به نام خدا

پردازش تصویر

تمرین شماره‌ی ۱

پایه‌های پردازش تصویر و عملیات‌های پایه‌ای

تاریخ تحویل: ۱۴۰۰/۱/۶

ارشین سلطان بایزیدی

۹۷۳۳۰۳۷

استاد درس: دکتر حامد آذرنوش

نیمسال بهار ۹۹-۰۰

سؤال ۱

subject: date:

سؤال 2.7 :

$$i(x,y) = ke^{-[(x-x_0)^2+(y-y_0)^2]}$$

$r = 1$

$k = 255$

$$f(x,y) = i(x,y)r(x,y) = 255e^{-[(x-x_0)^2+(y-y_0)^2]}$$

$$\frac{(255+1)}{2^n} = 8 \qquad 2^{n+3} = 256 \Rightarrow n = 5 \Rightarrow k = 2^5 = 32$$

bits of intensity

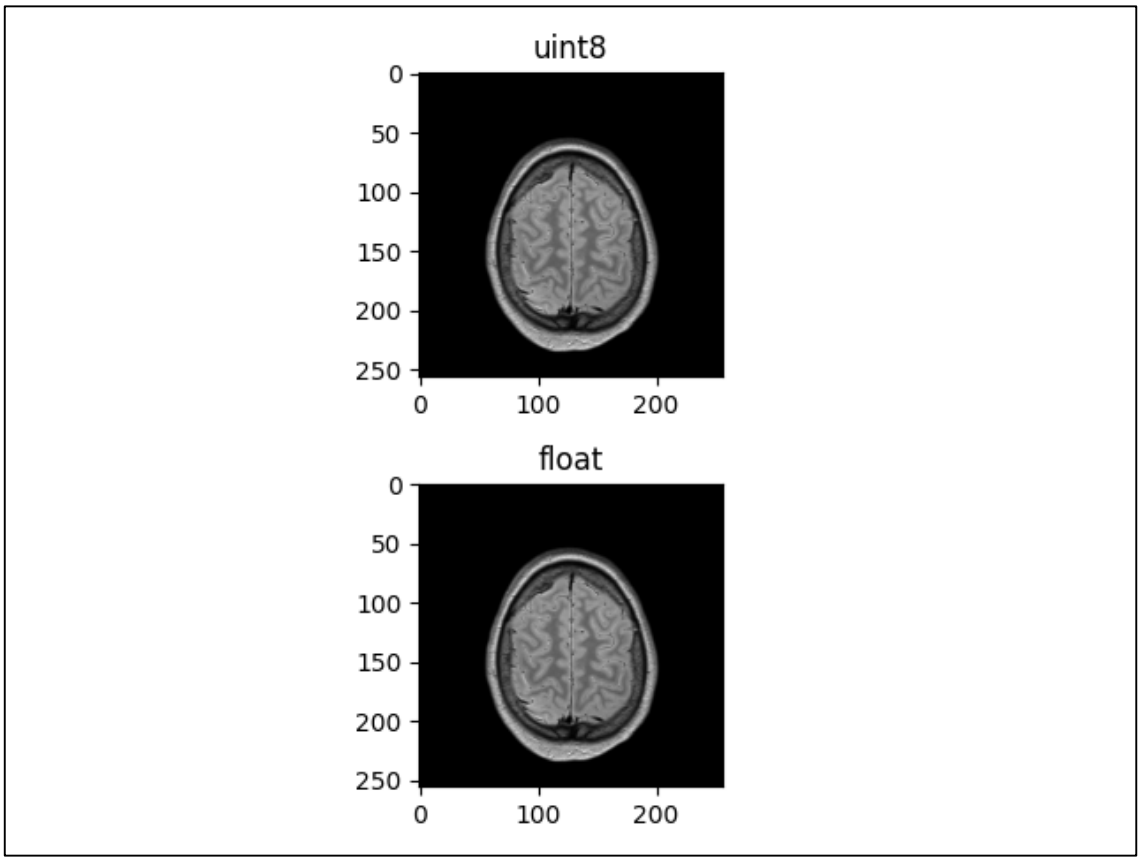
سؤال 2.9 :

