به نام خدا

پردازش تصویر

تمرین شماره‌ی ۳

فیلترگذاری مکانی

تاریخ تحویل: ۱۴۰۰/۲/۱۱

ارشین سلطان بایزیدی

۹۷۳۳۰۳۷

استاد درس: دکتر حامد آذرنوش

نیمسال بهار ۹۹-۰۰

سؤال ۱

Date

سؤال (۱) :

$$A = \begin{bmatrix} 0 & 0 & 10 & 10 & 0 \\ 0 & 10 & 20 & 20 & 10 \\ 0 & 10 & 20 & 20 & 10 \\ 0 & 0 & 10 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Reflect_padding :

$$A_p = \begin{bmatrix} 10 & 0 & 10 & 20 & 20 & 10 & 20 \\ 0 & 0 & 0 & 10 & 10 & 0 & 10 \\ 10 & 0 & 10 & 20 & 20 & 10 & 20 \\ 10 & 0 & 10 & 20 & 20 & 10 & 20 \\ 0 & 0 & 0 & 10 & 10 & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 10 & 0 & 10 \end{bmatrix}$$

برای نرم‌کردن تصویر از فیلتر Averaging استفاده می‌کنیم:

$$M_{(mask)} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 10 & 0 & 10 \\ 0 & 0 & 0 \\ 10 & 0 & 10 \end{bmatrix} * \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

$$= \begin{bmatrix} 10/9 & 0 & 10/9 \\ 0 & 0 & 0 \\ 10/9 & 0 & 10/9 \end{bmatrix} \rightarrow A_1 = \sum a_i = \frac{40}{9}$$

P4PCO

Subject:
Date

$$A_2 = \begin{bmatrix} 0 & 10 & 20 \\ 0 & 0 & 10 \\ 0 & 10 & 20 \end{bmatrix} * \begin{bmatrix} Mask \end{bmatrix} = \begin{bmatrix} 0 & 10/9 & 20/9 \\ 0 & 0 & 10/9 \\ 0 & 10/9 & 20/9 \end{bmatrix}$$

$$\Rightarrow A_2 = \frac{70}{9}$$

$$A_{25} = \begin{bmatrix} 10 & 0 & 10 \\ 0 & 0 & 0 \\ 10 & 0 & 10 \end{bmatrix} * Mask = \begin{bmatrix} \frac{10}{9} & 0 & \frac{10}{9} \\ 0 & 0 & 0 \\ \frac{10}{9} & 0 & \frac{10}{9} \end{bmatrix}$$

$$A_{25} = \frac{40}{9}$$

$$A_{Final} = \begin{bmatrix} \frac{40}{9} & \frac{70}{9} & \frac{120}{9} & \frac{120}{9} & \frac{120}{9} \\ \frac{60}{9} & \frac{90}{9} & \frac{120}{9} & \frac{120}{9} & \frac{120}{9} \\ \frac{60}{9} & \frac{90}{9} & \frac{120}{9} & \frac{120}{9} & \frac{120}{9} \\ \frac{40}{9} & \frac{60}{9} & \frac{70}{9} & \frac{70}{9} & \frac{70}{9} \\ 0 & \frac{20}{9} & \frac{50}{9} & \frac{40}{9} & \frac{40}{9} \end{bmatrix}$$

برای تیز کردن تصویر از فیلتر laplacian استفاده می‌کنیم:

$$L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 10 & 0 & 10 \\ 0 & 0 & 0 \\ 10 & 0 & 10 \end{bmatrix} * L = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Rightarrow A_1 = 0$$

$$A_2 = \begin{bmatrix} 0 & 10 & 20 \\ 0 & 0 & 10 \\ 0 & 10 & 20 \end{bmatrix} * L = \begin{bmatrix} 0 & 10 & 0 \\ 0 & 0 & 10 \\ 0 & 10 & 0 \end{bmatrix} \Rightarrow A_2 = 30$$

$$A_{25} = A_1 = 0$$

$$\rightarrow A_{Final} = \begin{bmatrix} 0 & 30 & 10 & 10 & 40 \\ 40 & -10 & -20 & -20 & 10 \\ -60 & 10 & -20 & -20 & 10 \\ 20 & 20 & -10 & -10 & 30 \\ 0 & 10 & 20 & 20 & 0 \end{bmatrix}$$

