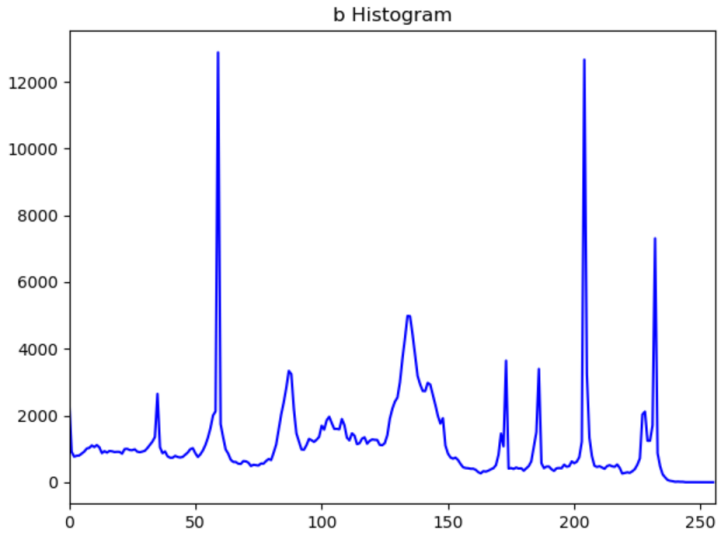
در حالت بدون محاسبه خطا، هر پیسکل تنها به مقدار خود وابسته است. برای همین در این حالت خطا شدیدتر است و جزئیات دقیق قابل مشاهده نیستند پس رنگ پیکسل‌ها تغییر بیشتری نسبت به حالت با خطا پیدا کرده‌اند.
پس با پخش خطا در پیکسل‌ها، هر کدام با یک مقدار متفاوت، به نتیجه بهتر و واقعی‌تری دست یافتیم چون هر نوع باند منحرف کننده نمونه را به حداقل می‌رسانیم و به نتیجه‌ای smooth تر می‌رسیم=> این حالت به تصویر واقعی نزدیک‌تر است.

سوال ۳

۳٫۱

هیستوگرام برای هر کانال به صورت زیر شد: (به ازای همان عکس فوق)





۳٫۲

برای پیاده‌سازی الگوریتم median-cut یک تابع بازگشتی نوشته شد که هر بار، لیست پیکسل‌ها طبق یکی از رنگ‌های قرمز و سبز و آبی سورت شده و ۰ یا ۱ به انتهای مقدار پیکسل‌های قبل و بعد از میانه اضافه می‌شود. این فراخوانی‌های بازگشتی باید از بیت ۰ ام تا ۷ ام تکرار شوند تا نهایتا به هر پیکسل یک مقدار ۸ بیتی (با نام value در کد) تعلق گیرد.

برای اینکه در انتهای کار ۲۵۶ رنگ کلی داشته باشیم، پیکسل‌های با value برابر را به رنگ‌های قرمز، آبی و سبز میانگین در کوچک‌ترین بلاک از پیکسل‌ها نگاشت می‌کنم. (در لیست mapping colors) و این mapping برای نمایش نهایی تصویر استفاده می‌شود.

۳/۳ خروجی حاصل از اجرای الگوریتم: (تصویر مربوطه کیفیت بالا نداشت و نتیجه برای عکس خام، بهتر قابل مشاهده است)

سوال ۴

فرمت PNG. : این فایل در قالب Portable Network Graphic ذخیره شده است و شامل یک بیت‌مپ از رنگ‌های ایندکس شده است و از فشرده‌سازی Lossless استفاده می‌نماید. این فرمت معمولا برای ذخیره‌سازی گرافیک تصاویر وب استفاده می‌شود و برای اولین بار برای رفع محدودیت‌های فرمت GIF به وجود آمد. گرچه نمی‌تواند مانند GIF امکان انیمیشن فراهم سازد، برای این کار می‌توانیم از فرمت MNG. استفاده کنیم. همچنین از آنجا که فایل های PNG. با هدف گرافیک‌های پیشرفته به وجود نیامده اند، حاوی پشتیبانی رنگ CMYK نیستند.

فرمت GIF. : بیانگر Graphics Interchange Format. در اصل یک فرمت برای تصاویر بوده و برای ذخیره‌سازی داده‌ی مربوط به تصویر، از indexed color استفاده می‌کند. یعنی می‌تواند تا ۲۵۶ رنگ را شامل شود. color mapping در GIF هنگام نمایش چندین عکس، می‌تواند برای هر عکس به طور جداگانه تعریف شود و یا یک نگاشت سراسری برای همه باشد. فرمت های 87a و 89a هر دو می‌توانند از انیمیشن، تصاویر transparent و metadata پشتیبانی کنند. اما فرمت 89a می‌تواند از delay در انیمیشن هم برای نمایش تصاویر متحرک در زمان‌های مختلف پشتیبانی نماید.