a) $\frac{(1024 \times 1024)(8+2)}{56000} = 187.24$ sec

b) $\frac{(1024 \times 1024)(10)}{3 \times 10^6} = 3.495$

سؤال ۲

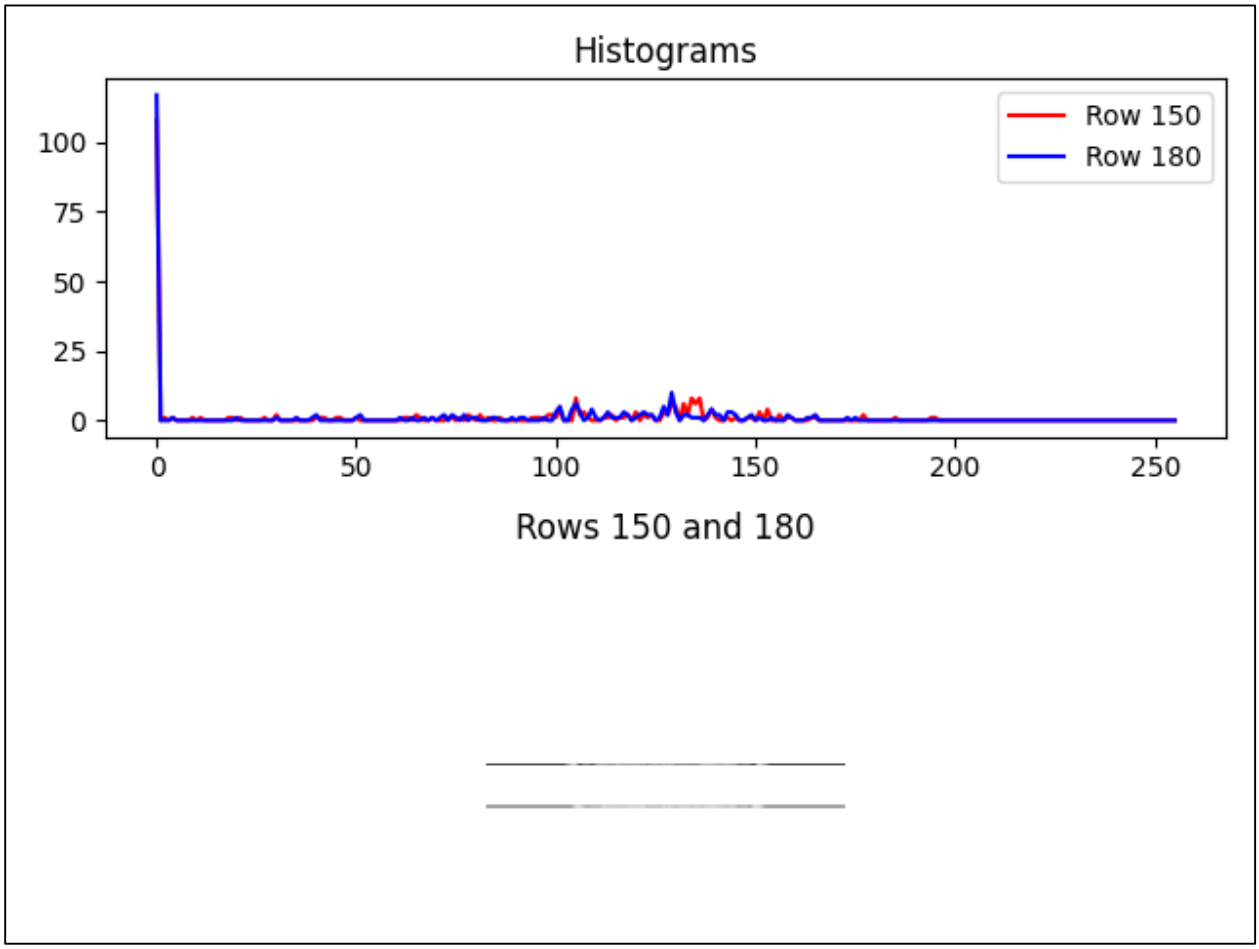
پس از خواندن عکس موردنظر، بخش‌های مختلف نمودار اول را رسم کرده و با دستورهای مناسب در کتابخانه‌ی NumPy خواسته‌های مسئله را انجام می‌دهیم. نمودار بخش الف به این صورت است:



در بخش ب، برای رسم هیستوگرام ردیف‌های موردنظر، می‌توانستیم از کتابخانه‌ی matplotlib استفاده کنیم؛ اما از آنجا که صورت سؤال از ما خواسته تا هیستوگرام را با plot نمایش دهیم و آن را به‌صورت نموداری مستقل نمی‌خواهد، از دستور رسم هیستوگرام در کتابخانه‌ی OpenCV یا NumPy می‌توان استفاده کرد که من اولی را انتخاب کردم تا هیستوگرام را به‌صورت عکس دریافت کنم.

در بخش ج که پایین نمودار دوم را در آن رسم می‌کنیم، می‌بینیم که اگر تنها با دستور plt.imshow ردیف‌های ۱۵۰ و ۱۸۰ را نشان دهیم، در نمودار روی یکدیگر قرار می‌گیرند. پس به‌جای این کار، کل عکس را به جز دو ردیف (دو بازه در ابعاد عکس) به‌رنگ سفید درمی‌آوریم تا تنها آن دو ردیف دیده شوند.

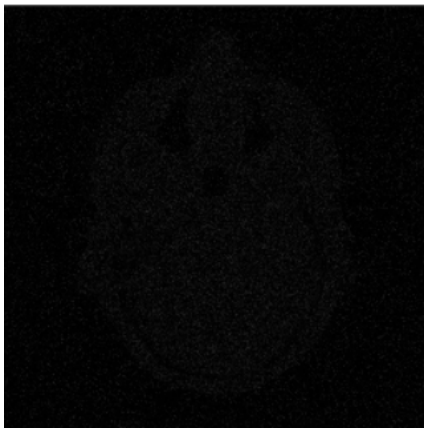
ب و ج:


Histograms
Row 150
Row 180
100
75
50
25
0
0
50
100
150
200
250
Rows 150 and 180

سؤال ۳

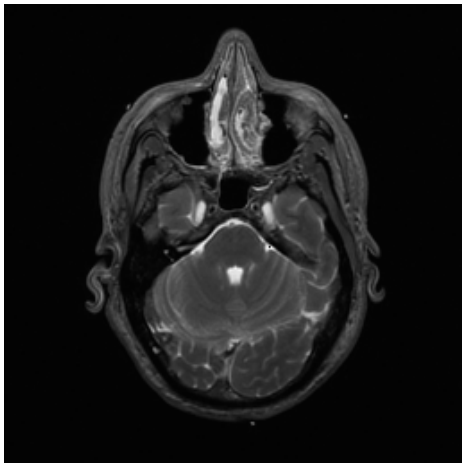
قسمت آ و ب: ابتدا تصویر و ویدیوی موردنظر را می‌خوانیم. برای استفاده از دستور int32 که در صورت‌مسئله به آن اشاره شده، می‌توان از ابتدا هنگام خواندن تصویر و همچنین تغییر رنگ ویدیو به خاکستری، استفاده کرد. در اینجا برای استفاده از ویدیو آن را خاکستری می‌کنیم تا ابعاد آن با تصویرمان که خاکستری است یکی شود و بتوانیم از یکدیگر کم‌شان کنیم. برای به‌دست آوردن فریم اول ویدیو، کافی‌ست دستورهای خواندن ویدیو را بدون حلقه نوشته تا فقط فریم اول خوانده شود. حال تفاوت خالص دو عکس را با دستور absdiff به‌دست می‌آوریم که نوع آن آرایه (ndarray) است، و با استفاده از کتابخانه‌ی NumPy میانگین و انحراف معیار عناصر آرایه را محاسبه می‌کنیم.

برای دیدن اختلاف دو تصویر به‌صورت یک تصویر، نویز را که یک آرایه است به uint8 تبدیل می‌کنیم. این تصویر نویز یا همان اختلاف عکس اصلی از فریم اول ویدیو است:



برای گرفتن تصویر میانگین ویدیو، یک لیست خالی می‌سازیم و با نوشتن یک حلقه‌ی for، ویدیو را می‌خوانیم و آن را به‌تعداد فریم‌های ویدیو تکرار می‌کنیم. به‌ازای هر فریمی که خوانده شود، آن فریم وارد لیست می‌شود. حال از عناصر این لیست میانگین می‌گیریم، چون uint8 تنها مقادیر مثبت را شامل می‌شود و میانگین درست عکس را به ما می‌دهد، میانگین را تبدیل به uint8 می‌کنیم و نشان داده و ذخیره می‌کنیم.