سؤال ۲

الف) تابع خواسته‌شده را تعریف می‌کنیم. برای اعمال فیلتر averaging، کرنلی می‌سازیم که همه‌ی درایه‌های آن ۱ باشد و کل آن در ۱/۹ ضرب شده باشد. این کرنل مخصوص میانگین‌گیری تصاویر است و برای نرم‌کردن تصویر به‌کار می‌رود. با دستور cv.copyMakeBorder حاشیه‌ی تصویر را بازتاب می‌کنیم. اکنون باید دستوری بنویسیم که کرنل را در تصویرمان کانوالو کند؛ به‌این‌صورت که به‌ترتیب مانند ماسک روی تصویر گذاشته شود و درایه‌های آن بخش از تصویر در درایه‌های کرنل ضرب شده و همه‌ی حاصل‌ضرب‌ها جمع شوند. سپس این حاصل‌جمع مقدار یک پیکسل تصویر جدید ما می‌شود. برای ایجاد تصویر جدید، آرایه‌ی خالی‌ای از نوع float درست می‌کنیم و با استفاده از حلقه‌ای که از چهار for تشکیل شده و حد دو for اول از n در تصویر (که n متوسط اندازه‌ی کرنل است) تا فاصله‌ی بین حاشیه‌ی تصویر و n انتخاب می‌کنیم تا هنگامی که ماسک را روی تصویر قرار می‌دهیم، اولین مقدار دقیقاً وسط ماسک باشد و تعداد همسایگی‌های آن کمتر نباشد (مثلاً اگر کرنل ما ۳در۳ است باید در همسایگی درایه‌ی مرکزی ۸ پیکسل داشته باشیم پس تصویر ما از [۱،۱] شروع می‌شود. دو for دوم نیز در حد n– تا n است). اکنون رابطه‌ای می‌نویسیم که با جمع‌شدن متغیرهای حلقه در هر پیکسل از تصویر، درایه‌ها مطابق با آنچه گفته شد کانوالو شوند و در تصویر خروجی ذخیره شوند.

برای فیلتر minimum نیز ۴ for مینویسیم. لیستی به‌اندازه‌ی اندازه‌ی کرنل به‌توان ۲ می‌سازیم و شمارشگر c را قرار می‌دهیم تا با تکرار شدن هر دو حلقه‌ی for دوم، یکی به آن اضافه شود. سپس لیست خالی را در هر دور که به‌اندازه‌ی ماتریس کرنل از تصویرمان یک دسته پیکسل می‌گیرد، با مقادیر تصویر پر می‌کنیم (به‌کمک شمارشگر اعضای لیست یکی‌یکی پر می‌شود) و با دستور scaled از کوچک به بزرگ مرتب می‌کنیم. در حالی که همان دوحلقه‌ی دوم تکرار می‌شود، کم‌ترین مقدار لیست را انتخاب کرده و در آرایه‌ی خالی جدید قرار می‌دهیم تا پیکسل‌های تصویر جدید ایجاد شوند.

برای فیلتر median نیز به همین ترتیب عمل می‌کنیم با این تفاوت که مقدار میانی لیست را انتخاب می‌کنیم.

برای فیلترهای sobel_y و laplacian ما کرنل‌های آماده داریم که آن‌ها را با استفاده از دو حلقه‌ی for بر تصویر کانوالو می‌کنیم.

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

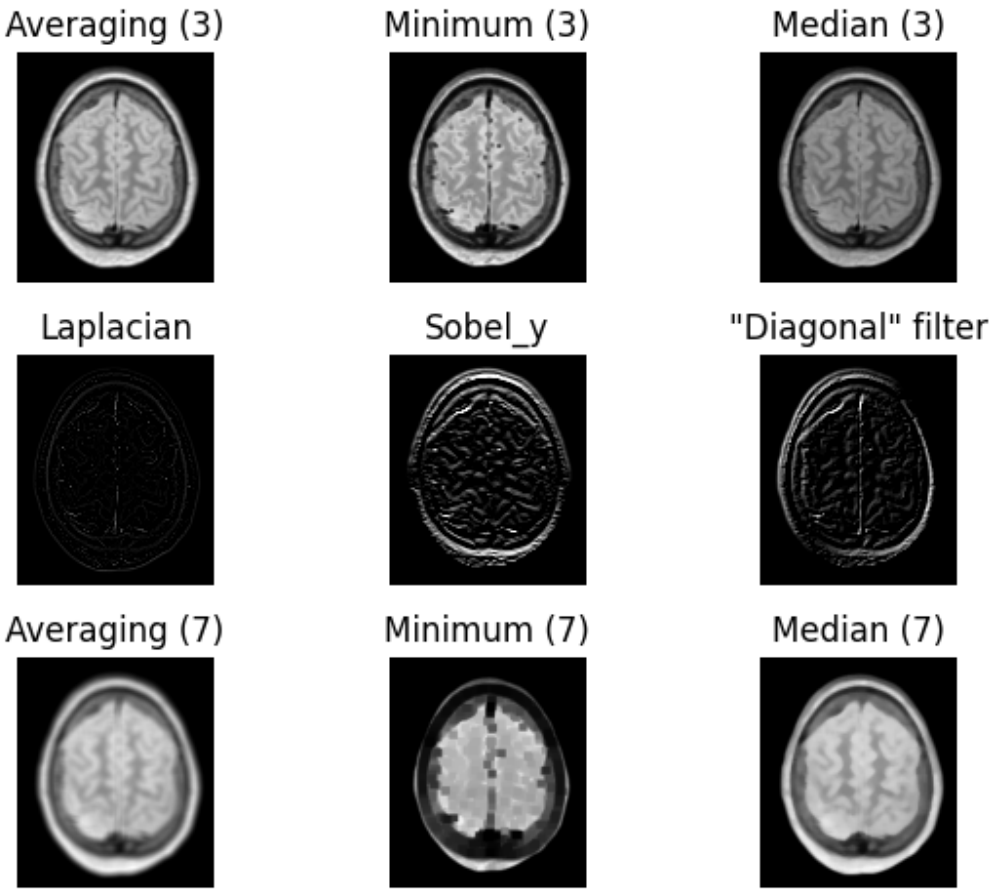
Sobel_y filter

0	1	0
1	−4	1
0	1	0

Laplacian filter

ب) این فیلتر به صورتی است که انگار به‌صورت اریب لبه‌های تصویر را آشکارسازی می‌کند. مثلاً اگر تصویر را دارای ۴ ربع در نظر بگیریم، ربع‌های دوم و چهارم لبه‌های مشخص‌تر و روشن‌تری دارند. نام آن را diagonal filter گذاشته‌ام.

ج) تصویر را با مقادیر خواسته‌شده در تابع قرار داده و خروجی‌ها را در یک پنجره نمایش می‌دهیم. مقدار vmin, vmax را طبق خواسته‌ی سؤال برای همه‌ی تصاویر جز فیلترهای laplacian، sobel_y و diagonal آزاد می‌گذاریم (برای این سه اگر مقادیر را قرار ندهیم، تصویر خاکستری و ناواضح می‌شود).



سؤال ۳

الف) در ابتدا، با دستور cv.copyMakeBorder لبه‌های تصویر را بازتاب می‌کنیم تا هنگام اعمال فیلتر حاشیه‌ی تصویرمان دچار مشکل نشود. فیلترهای میانه‌گیری (median) و میانگین‌گیری (averaging) را با دستورهای آماده کتابخانه‌ی OpenCV و با ایجاد کرنل‌های مناسب که در زیر آورده شده است، روی تصویرمان اعمال می‌کنیم.