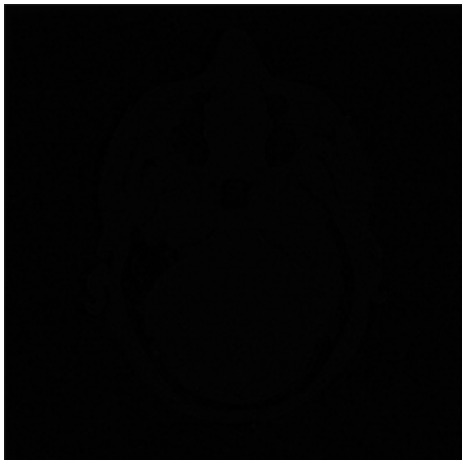
تصویر نهایی:



در قسمت ج، تصویرمان را تبدیل به int32 می‌کنیم و مثل قسمت قبل میانگین اختلاف بین تصویر اول و نهایی را محاسبه می‌کنیم. هم میانگین و هم انحراف‌معیار دوم از اولی کمتر هستند.

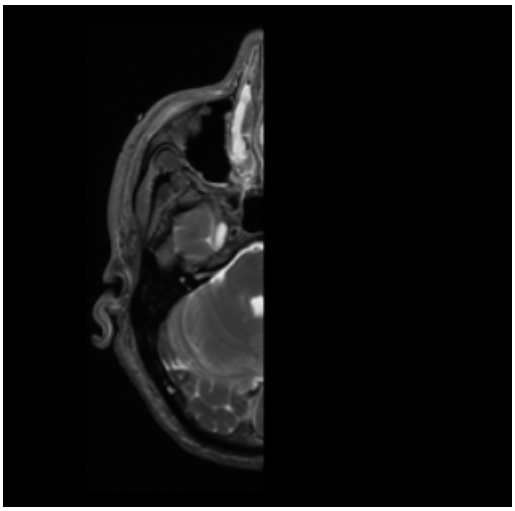
```
Average: 5.5281219482421875
Standard Deviation: 5.997128335074645

Average 2: 2.3891754150390625
Standard Deviation 2: 0.9314672519452099
```

نویز دوم:



می‌بینیم که نویز تقریباً سیاه است و یعنی اختلاف بین تصویر میانگین ویدیو و تصویر اول کمتر از مورد قبلی است.

در بخش د، برای اینکه تنها در بخش سفیدِ mask تصویر نهایی بیفتد، حلقه‌ی تودرتو for می‌سازیم و شرط می‌گذاریم که هرجا در عکس mask رنگ سفید داشتیم، در نقاط متناظر آن، تصویر نهایی روی عکس mask قرار بگیرد:



سؤال ٤

الف و ب: ابتدا باتوجه به خواسته‌های مسئله، رنگ عکس را تغییر داده و خروجی‌های موردنظر رو نشان می‌دهیم.