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

Averaging filter

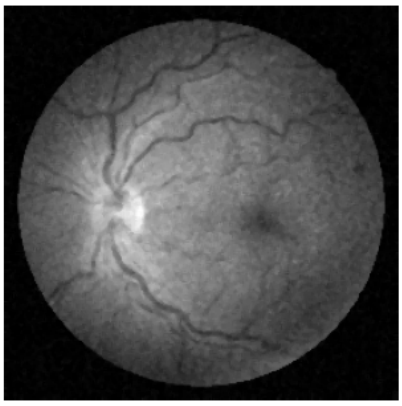
$$\begin{bmatrix}1&1&1\\1&-8&1\\1&1&1\end{bmatrix}$$

Laplacian filter

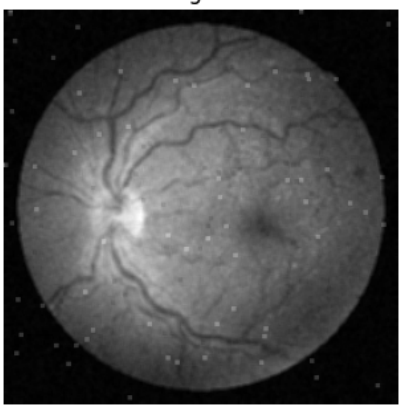
تصویر اولیه:

خروجی:

Median filter



Average filter



با اعمال هردو فیلتر، تصویر کمی نرم شده است (این فیلترها برای نرم‌کردن تصاویر به‌کار برده می‌شوند). فیلتر averaging مقدار نویزهای سفیدی را که روی کل تصویر است کمرنگ‌تر کرده و روشنایی دایره‌ی وسط را بیشتر کرده است. فیلتر median نیز روشنایی تصویر را بالا برده اما نویز را در حدی کم کرده که تقریباً هیچ نقطه‌ی سفیدی در تصویر دیده نمی‌شود.

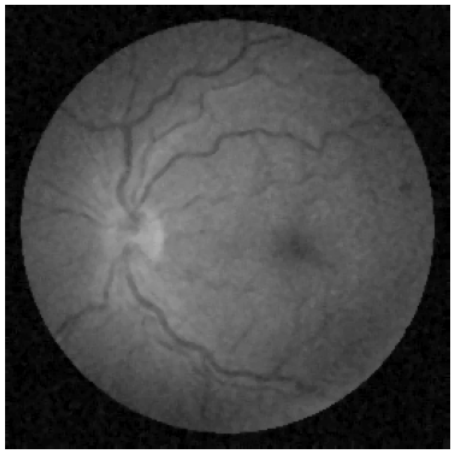
ب) تابع موردنظر را باتوجه به رابطه‌ی تبدیل توانی یا تبدیل گاما، نوشته و خروجی را طبق خواسته‌ی سؤال، uint8 می‌کنیم.

$$s = cr^{\gamma}, c = \left(L-1\right)^{1-\gamma}$$

ج) بینایی انسان در شرایط روشنایی معمولی، از یک تابع توانی تقریبی پیروی می‌کند که حساسیت بیشتری به تفاوت‌های نسبی میان رنگ‌های تیره در مقایسه با رنگ‌های روشن دارد. در حالت کلی، نقش تبدیل گاما این است که با استفاده از این ویژگی بینایی انسان که رفتاری غیرخطی دارد، استفاده از بیت‌ها را در تصویر بهینه‌سازی کند. در صورتی که تصاویر با تبدیل گاما تنظیم نشوند، بیت‌های بیش از حدی به قسمت‌هایی از تصویر اختصاص داده می‌شود که چشم انسان قادر به ایجاد تمایز بین آن‌ها نیست و بیت‌های کم‌تری به قسمت‌هایی اختصاص داده می‌شود که چشم انسان حساسیت بیشتری به آن شدت از روشنایی دارد.

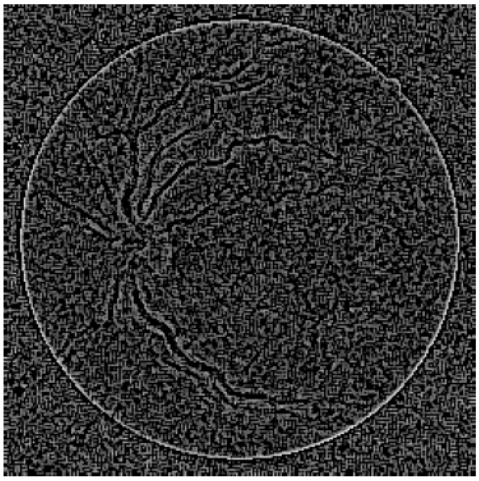
با دادن گامای کوچک‌تر از ۱، تصویر روشن‌تر و با گامای بزرگ‌تر از ۱، تصویر تیره‌تر می‌شود. در اینجا ما گامای کمتر از ۱ را وارد می‌کنیم و می‌بینیم که تصویر میانه‌گیری‌شده‌ی ما کمی محوتر و نرم‌تر شده و کنتراست آن کاهش یافته است. نواحی‌ای که در تصویر میانه‌گیری‌شده روشنایی بیشتری دارند، متعادل‌تر شده و برعکس.

Gamma transformation with gamma=2/3 on median filtered image



د) با اعمال فیلتر لاپلاسین بر تصویر و دادن گامای ۱/۳، تصویر ما به‌همراه نویزها روشنایی بیشتری پیدا می‌کند و کل تصویر حالتی تیز و نویزی می‌یابد. بخش‌هایی از کره‌ی چشم مانند رگ‌ها که نسبت به اطراف‌شان تیره‌تر هستند و همچنن محیط دایره، کمتر تغییر یافته‌اند (به‌خاطر اعمال فیلتر لاپلاسین و آشکارشدن لبه‌های داخل تصویر) ولی بخش بیرون از دایره که فقط سیاه با مقداری نویز بود، کاملاً نویزی شده است.

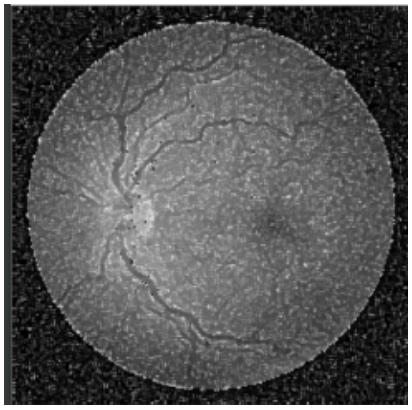
Gamma transformation with gamma=1/3 on laplacian filtered image



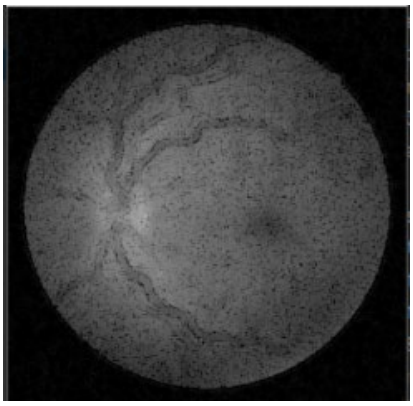
ه) اکنون با دستورات مناسب در کتابخانه‌ی OpenCV ویدیویی را با ۲۰ فریم بر ثانیه و هم‌اندازه‌ی تصویر اصلی می‌سازیم. سپس دنباله‌ی موردنظر را ساخته و عکس‌ها را به نوع float تبدیل می‌کنیم. در یک حلقه‌ی for به تعداد اعضای دنباله‌ی حسابی، رابطه‌ی موردنظر را نوشته و مقادیر خارج از شدت‌های ۰ و ۲۵۵ را صفر می‌کنیم. سپس ویدیو را تبدیل به uint8 کرده و آن را ذخیره و پخش می‌کنیم.