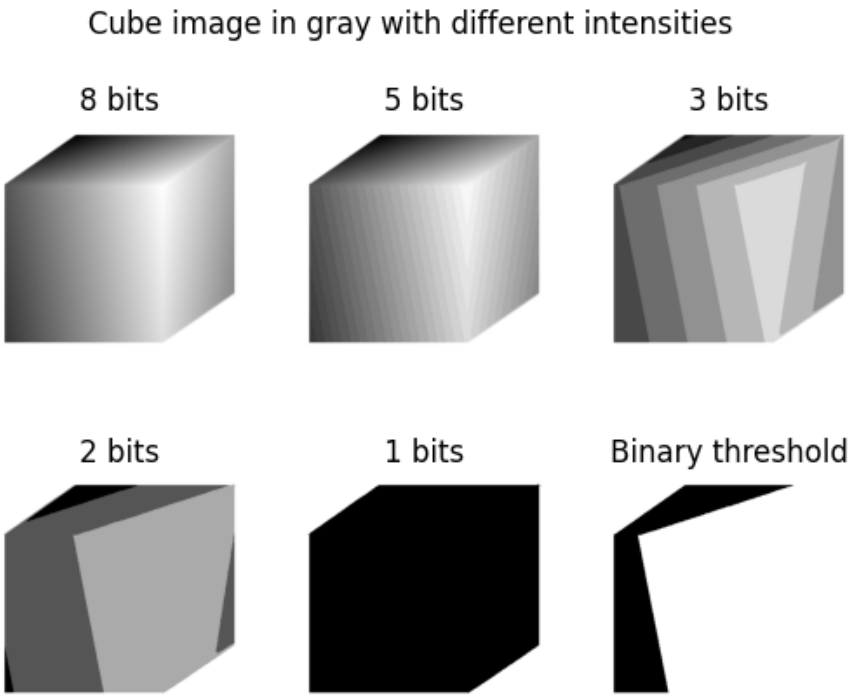
```
Image dimensions: (366, 409, 3)
Gray image dimensions: (366, 409)
Image datatype: uint8
```

در قسمت ج، تابعی با ورودی عکس و بیت می‌نویسیم که در آن ورودیِ عکس را که یک آرایه است، بر ۲۵۵ تقسیم کند و در 2^bit-1 ضرب کند و این‌گونه شدت موردنظر به‌دست می‌آید. از آنجا که با تقسیم بر ۲۵۵ تصویر ما تقریباً سیاه و عددی بین ۰ و ۱ است و بسیار کوچک است، آن را دوباره در ۲۵۵ ضرب و بر 2^bit-1 تقسیم می‌کنیم تا تصویر را بتوانیم ببینیم.

حال با ساختن subplot نمودار شدت روشنایی تصویر خاکستری را در بیت‌های موردنظر مسئله نشان می‌دهیم.

در قسمت ه، با استفاده از دستور threshholding، عکس را به آستانه‌ی ۱۲۷ الی ۲۵۵ باینری می‌کنیم (THRESH_BINARY) و در نمودار نشان می‌دهیم. در نهایت نیز آن را به‌صورت png ذخیره می‌کنیم.

نمودار خروجی:



سؤال ۵

در این سؤال نخست برای بخش‌کردن عکس، از یک حلقه‌ی تودرتو استفاده می‌کنیم که به‌تعداد ردیف‌ها و ستون‌های عکس به‌ترتیب تقسیم بر ۲ و ۳ (برای ۶ قسمت شدن عکس)، دستوری را اجرا کند. در این دستور هربار یک بخش از تصویر که ۱/۶ کل تصویر است و طوری تقسیم می‌شود که هر حرف در وسط بخش بیفتد، انتخاب و ذخیره می‌شود. برای نامگذاری آن‌ها هم از الگوریتمی استفاده کرده‌ام که از عدد ۱ تا ۶ نام هر بخش را قرار دهد و در قسمت print از fstring برای اعمال عدد در نام خروجی استفاده کرده‌ام.

در قسمت ب، با دستور resize در کتابخانه‌ی OpenCV اندازه‌ی تصویر را تغییر داده و دوبرابر می‌کنیم؛ سپس برای crop کردن، کل طول و عرض عکس بزرگ‌تر را منهای طول و عرض عکس کوچک‌تر کرده، بر ۲ تقسیم می‌کنیم و با دادن مختصات مناسب، بخش وسط عکس را انتخاب می‌کنیم.

در بخش‌های ج و د، اول ماتریسی می‌سازیم که ماتریس تبدیل موردنظر است (طبق جدول کتاب) و سپس با دستور warpAffine تغییر شکل مناسب را انجام می‌دهیم.