با تغییردادن مقادیر دنباله، درمی‌یابیم که دنباله در اعداد منفی، ویدیویی به ما می‌دهد که روشنایی کمتر دارد و تمایز بین دایره و پس‌زمینه‌ی تیره‌ی آن بیشتر و تصویر نرم‌تر است. در مقادیر مثبت، تصویر روشنایی و نویز بیشتری دارد و بسیار تیزتر است و تقریباً مانند تصویر بخش قبل می‌شود؛ چون ماتریس mask ما که برای فیلتر لاپلاسین است، در عدد مثبت‌تری ضرب می‌شود. هرچه این اعداد بزرگ‌تر باشند، طول ویدیو بیشتر است و هرچه گام‌های دنباله کوچک‌تر باشد، ویدیو با سرعت کم‌تری تغییر می‌کند.

در دنباله‌ی موردنظر سؤال، همانطور که دنباله به سمت مثبت‌شدن می‌رود، تصویر تیزتر و نویزی‌تر می‌شود:



پس از اینکه دنباله به مقادیر مثبت رسید



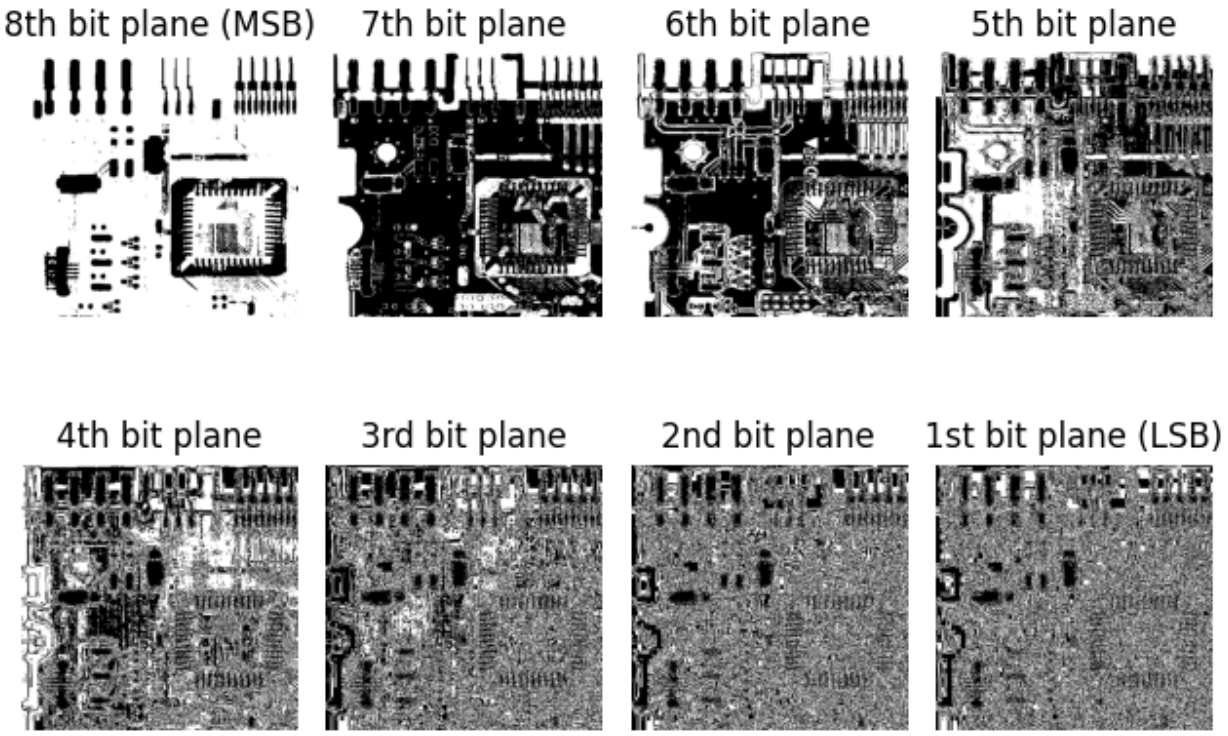
قبل از اینکه دنباله به صفر برسد

اگر از همان مقادیر ۸بیتی بی‌علامت (uint8) در محاسبات استفاده می‌کردیم، بیشتر تصویر ما سفید می‌شد و شدت‌های تصویر به‌هم می‌ریخت و عملیات به‌درستی روی تصویر انجام نمی‌شد.

سؤال ٤

الف) برای نوشتن تابع bit plane slicing، باید هر پیکسل تصویر را به مبنای ۲ تبدیل کنیم، سپس برای بریدن (slicing) صفحه‌ی موردنظر، بیت جایگاه n از عددهای مبنای ۲ هر پیکسل را جدا کرده و در آرایه‌ای جدید قرار دهیم. یعنی اگر صفحه‌ی اول را می‌خواهیم، باید بیت اول (LSB) هر پیکسل را برداریم و در پیکسل متناظرِ آرایه‌ی جدید قرار دهیم؛ و به همین ترتیب تا صفحه‌ی هشتم.

در تابعی که نوشته‌ایم، به این ترتیب عمل می‌کنیم که یک لیست خالی درست کرده و با دستور np.binary_repr در یک حلقه‌ی تودرتو، یکی‌یکی پیکسل‌های تصویر را خوانده و تبدیل به مبنای ۲ می‌کنیم و مقدار باینری هر پیکسل را به‌صورت استرینگ در لیست ذخیره می‌کنیم، طول آن را هم ۸ [بیت] درنظر می‌گیریم. اکنون یک لیست داریم که اعضای آن استرینگ‌هایی به‌تعداد پیکسل‌های تصویر و هر استرینگ یک عدد ۸ بیتی است. حال خروجی را تعریف می‌کنیم. در خروجی باتوجه به عدد n، n-۸ امین بیت هر عدد باینری (استرینگ) را در یک آرایه‌ی جدید می‌ریزیم (n-۸ به این دلیل است که بیت هشتم ما، عضو اول استرینگ است) و آن را به uint8 تبدیل کرده و در آخر با دستورreshape اندازه‌ی آرایه را مساوی با تصویر می‌کنیم. اکنون ما یک تصویر داریم که صفحه‌ی nام تصویر باینری ما است.



ج) سه تصویر را می‌خوانیم و دوتا دوتا با یکدیگر مقایسه می‌کنیم. برای ساختن آشکارساز حرکت، طبق گفته‌ی سؤال باید صفحات متناظر هر تصویر باهم XOR شده و در ضریبی ضرب شوند و سپس جمع شوند. در این بخش از ما خواسته شده که صفحات ۵، ۶، ۷ و ۸ را مقایسه کنیم.

در رابطه‌ی زیر که برای ساختن آشکارساز از طریق XOR کردن استفاده می‌شود، c برابر با حاصل XOR دو تصویر و k صفحه‌ی موردنظرمان است و Y خروجی. طبق این رابطه، ضریبی که حاصل XOR در آن ضرب می‌شود، ۲ به توان صفحه‌ای است که داریم با آن دو تصویر را مقایسه می‌کنیم.

$$\sum_{k=4}^{7} 2^k * c_k = Y,$$

پس حلقه‌ای می‌نویسیم که صفحات ۵ تا ۸ دو تصویر را با هم مقایسه کند، در ضریب ضرب کند و در یک آرایه‌ی خالی جدید با اندازه‌ی تصویرهای‌مان بریزد. آرایه‌های Compare1 و compare2 تصاویر حاصل ما هستند که به‌ترتیب حاصل مقایسه‌ی تصاویر اول و دوم، و دوم و سوم هستند.