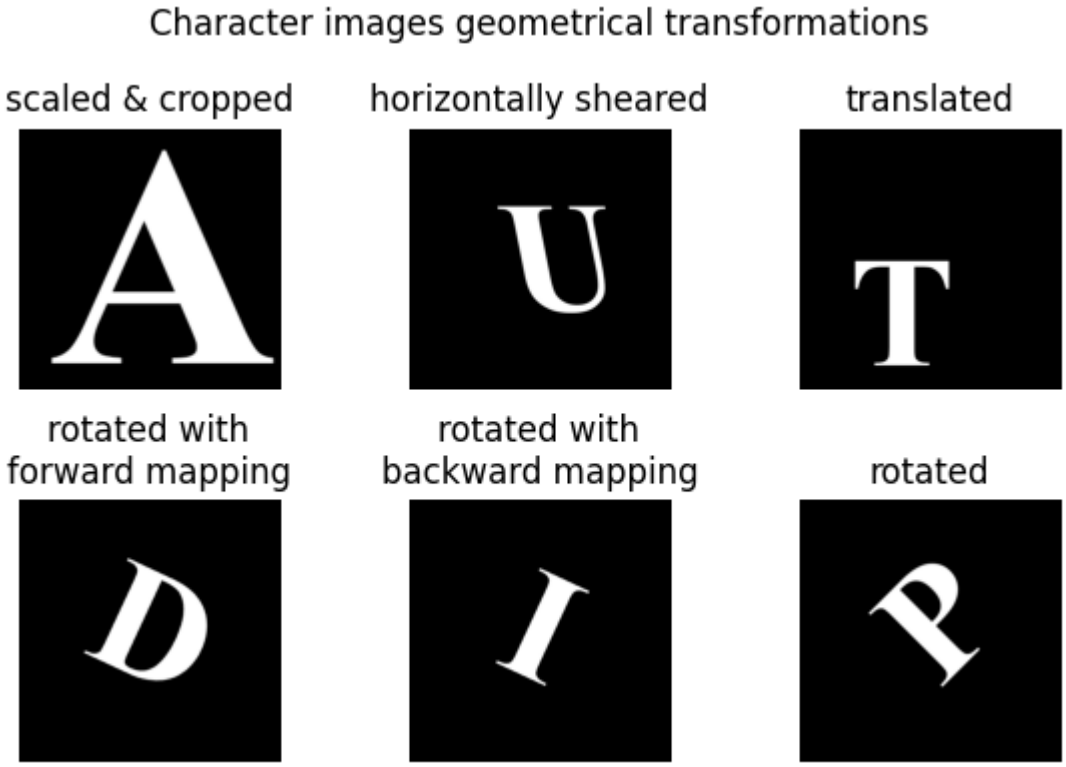
در بخش ه و و، ابتدا ماتریس صفری می‌سازیم که ابعاد آن با ابعاد عکس برابر باشد و نوع آن را uint8 می‌کنیم. سپس با انتخاب مرکز عکس به‌عنوان نقطه‌ی (0,0)، و تبدیل درجه به رادیان، با کمک توابع کتابخانه‌ی math ماتریس تبدیل برای چرخش مستقیم (forward) را می‌نویسیم (این ماتریس نیز در جدول تبدیل‌ها در مطالب تدریس‌شده قرار دارد). در انتها، یک شرط قرار می‌دهیم که به‌اندازه‌ی ابعاد تصویر، ماتریس صفر از فرمول ماتریس تبدیل، تبعیت کرده و آن را اجرا کند (در نقاط متناظر).

برای چرخش به‌روش backward یا معکوس نیز باید از تصویر forward به تصویر اصلی دست یابیم. ماتریس تبدیل آن تقریباً مشابه با روش مستقیم خواهد بود (علامت درایه‌ها تغییر می‌کند) و همان شرط قبلی را نیز اعمال می‌کنیم.

در بخش ز، با استفاده از تابع آماده‌ی getRotationMatrix2D در کتابخانه‌ی OpenCV و همچنین warpAffine عکس را به‌قدر خواسته‌شده می‌چرخانیم. آرگومان سوم تابع مذکور برای مشخص‌کردن میزان بزرگی یا zoom تصویر است که ما اینجا ۱ قرار می‌دهیم.

در قسمت ح، با ایجاد subplot، تصاویر حاصل هر بخش را در یک نمودار قرار می‌دهیم و با figsize متناسب، اندازه‌ی عکس‌ها و عنوان‌های‌شان را تنظیم می‌کنیم.

خروجی:


Character images geometrical transformations
scaled & cropped
horizontally sheared
translated
rotated with
forward mapping
rotated with
backward mapping
rotated